UP2ME: Univariate Pre-training to Multivariate Fine-tuning as a General-purpose Framework for Multivariate Time Series Analysis

Yunhao Zhang [1] **Minghao Liu** [1] **Shengyang Zhou** [1] **Junchi Yan** [1]

Abstract

Despite the success of self-supervised pre-training in texts and images, applying it to multivariate time series (MTS) falls behind tailored methods for tasks like forecasting, imputation and anomaly detection. We propose a general-purpose framework, named UP2ME (**U**nivariate **P**re-training to **M**ultivariate Fin**e**-tuning). It conducts task-agnostic pre-training when downstream tasks are unspecified. Once the task and setting (e.g. forecasting length) are determined, it gives sensible solutions with frozen pre-trained parameters, which has not been achieved before. UP2ME is further refined by fine-tuning. A univariate-to-multivariate paradigm is devised to address the heterogeneity of temporal and cross-channel dependencies. In univariate pre-training, univariate instances with diverse lengths are generated for Masked AutoEncoder (MAE) pre-training, discarding cross-channel dependency. The pre-trained model handles downstream tasks by formulating them into specific mask-reconstruction problems. In multivariate fine-tuning, it constructs a dependency graph among channels using the pre-trained encoder to enhance cross-channel dependency capture. Experiments on eight real-world datasets show its SOTA performance in forecasting and imputation, approaching task-specific performance in anomaly detection. Our code is available at https://github.com/Thinklab-SJTU/UP2ME.

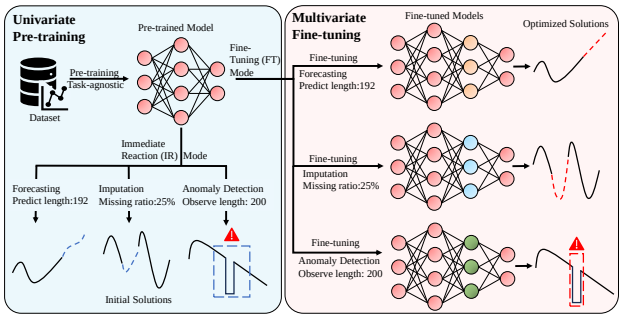


Figure 1: UP2ME Workflow: Given the dataset, UP2ME performs task-agnostic univariate pre-training. The resulting pre-trained model can execute immediate forecasting, imputation and anomaly detection across various settings without parameter modifications. Once the downstream task and its setting are determined, multivariate fine-tuning tailors UP2ME to the specific task for more accurate solutions.

1. Introduction

Recently, deep learning for multivariate time series (MTS) analysis has developed rapidly and has been applied to many tasks, such as forecasting, imputation and anomaly detection. Among these methods, task-specific ones tailored to tasks' characteristics constitute the most significant proportion. For example, models have been developed based on trend-season decomposition for forecasting (Wu et al., 2021), conditional diffusion for imputation (Tashiro et al., 2021) and association discrepancy for anomaly detection (Xu et al., 2022). Despite the effectiveness, selecting proper task-specific methods for different tasks can be exhausting. Even within the same task, when the setting (e.g. the forecasting length) changes, the model often needs to be retrained (Bi et al., 2023).

Until recently, a few MTS backbones emerged towards the goal of general-purpose analysis (Wu et al., 2023). While the same main architecture (except the output layer) is shared within tasks, the parameters need to be trained from scratch for each task and its corresponding setting.

Self-supervised pre-training, which pre-trains a model on downstream agnostic tasks and then fine-tunes it for spe-

[1]School of Artificial Intelligence & Department of Computer Science and Engineering & MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University. Correspondence to: Junchi Yan <yanjunchi@sjtu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

cific usages, is a promising way to achieve multi-task models. Following the success in natural language processing (NLP) (Devlin et al., 2019; Brown et al., 2020) and computer vision (CV) (Chen et al., 2020; He et al., 2022), pre-training methods have also been proposed for MTS, and mainly fall into contrastive learning and mask-reconstruction modeling. The former (Eldele et al., 2021; Yue et al., 2022) learns representation by discriminating pre-defined positive and negative pairs. The latter (Zerveas et al., 2021; Dong et al., 2023) masks a proportion of the data and uses the remaining to reconstruct masked parts. Different from NLP and CV, most previous pre-training methods for MTS can not rival carefully designed task-specific methods. Moreover, they only serve as model initializers and can not perform downstream tasks without parameter or architecture modification. Note that instead of pre-training on MTS data, a series of very recent works adapt pre-trained large language models (LLMs) to MTS tasks (Nate Gruver & Wilson, 2023; Zhou et al., 2023), which goes beyond the scope of this work.

To fill the gap, we propose a unified framework for MTS analysis, named UP2ME (**U**nivariate **P**re-training to **M**ultivariate **F**ine-tuning). As shown in Figure 1, when data is available but the downstream tasks and settings are undetermined, UP2ME performs task-agnostic pre-training. Without any parameter modification, the pre-trained model provides initial reasonable solutions to forecasting, imputation and anomaly detection in immediate reaction (IR) mode. Once the downstream tasks and settings are determined, UP2ME further adapts to the task and provides more accurate solutions in fine-tuning (FT) mode.

Temporal dependency captures relations over time, while cross-channel dependency captures relations between data with distinct physical meanings. The latter remains relatively stable due to underlying physical dynamics (Kipf et al., 2018). Inspired by the heterogeneity, we develop a univariate to multivariate paradigm. UP2ME omits cross-channel dependency during univariate pre-training, prioritizing temporal dependency. Channel dependency is subsequently incorporated during multivariate fine-tuning. Specifically, UP2ME uses variable window length and channel decoupling (see details in Sec. 2.1.1) to generate univariate instances with diverse lengths for MAE pre-training (He et al., 2022). Formulated as specific mask-reconstruction tasks, the pre-trained model can directly execute forecasting, imputation and anomaly detection without parameter or architecture modification. Freezing the pre-trained encoder and decoder during fine-tuning, UP2ME introduces trainable Temporal-Channel (TC) layers to capture cross-channel dependency and further refine temporal dependency. TC layers require a dependency graph representing relationships among channels, and this graph is constructed using the pre-trained frozen encoder. **The contributions are:**

1) Inspired by the heterogeneity of temporal and cross-channel dependency, we propose a general-purpose framework for MTS named UP2ME, where univariate pre-training concentrates on temporal dependency and cross-channel dependency is incorporated in multivariate fine-tuning.

2) In univariate pre-training, variable window length and channel decoupling are used to generate instances for MAE pre-training. The pre-trained model handles multiple tasks by formulating them into specific mask-reconstruction problems. In multivariate fine-tuning, UP2ME refines temporal dependency and constructs a graph among channels via the pre-trained encoder to capture cross-channel dependency.

3) We evaluate UP2ME on eight real-world datasets, addressing three downstream tasks: forecasting, imputation and anomaly detection. Using the original Transformer and Graph Transformer without inductive-biased architectures, pre-trained UP2ME(IR) with frozen parameters is comparable with previous state-of-the-art (SOTA) methods on several datasets. The fine-tuned UP2ME(FT) surpasses all previous task-specific, general-purpose and pre-training methods in forecasting and imputation, approaching task-specific performance in anomaly detection.

2. Methodology

As the overview in Figure 2 shows, we are given a multivariate time series dataset $\mathbf{X} \in \mathbb{R}^{T \times C}$, where $T > 0$ is the number of timestamps and $C > 1$ is the number of channels. In Section 2.1, UP2ME is pre-trained on the given dataset in the univariate setting to capture temporal dependency. The pre-trained model can provide initial solutions in IR mode. In Section 2.2, UP2ME is fine-tuned in the multivariate setting to capture cross-channel dependency and refine temporal dependency for more accurate solutions.

2.1. Univariate Pre-training

For pre-training, we specifically use MAE (He et al., 2022), which has proven effective for images. Compared with previous mask-reconstruction methods for time series, we propose two techniques for instance generation: variable window length and channel decoupling.

2.1.1. Instance Generation

Variable Window Length. Images are often cropped to a fixed size in CV (e.g. 224×224 for Imagenet), while for time series, it is unknown during pre-training and should be determined by the setting of downstream tasks (e.g. past and future window length for forecasting). To meet the uncertain requirements for window length, we make it variable during pre-training. Specifically, for each training step, we first randomly sample a window length L then generate a batch

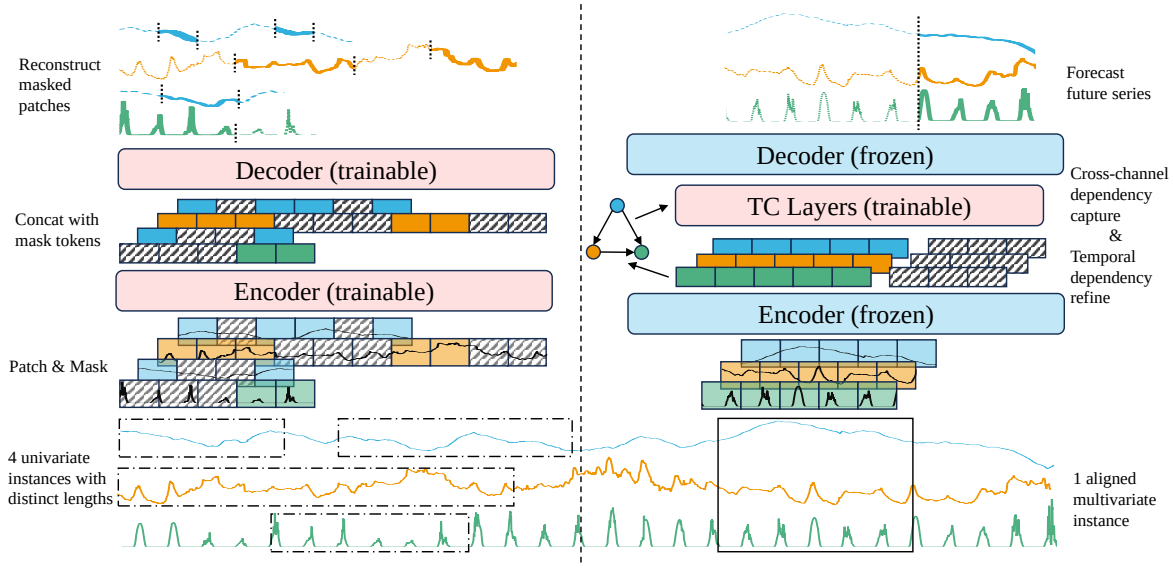


Figure 2: Overview of UP2ME framework. **Left: Univariate Pre-training.** Univariate instances are generated using variable window length and channel decoupling in Sec. 2.1.1. Generated instances are fed into the encoder and decoder for MAE pre-training in Sec. 2.1.2. Formulating downstream tasks as specific mask-reconstruction problems, UP2ME can give sensible solutions without parameter modification in Sec. 2.1.3 (right part without TC layers). **Right: Multivariate Fine-tuning** (forecasting in this example). The pre-trained frozen encoder encodes a multivariate series into latent tokens. The tokens are used to construct a dependency graph among channels in Sec. 2.2.1. Learnable Temporal-Channel (TC) layers which take constructed graph as input, are inserted before the frozen decoder for fine-tuning in Sec. 2.2.2.

of instances with this length:

$$n \sim \{N_{min}, \dots, N_{max}\}, L = nP \tag{1}$$

where P is the patch size to divide time series (Nie et al., 2023; Zhang & Yan, 2023), N_{min}, N_{max} are hyper-parameters for minimal/maximum patch number.

Channel Decoupling. To generate an instance of length L, previous works sample a timestamp t and get multivariate sub-series $\mathbf{X}_{t+1:t+L} \in \mathbb{R}^{L \times C}$ as an instance for model input. With channel decoupling, we independently sample the timestamp $t \sim \{0, \dots, T-L\}$ and channel index $c \sim \{1, \dots, C\}$. The univariate sub-series with its channel index $(\mathbf{X}_{t+1:t+L,c}, c)$ will be viewed as an instance.

Note that channel decoupling is different from channel independence (Nie et al., 2023). Channel independence is a technique to process multivariate sub-series: $\mathbf{X}_{t+1:t+L}$ is split into C univariate series and input to the model together. Though channels are processed independently, their co-occurrence reflects cross-channel dependency to some extent. With channel decoupling, we completely discard cross-channel dependency and only focus on temporal dependency during pre-training.

Another advantage of channel decoupling is its efficiency for high-dimensional data. For dataset with large C, we can pack $B (B << C)$ decoupled univariate series into a mini-batch instead of processing all channels at once. Experimental evaluation of pre-training overhead with and without channel decoupling is shown in Figure 5 of Appendix C.

2.1.2. Masked AutoEncoder Pre-training

With the above two instance generation techniques, the pre-training process is similar to MAE for image (He et al., 2022). For convinence, we use $(\mathbf{x}^{(pt)}, c), \mathbf{x}^{(pt)} \in \mathbb{R}^L, c \in \{1, \dots, C\}$ to represent a generated instance. The univariate series $\mathbf{x}^{(pt)}$ is first split into non-overlapping patches of length P: $\{\mathbf{x}_1^{(patch)}, \dots, \mathbf{x}_N^{(patch)}\}$, where $\mathbf{x}_i^{(patch)} \in \mathbb{R}^P, N = L/P$. Then each patch is embedded into a token with d_{model} dimensions through linear projection, added with learnable positional and channel embeddings:

$$\mathbf{h}_i^{(patch)} = \mathbf{W}^{(emb)} \mathbf{x}_i^{(patch)} + \mathbf{p}_i^{(pos)} + \mathbf{v}_c^{(ch)} \tag{2}$$

where $\mathbf{W}^{(emb)} \in \mathbb{R}^{d_{model} \times P}$ is the projection matrix, $\mathbf{p}_i^{(pos)}, \mathbf{v}_c^{(ch)} \in \mathbb{R}^{d_{model}}$ are positional embedding for index i and channel embedding for channel c. We randomly sample a subset of patches with ratio α to mask, here we use $\mathcal{M}$ to represent indices of masked patches and $\mathcal{U}$ for unmasked patches. Unmasked patches are input to an encoder to capture temporal dependency:

$$\mathbf{h}_i^{(enc)} = \text{Encoder}(\bigcup_{j \in \mathcal{U}} \mathbf{h}_j^{(patch)})_i, \forall i \in \mathcal{U} \tag{3}$$

These encoded unmasked patches, together with learnable tokens indicating masked patches, are input to a decoder for masked patch reconstruction:

$$\begin{aligned}\mathbf{h}_i^{(dec-in)} &= \begin{cases} \mathbf{u}^{(mask)} + \mathbf{p}_i^{(pos)} + \mathbf{v}_c^{(ch)} & i \in \mathcal{M} \\ \mathbf{W}^{(proj)}\mathbf{h}_i^{(enc)} & i \in \mathcal{U} \end{cases} \\ \mathbf{h}_i^{(dec-out)} &= \text{Decoder}\left([\mathbf{h}_1^{(dec-in)}, \ldots, \mathbf{h}_N^{(dec-in)}]\right)_i \\ \mathbf{x}_i^{(rec)} &= \mathbf{W}^{(rec)}\mathbf{h}_i^{(dec-out)}, \forall i \in \mathcal{M} \end{aligned} \tag{4}$$

where $\mathbf{u}^{(mask)} \in \mathbb{R}^{d_{model}}$ denotes the presence of a masked patch, $\mathbf{W}^{(proj)} \in \mathbb{R}^{d_{model} \times d_{model}}$ projects latent tokens from encoder space to decoder space, $[\cdot,\cdot]$ denotes concatenation, $\mathbf{W}^{(rec)} \in \mathbb{R}^{P \times d_{model}}$ projects latent tokens to original patches. The encoder/decoder are both composed of standard Transformer layers. The normalization technique, RevIN (Kim et al., 2022), is used to reduce distribution shift.

Mean squared error (MSE) between the reconstructed and ground truth masked patches is used as pre-training loss:

$$\mathcal{L} = \frac{1}{P|\mathcal{M}|} \sum_{i \in \mathcal{M}} \|\mathbf{x}_i^{(patch)} - \mathbf{x}_i^{(rec)}\|_2^2 \tag{5}$$

2.1.3. Immediate Reaction Mode

Different from the original MAE for images where the decoder is removed after pre-training, UP2ME preserves both the pre-trained encoder and decoder for potential downstream tasks. Formulating different downstream tasks as specific mask-reconstruction problems, UP2ME can perform immediate forecasting, anomaly detection and imputation with frozen parameters. As the immediate reaction (IR)[1] mode only utilizes temporal dependency and acts similarly on each channel, we only describe the computation process for a single channel in this section.

Forecasting. To forecast future time series $\mathbf{x}^{(future)} \in \mathbb{R}^{L_f}$ based on its past $\mathbf{x}^{(past)} \in \mathbb{R}^{L_p}$, we view past series as unmasked patches and future series as masked patches[2]:

$$\begin{aligned}\mathcal{M}^{forecast} &= \{i | \frac{L_p}{P} < i \leq \frac{L_p + L_f}{P}\} \\ \mathcal{U}^{forecast} &= \{i | 1 \leq i \leq \frac{L_p}{P}\}\end{aligned} \tag{6}$$

Imputation. Point-wise missing can be easily handled by traditional interpolation methods. Moreover, missing patterns in real world are often structured and appear consecutively (Tashiro et al., 2021). We focus on this more challenging scenario where continuous blocks of data are missing. Given observed data $\mathbf{x}^{(imp)} \in \mathbb{R}^{L_{imp}}$ and a mask for missing positions $\mathbf{m} \in [0,1]^{L_{imp}}$ ($m_i = 1$ if i-th value is missing), we patch $\mathbf{m}$ into $\{\mathbf{m}_1^{(patch)}, \ldots, \mathbf{m}_{L_{imp}/P}^{(patch)}\}$ using the same patching process for $\mathbf{x}^{(imp)}$. Patches containing at least one missing point are viewed as masked patches and fully-observed patches are viewed as unmasked:

$$\begin{aligned}\mathcal{M}^{imputate} &= \{i \,\Big|\, \|\mathbf{m}_i^{(patch)}\|_0 \geq 1\} \\ \mathcal{U}^{imputate} &= \{i \,\Big|\, \|\mathbf{m}_i^{(patch)}\|_0 = 0\}\end{aligned} \tag{7}$$

Anomaly Detection. Due to the rarity and irregularity of anomalies, it is much more difficult to reconstruct them from other parts of the series than normal points. Based on this, to detect anomalies from observed series $\mathbf{x}^{(det)} \in \mathbb{R}^{L_{det}}$, we iteratively mask each patch and use other unmasked patches to reconstruct it, MSE between reconstructed series and original series is used as the anomaly score:

$$\begin{aligned}&\text{for } i = 1, \ldots, \frac{L_{det}}{P}: \\ &\quad \mathcal{M}_i^{detect} = \{i\} \quad \mathcal{U}_i^{detect} = \{1, \ldots, \frac{L_{det}}{P}\} \backslash \{i\} \\ &\widetilde{\mathbf{x}}^{(det)} = \left[\mathbf{x}_1^{(rec)}, \ldots, \mathbf{x}_{L_{det}/P}^{(rec)}\right] \\ &\text{AnomoalyScore}(t) = |x_t^{(det)} - \widetilde{x}_t^{(det)}|^2, 1 \leq t \leq L_{det}\end{aligned} \tag{8}$$

In each iteration, we use $\mathcal{U}_i^{detect}$ as unmasked patches to reconstruct $\mathcal{M}_i^{detect}$, resulting in $\mathbf{x}_i^{(rec)} \in \mathbb{R}^P$. These reconstructed patches are concatenated into $\widetilde{\mathbf{x}}^{(det)} \in \mathbb{R}^{L_{det}}$ to compute anomaly score. For multivariate data, we compute the anomaly score for each channel and use the average across channels as the final anomaly score.

Despite distributions and ratios of mask in three downstream tasks being different from those in pre-training, experiments in Section 3 show that our IR mode generalizes well and is on par with some task-specific methods.

2.2. Multivariate Fine-tuning

In fine-tuning, the downstream task and the corresponding setting are given. UP2ME takes a multivariate instance $\mathbf{x}^{(ft)} \in \mathbb{R}^{L_{ft} \times C}$ as input and performs forecasting/detection/imputation. Specifically, we freeze parameters of the pre-trained encoder and decoder while incorporating learnable Temporal-Channel (TC) layers between them. The main function of the TC layer is to capture dependency among channels and, incidentally, adjust temporal dependency to reduce the gap between pre-training and downstream tasks.

[1] We use this new terminology instead of zero-shot as the pre-training and inference are within the same dataset, unlike a strict zero-shot setting where models operate on completely unseen data.

[2] Without loss of generality, we assume all windows are divisible by P; otherwise, we can pad them to proper length.

2.2.1. Sparse Dependency Graph Construction

A straightforward method for capturing cross-channel dependency is to employ self-attention across C channels, which corresponds to constructing a fully connected dependency graph. However, the $O(C^2)$ complexity limits its applicability to potentially high-dimensional datasets (Zhang & Yan, 2023). Hence, it is necessary to build a sparse graph that preserves most dependency with fewer edges to guide cross-channel dependency capturing.

Since the encoder has acquired meaningful representations through pre-training, we leverage it for sparse graph construction. Each channel of $\mathbf{x}^{(ft)}$ is first patched and then independently input to the encoder to get latent tokens, denoted as $\{\bigcup_{i\in\mathcal{U}_c}\mathbf{h}_{i,c}^{(enc)}\}_{c=1}^{C}$, where $\mathcal{U}_c$ indicates unmasked patches in channel c, $\mathbf{h}_{i,c}^{(enc)}$ indicates the i-th encoded patch in channel c. Then we use these latent tokens to construct a dependency graph among channels:

$$\begin{aligned}\mathbf{h}_c^{(ch)} &= \text{Max\_Pooling}(\bigcup_{i\in\mathcal{U}_c}\mathbf{h}_{i,c}^{(enc)}), \forall c\in\{1,\dots,C\}\\ A_{c,c'} &= \frac{\langle\mathbf{h}_c^{(ch)},\mathbf{h}_{c'}^{(ch)}\rangle}{\|\mathbf{h}_c^{(ch)}\|_2\|\mathbf{h}_{c'}^{(ch)}\|_2}, \forall c,c'\in\{1,\dots,C\}\\ \mathbf{E} &= \text{topK}(\mathbf{A},rC)\wedge\text{KNN}(\mathbf{A},r)\end{aligned} \tag{9}$$

We use max pooling to get a token $\mathbf{h}_c^{(ch)}$ representing channel c. The correlation matrix $\mathbf{A}$ is defined as the pairwise cosine similarity of these tokens. The intersection of rC largest elements and r-nearest neighbors of each channel is used as the final graph $\mathbf{E}$, with r as a constant hyperparameter.

Note that it is common to measure dependency between sentences or words using cosine similarity of latent embeddings from pre-trained models (Reimers & Gurevych, 2019; Ren et al., 2023). For MTS, Shao et al. (2022) also employs the cosine similarity of latent tokens from a separate pre-trained model to guide spatial-temporal graph neural network learning. Our graph construction process differs from previous works for MTS which use channel independence (Nie et al., 2023), low-rank approximation (Zhang & Yan, 2023), statistics or learning methods (Wu et al., 2020). Using the pre-trained encoder, we construct a non-linear and sparse graph with at most rC edges. The additional computational cost incurred by the construction is minimal, as there is no learning component.

2.2.2. Temporal-Channel layer

Concatenating encoded patches with tokens indicating unmasked patches (Eq. 4), we get $\mathbf{H}^{(TC-in)}\in\mathbb{R}^{N^{(dec)}\times C\times d_{model}}$, where $N^{(dec)}$ is the number of patches in each channel for decoding. Before input to the decoder, $\mathbf{H}^{(TC-in)}$ and the constructed dependency graph $\mathbf{E}$ pass through $K\geq 1$ Temporal-Channel (TC) layers to capture cross-channel dependency and adjust temporal dependency. With few inductive biases, our TC layer contains a standard Transformer layer for temporal dependency and a standard Graph Transformer layer (Dwivedi & Bresson, 2021) for cross-channel dependency :

$$\begin{aligned}&\mathbf{H}^{(ch,0)} = \mathbf{H}^{(TC-in)}\\ &\text{for } k=1,\dots,K:\\ &\quad\mathbf{H}_{:,c}^{(time,k)} = \text{Transformer}(\mathbf{H}_{:,c}^{(ch,k-1)}), \forall c\\ &\quad\mathbf{H}_{i,:}^{(ch,k)} = \text{Graph\_Transformer}(\mathbf{H}_{i,:}^{(time,k)},\mathbf{E}), \forall i\\ &\mathbf{H}^{(dec-in)} = \mathbf{H}^{(TC-out)} = \mathbf{H}^{(ch,K)}\end{aligned} \tag{10}$$

where $\mathbf{H}_{:,c}^{(ch,k-1)}\in\mathbb{R}^{N^{(dec)}\times d_{model}}$ denotes tokens of all steps in channel c and $\mathbf{H}_{i,:}^{(time,k)}\in\mathbb{R}^{C\times d_{model}}$ denotes all channels at step i. Passing through several learnable TC layers, the final $\mathbf{H}^{(dec-in)}\in\mathbb{R}^{N^{(dec)}\times C\times d_{model}}$ is input to the frozen decoder to get solutions for downstream tasks. A Graph Transformer layer functionally equals a standard Transformer layer using the graph structure as the attention weight mask, but is more efficient on sparse graphs (Dwivedi & Bresson, 2021). The overall computation complexity of a TC layer is $O(CN^2)+O(rCN)=O(CN^2)$, which is linear w.r.t C thanks to the constructed sparse graph, making UP2ME scalable to high dimensional data.

3. Experiments

We conduct experiments on eight real-world datasets: **1)ETTm1, 2)Weather, 3)Electricity, 4)Traffic, 5)SMD, 6)PSM, 7)SWaT, 8)GECCO.** On each dataset, we perform three different downstream tasks: forecasting, imputation and anomaly detection[3]. We vary the specific settings for tasks, such as prediction length for forecasting and missing ratio for imputation, etc. We pre-train one UP2ME for each dataset as the base model and fine-tune it to adapt to different downstream tasks and settings. Results of two UP2ME modes are reported: **1)UP2ME(IR)**: immediate reaction mode which directly provides initial solutions with the pre-trained model; **2)UP2ME(FT)**: fine-tuning mode which adapts to specific downstream tasks and settings. For each task, three categories of methods are compared: 1) task-specific methods; 2) general-purpose methods; 3) pre-training methods. Detailed setup is shown in Appendix A.

3.1. Main Results

Forecasting. We select **PatchTST** (Nie et al., 2023), **DLinear** (Zeng et al., 2023), **Crossformer** (Zhang & Yan, 2023), **FEDformer** (Zhou et al., 2022) as task-specific methods; **TimesNet** (Wu et al., 2023) as the general-purpose method;

[3]Anomaly detection is only evaluated on the last four datasets as the first four lack ground truth anomaly annotations.

TS2Vec (Yue et al., 2022) and **SimMTM** (Dong et al., 2023) as pre-training methods. Each method predicts future series with different lengths (L_f). Evaluation metrics are Mean Square Error (mSE) and Mean Absolute Error (mAE) [4].

The results are shown in Table 1. Without any adjustment to model parameters, UP2ME(IR) is comparable with previous SOTA methods on ETTm1, Electricity, Traffic and GECCO datasets. This indicates that our instance generation techniques enable the model to generalize to downstream tasks where the mask distribution and ratio are different from pre-training. After fine-tuning, UP2ME makes more accurate forecasting and outperforms previous SOTAs on all datasets. Note that our UP2ME only uses standard Transformer and Graph Transformer layers, without inductive biased architecture designs e.g. flattened projection (PatchTST, TimesNet, SimMTM), trend-season decomposition (DLinear, FEDformer), hierarchical encoder-decoder (Crossformer) or frequency domain enhancement (FEDformer, TimesNet).

Imputation. We select **SAITS** (Du et al., 2023), **GRIN** (Cini et al., 2022), **LI** (Linear Interpolation) and **SI** (Spline Interpolation) as task-specific methods; general-purpose and pre-training methods are same as those for forecasting. We evaluate performances on different missing ratio levels and use mSE and mAE as evaluation metrics.

Table 2 shows that UP2ME(IR) outperforms most previous baselines on Electricity, Traffic, SMD and GECCO and also achieves comparable results on ETTm1 and SWaT. It is worth mentioning that the architecture and parameters are the same as those for forecasting, showing UP2ME's capability to handle multiple tasks. Equipped with cross-channel dependency after fine-tuning, UP2ME(FT) outperforms all other methods over a large margin on 7 out of 8 datasets.

Anomaly Detection. We select **DCdetector** (Yang et al., 2023), **AnomalyTrans** (Xu et al., 2022), **iForest** (Liu et al., 2012), **OCSVM** (Schölkopf et al., 2001) as task-specific methods and use the same general and pre-training methods as forecasting and imputation. Following Yang et al. (2023); Xu et al. (2022), we use the train and validation set to select a threshold and label anomalies with it for F1-score evaluation. Moreover, to mitigate the impact of threshold selection for a more comprehensive comparison, we also threshold at every possible point to evaluate the Average Precision (AP) (Manning, 2009). The widely-used segment adjustment strategy (Shen et al., 2020; Xu et al., 2022; Yang et al., 2023) is utilized for F1-score and AP evaluation.

Table 3 shows that two task-specific SOTA methods, DCdetector and AnomalyTrans, perform better on the first three datasets, but our UP2ME still outperforms traditional task-specific, general-purpose and pre-training methods and approaches task-specific methods. While on the more challenging GECCO dataset with various types of anomalies (Yang et al., 2023), UP2ME outperforms task-specific methods over a large margin, indicating the superiority of our univariate pre-training to multivariate fine-tuning paradigm.

[4]For distinction, we use "MAE" to denote Masked AutoEncoder and "mAE" for Mean Absolute Error.

3.2. Model Analysis

Ablations of Pre-training. As Figure 3(a) shows, training from scratch is less effective than pre-training and then fine-tuning, though the network architectures are the same. Without variable window length, the IR mode can not handle varying prediction lengths, thus performs poorly. Channel decoupling slightly improves both modes and contributes more to fine-tuning. Also, channel decoupling is indispensable for pre-training on high-dimensional datasets (e.g. Traffic ($C = 862$)); otherwise, the computational overhead would be unaffordable (see experiments in Appendix C)

Mask Ratio α in Pre-training. Figure 3(b) shows the influence of mask ratio in pre-training. A lower mask ratio ($\leq 30\%$) would result in decreased performance of IR mode. While an excessively high mask ratio ($\geq 70\%$) has negative impacts on both IR and FT modes. The optimal ratios are $40\% \sim 60\%$, within which both two modes perform well and outperform training from scratch (0.369). The default mask ratio used in our main experiments is 50%.

Graph Construction in Fine-tuning. Figure 3(c) shows that channel independence without graph structure performs the worst. The graph constructed by the pre-trained encoder outperforms those via random processes, Pearson correlation and Euclidean distance. We failed to evaluate Dynamic Time Wrapping (DTW) due to its quadratic complexity and non-parallelizability. UP2ME approaches the theoretical upper bound, i.e. full connection, with small computational overhead (see memory occupancy in Figure 3(d)). Note that the full connection is unaffordable for high-dimensional datasets (see Figure 6 in Appendix C). Figure 4(a) shows a correlation matrix for graph construction on Weather, where channels #4, #7, #9 and #10 are highly correlated and form a connected community. Actually, they are related to dew point, water vapor and humidity, indicating that our graph construction is reasonable.

Hyper-parameter r in Fine-tuning. Figure 3(d) shows that increasing hyper-parameter r in fine-tuning improves performance, but also increases memory occupancy. Exceeding a certain threshold, the improvement becomes marginal, but memory occupancy rises rapidly. This indicates that UP2ME can preserve the most important correlations with a relatively sparse graph. To strike a balance between performance and efficiency, the default r is set to $r = \min(10, \lceil 0.5C \rceil)$.

Varying Past Window for Forecasting. Zeng et al. (2023)

Table 1: mSE/mAE of forecasting. The prediction length L_f is set to $\{96, 192, 336, 720\}$ for the first four datasets and $\{50, 100, 150, 200\}$ for the last four. Results are averaged over 4 different lengths. Bold/underline indicates the best/second. Our methods are marked in gray. OOM: out-of-memory problem. See Table 5 in Appendix B for the full results.

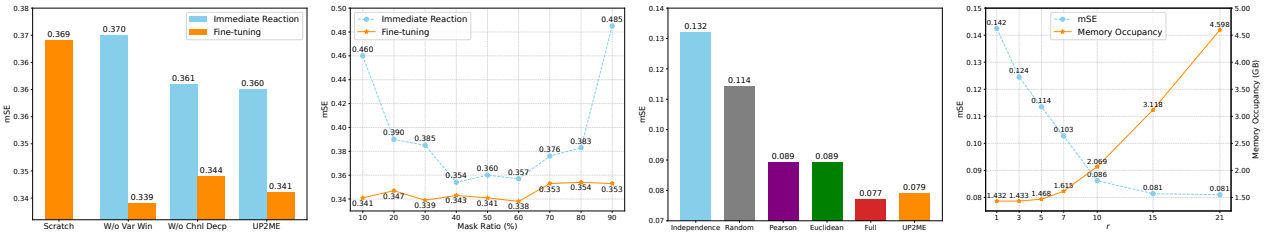
Methods	Task-Specific								General		Pre-Training							
	PatchTST		DLinear		Crossformer		FEDformer		TimesNet		TS2Vec		SimMTM		UP2ME(IR)		UP2ME(FT)	
Metric	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE
ETTm1	0.353	0.382	0.358	0.379	0.443	0.461	0.427	0.447	0.393	0.408	0.787	0.651	0.350	0.384	0.360	**0.372**	**0.341**	0.374
Weather	0.228	0.264	0.244	0.297	0.239	0.299	0.311	0.365	0.247	0.283	0.261	0.330	0.232	0.270	0.266	0.288	**0.221**	**0.260**
Electricity	0.164	0.255	0.168	0.265	0.205	0.306	0.239	0.349	0.194	0.293	0.377	0.451	OOM		0.165	0.252	**0.155**	**0.245**
Traffic	0.395	0.265	0.436	0.300	0.528	0.292	0.658	0.413	0.622	0.332	0.973	0.569	OOM		0.401	0.257	**0.390**	**0.253**
SMD	0.893	0.174	0.992	0.199	0.925	0.185	1.020	0.236	0.995	0.188	1.289	0.427	0.894	0.187	0.924	0.201	**0.872**	**0.169**
PSM	0.303	0.310	0.314	0.341	0.377	0.326	0.326	0.330	0.301	0.311	0.687	0.580	0.315	0.325	0.602	0.499	**0.290**	**0.300**
SWaT	0.217	0.066	0.354	0.168	0.237	0.110	0.263	0.103	0.226	0.063	10.082	1.559	0.248	0.138	0.292	0.093	**0.210**	**0.062**
GECCO	1.656	0.322	1.703	0.457	2.637	0.657	1.735	0.383	1.655	0.314	2.469	0.771	1.615	0.331	1.476	0.328	**1.413**	**0.299**

Table 2: mSE/mAE of imputation. Results are averaged over 4 missing ratio settings ($0\% \sim 12.5\%$, $12.5\% \sim 25\%$, $25\% \sim 37.5\%$, $37.5\% \sim 50\%$). See Table 6 in Appendix B for the full results.

Methods	Task-Specific								General		Pre-Training							
	SAITS		GRIN		LI		SI		TimesNet		TS2Vec		SimMTM		UP2ME(IR)		UP2ME(FT)	
Metric	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE
ETTm1	0.201	0.278	0.492	0.496	0.912	0.577	2.046	1.060	0.172	0.266	0.275	0.353	0.328	0.380	0.275	0.318	**0.128**	**0.221**
Weather	0.103	0.160	0.232	0.306	0.179	0.198	1.016	0.741	0.113	0.166	0.125	0.201	0.171	0.228	0.150	0.191	**0.079**	**0.108**
Electricity	0.211	0.319	0.313	0.399	1.277	0.860	1.759	1.058	0.140	0.256	0.227	0.324	OOM		0.107	0.204	**0.097**	**0.193**
Traffic	0.573	0.311	0.509	0.291	2.253	0.994	2.696	1.092	0.508	0.276	0.559	0.299	OOM		0.338	0.223	**0.294**	**0.197**
SMD	0.865	0.207	1.148	0.346	1.289	0.167	3.215	0.870	0.877	0.178	1.217	0.372	0.842	0.155	0.839	0.166	**0.756**	**0.103**
PSM	0.601	0.450	0.821	0.486	0.232	0.257	3.026	1.132	0.302	0.330	1.497	0.733	0.225	0.269	0.340	0.334	**0.144**	**0.197**
SWaT	2.929	0.754	6.140	0.872	0.186	0.055	13.579	1.564	0.240	0.102	6.091	1.257	0.155	0.066	0.192	0.064	**0.121**	**0.045**
GECCO	3.828	1.142	5.091	1.216	1.634	**0.250**	11.720	2.063	1.778	0.412	6.065	1.401	1.515	0.314	**1.405**	0.299	1.468	0.290

Table 3: F1-score/AP (in %) of anomaly detection. See Table 7 in Appendix B for the full results.

Methods	Task-Specific								General		Pre-Training							
	DCdetector		AnomalyTrans		iForest		OCSVM		TimesNet		TS2Vec		SimMTM		UP2ME(IR)		UP2ME(FT)	
Metric	F1	AP	F1	AP	F1	AP	F1	AP	F1	AP	F1	AP	F1	AP	F1	AP	F1	AP
SMD	84.40	82.75	**90.98**	93.49	54.92	80.65	67.23	73.42	83.09	90.59	74.13	87.79	83.01	**93.91**	82.69	93.90	83.31	93.58
PSM	**97.50**	98.73	97.37	98.80	90.82	96.61	86.53	96.86	90.53	99.70	87.44	96.48	93.37	99.73	96.05	99.75	97.16	**99.76**
SWaT	**96.52**	**99.57**	95.01	98.92	37.36	91.46	61.61	88.55	90.83	97.37	26.20	82.12	89.29	97.38	92.89	97.83	93.85	98.07
GECCO	31.71	31.87	34.45	48.54	26.37	38.92	52.41	62.08	46.45	**68.53**	16.77	37.76	47.32	63.30	55.91	65.47	**63.39**	65.09



(a) (b) (c) (d)

Figure 3: Analysis of pre-training and fine-tuning. (a) Forecasting mSE of ablations in pre-training on ETTm1. (b) Forecasting mSE against mask ratios α in pre-training on ETTm1. (c) Imputation mSE of different graph constructions in fine-tuning on Weather. (d) mSE and memory occupancy against r in fine-tuning on Weather with $25\% \sim 37.5\%$ missing.

argues that many methods fail to leverage longer past windows for better forecasting. Figure 4(b) shows that besides TimesNet, performances of other models show an increasing trend while the window length increases from 120 to 720. Further increasing it to 1440, performances of PatchTST, SimMTM and UP2ME(FT) get worse, while UP2ME(IR) can further utilize the expanded receptive field to improve forecasting. As UP2ME(IR) does not require re-training for different lengths, we can efficiently adjust the past window length to get better performance in practice.

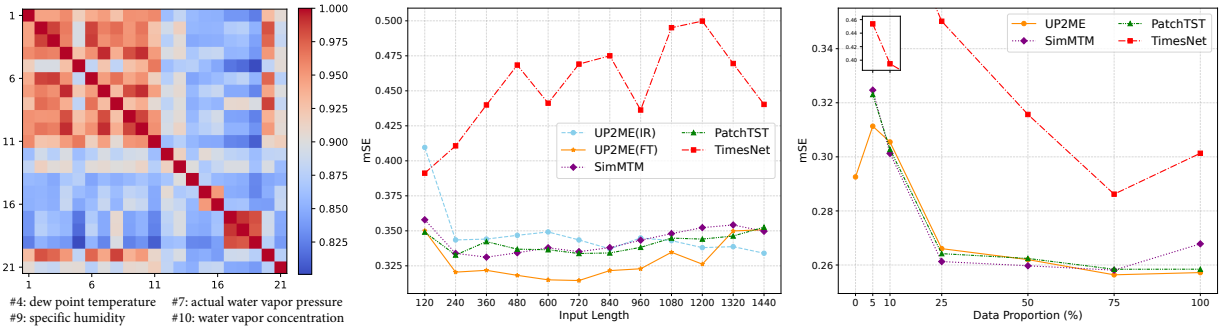


Figure 4: (a) A correlation matrix $\mathbf{A}$ utilized for graph construction during imputation fine-tuning on the Weather dataset. (b) Forecasting mSE against varying past window length on ETTm1, the prediction length is set to 192. (c) Forecasting mSE on ETTm2 dataset against varying available data proportions. TimesNet and PatchTST are trained from scratch, SimMTM and UP2ME are pre-trained on ETTm1 and fine-tuned on the available ETTm2 data. 0% for UP2ME stands for IR mode.

Adaption to Limited Data Scenarios Following Dong et al. (2023), we evaluate the performance of UP2ME in limited data scenario on ETTm2 dataset in Figure 4(c). With enough data, PatchTST, SimMTM and UP2ME achieve similar performance. However, in data-limited scenarios ($\leq 10\%$), UP2ME(IR) without fine-tuning achieves the best performance, showing our UP2ME has a certain degree of transferability, which is critical to limited data scenarios.

4. Related Works

4.1. Task-specific Methods for Time Series

Forecasting. Early works employ RNNs (Flunkert et al., 2017), CNNs (Lea et al., 2017), and GNNs (Wu et al., 2020) as backbones. Later, Transformers were adapted. Li et al. (2019b); Zhou et al. (2021) use sparse attention. Liu et al. (2022) introduces a hierarchical module that captures features at multiple scales. Wu et al. (2021); Zhou et al. (2022); Huang et al. (2023) introduce frequency domain features into Transformers. Nie et al. (2023); Zhang & Yan (2023) divide series into patches and propose channel independence/dependence to model cross-channel dependency. Besides Transformers, recent works also employ linears (Zeng et al., 2023) and MLPs (Ekambaram et al., 2023).

Imputation. Imputation fills the missing values in MTS caused by sensor issues, etc. Early methods train RNNs with supervised learning (Cao et al., 2018; Yoon et al., 2018), followed by works using Transformers (Du et al., 2023). With the development of deep generative models, VAEs (Ramchandran et al., 2021; Fortuin et al., 2020), GANs (Liu et al., 2019; Luo et al., 2019) and Diffusion models (Tashiro et al., 2021) have also been introduced for MTS imputation. Besides temporal dependency, Cini et al. (2022) incorporates GNNs to capture cross-channel dependency.

Anomaly Detection. Anomaly detection aims to identify unusual patterns or outliers caused by irregular events, etc. Early works introduce probabilistic clustering (Tariq et al., 2019) and stochastic process (Su et al., 2019) into RNNs. Zhao et al. (2020); Deng & Hooi (2021) use GNN to capture cross-channel dependency to improve detection. Li et al. (2019a); Zhou et al. (2019); Li et al. (2022) use deep generative models. Yu & Sun (2020) models it as a decision process solved by RL. More recent works introduce prior knowledge into Transformers: Xu et al. (2022) discriminates abnormal points via their local and global association patterns, Yang et al. (2023) utilizes contrastive consistency and performs detection via the difference of two views.

General-purpose Architecture. Wu et al. (2023) proposes such a general-purpose architecture for MTS analysis. It transforms data into a set of 2D tensors based on its multiple periods and utilizes CNNs to extract features. Changing the output layer and training criterion, it can perform multiple tasks with the same main architecture.

Despite the effectiveness of task-specific methods, selecting and switching between them for different tasks is challenging. Although Wu et al. (2023) maintains the main architecture, it is still required to train its parameters from scratch for each task and each setting.

4.2. Pre-training Methods for Time Series

Beyond CV and NLP, pre-training methods have also been devised for MTS recently and can be roughly classified into contrastive learning and mask-reconstruction.

Contrastive Learning. It learns representations by discriminating between positive and negative pairs. Methods differ in how to define pairs for MTS. Mohsenvand et al. (2020); Eldele et al. (2021) use augmentations, such as amplitude scale and time shift, to generate positive pairs.Woo et al. (2022) decomposes time series into trend and season

components and performs contrastive learning on them respectively. Yang & Hong (2022) utilizes features in time and spectral domains. Zhang et al. (2022) further requires representations in two domains to be consistent. Yue et al. (2022); Franceschi et al. (2019) view overlapped timestamps of different sub-series as positive pairs. Hyvarinen & Morioka (2017); Agrawal et al. (2022) define a pair of consecutive sub-series as a positive pair, and Tonekaboni et al. regard nearby sub-series in the time domain as positive.

Mask-Reconstruction. It learns representations by masking a proportion of the data and reconstructing it via the remaining unmasked parts (Devlin et al., 2019; Brown et al., 2020; He et al., 2022). As for MTS, Zerveas et al. (2021) masks consecutive sub-series in the original space and uses a Transformer with a linear output layer to reconstruct them. Nie et al. (2023) divides time series into patches and masks patches in the latent space. Following MAE in CV (He et al., 2022), decoupled encoder-decoders are used to perform point-wise (Li et al., 2023) and patch-wise (Shao et al., 2022) mask-reconstruction. Dong et al. (2023) reconstructs the original time series from multiple randomly masked series. Instead of reconstruction, Cheng et al. (2023) proposes masked codeword classification and representation regression for learning in latent space.

Besides the above two, there are alternative paradigms (Malhotra et al., 2017; Lyu et al., 2018; Sarkar & Etemad, 2020). Unlike NLP (Radford et al., 2019) and CV (Bai et al., 2023), to our knowledge, existing pre-training methods, cannot solve downstream tasks without parameter modification, and they can hardly rival those carefully designed task-specific methods. Recent efforts involve adapting pre-trained large language models (LLMs) for time series tasks instead of pre-training models with MTS datasets (Nate Gruver & Wilson, 2023; Zhou et al., 2023; Chang et al., 2024), which extends beyond the scope of this work.

5. Conclusion, Limitations and Future Works

We have proposed UP2ME as a general-purpose framework for MTS analysis. Technically, it utilizes a univariate pre-training to multivariate fine-tuning paradigm which captures temporal dependency during pre-training and incorporates cross-channel dependency during fine-tuning. Functionally, the pre-trained UP2ME provides initial sensible solutions to forecasting, imputation and anomaly detection without parameter modification, which has not been achieved before. Further accuracy is achieved through fine-tuning.

Due to the different data formats and the feasibility of conducting classification without parameter modification in IR mode, UP2ME currently does not support classification. Unlike NLP foundation models, which are pre-trained on multiple datasets and can execute downstream tasks in zero-shot mode on unseen data, our approach pre-trains one model on a single dataset and adapts it to multiple downstream tasks within the same dataset, consistent with prior works for MTS (Yue et al., 2022; Dong et al., 2023). But our univariate-to-multivariate paradigm is suitable for large-scale multi-dataset pre-training: 1) univariate pre-training ensures that varying numbers of channels across datasets do not hinder pre-training; 2) when the downstream dataset is determined, multivariate fine-tuning further incorporates channel dependency to enhance performance. However, building a foundation model that can adapt to diverse datasets for MTS remains an open question for future exploration.

Acknowledgement

This work was in part supported by National Natural Science Foundation of China (62222607, 92370201, 72342023).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agrawal, M. N., Lang, H., Offin, M., Gazit, L., and Sontag, D. Leveraging time irreversibility with order-contrastive pre-training. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Bai, Y., Geng, X., Mangalam, K., Bar, A., Yuille, A., Darrell, T., Malik, J., and Efros, A. A. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Chang, C., Wang, W.-Y., Peng, W.-C., and Chen, T.-F. Llm4ts: Aligning pre-trained llms as data-efficient time-

series forecasters. *arXiv preprint arXiv:2308.08469*, 2024.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020.

Cheng, M., Liu, Q., Liu, Z., Zhang, H., Zhang, R., and Chen, E. Timemae: Self-supervised representations of time series with decoupled masked autoencoders. *arXiv preprint arXiv:2303.00320*, 2023.

Cini, A., Marisca, I., and Alippi, C. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Deng, A. and Hooi, B. Graph neural network-based anomaly detection in multivariate time series. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2019.

Dong, J., Wu, H., Zhang, H., Zhang, L., Wang, J., and Long, M. Simmtm: A simple pre-training framework for masked time-series modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Du, W., Côté, D., and Liu, Y. Saits: Self-attention-based imputation for time series. *Expert Systems with Applications*, 2023.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Ekambaram, V., Jati, A., Nguyen, N., Sinthong, P., and Kalagnanam, J. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2023.

Eldele, E., Ragab, M., Chen, Z., Wu, M., Keong, C., Kwoh, X. L., and Guan, C. Time-series representation learning via temporal and contextual contrasting. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Flunkert, V., Salinas, D., and Gasthaus, J. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2017.

Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. Gp-vae: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Huang, Q., Shen, L., Zhang, R., Ding, S., Wang, B., Zhou, Z., and Wang, Y. CrossGNN: Confronting noisy multivariate time series via cross interaction refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Hyvarinen, A. and Morioka, H. Nonlinear ica of temporally dependent stationary sources. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations (ICLR)*, 2022.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *International Conference on Machine Learning (ICML)*, 2018.

Lea, C. S., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Li, D., Chen, D., Jin, B., Shi, L., Goh, J., and Ng, S.-K. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks (ICANN)*, 2019a.

Li, L., Yan, J., Wen, Q., Jin, Y., and Yang, X. Learning robust deep state space for unsupervised anomaly detection in contaminated time-series. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

Li, Z., Wang, P., Rao, Z., Pan, L., and Xu, Z. Ti-MAE: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2012.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations (ICLR)*, 2022.

Liu, Y., Yu, R., Zheng, S., Zhan, E., and Yue, Y. Naomi: Non-autoregressive multiresolution sequence imputation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Luo, Y., Zhang, Y., Cai, X., and Yuan, X. E2gan: End-to-end generative adversarial network for multivariate time series imputation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Lyu, X., Hueser, M., Hyland, S. L., Zerveas, G., and Raetsch, G. Improving clinical predictions through unsupervised time series representation learning. *Machine Learning for Health (ML4H) Workshop at NeurIPS*, 2018.

Ma, J. and Perkins, S. Time-series novelty detection using one-class support vector machines. In *International Joint Conference on Neural Networks (IJCNN)*, 2003.

Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. Timenet: Pre-trained deep recurrent neural network for time series classification. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2017.

Manning, C. D. *An introduction to information retrieval*. Cambridge university press, 2009.

Mohsenvand, M. N., Izadi, M. R., and Maes, P. Contrastive representation learning for electroencephalogram classification. In *Machine Learning for Health (ML4H)*, 2020.

Nate Gruver, Marc Finzi, S. Q. and Wilson, A. G. Large Language Models Are Zero Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Nie, Y., H. Nguyen, N., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Ramchandran, S., Tikhonov, G., Kujanpää, K., Koskinen, M., and Lähdesmäki, H. Longitudinal variational autoencoder. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Ren, S., Wu, Z., and Zhu, K. Q. Emo: Earth mover distance optimization for auto-regressive language modeling. *arXiv preprint arXiv:2310.04691*, 2023.

Sarkar, P. and Etemad, A. Self-supervised ecg representation learning for emotion recognition. *IEEE Transactions on Affective Computing (TAC)*, 2020.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. Estimating the support of a high-dimensional distribution. *Neural computation*, 2001.

Shao, Z., Zhang, Z., Wang, F., and Xu, Y. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2022.

Shen, L., Li, Z., and Kwok, J. Timeseries anomaly detection using temporal hierarchical one-class network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., and Pei, D. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

Tariq, S., Lee, S., Shin, Y., Lee, M. S., Jung, O., Chung, D., and Woo, S. S. Detecting anomalies in space using multivariate convolutional lstm with mixtures of probabilistic pca. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tonekaboni, S., Eytan, D., and Goldenberg, A. Unsupervised representation learning for time series with temporal neighborhood coding. In *International Conference on Learning Representations (ICLR)*.

Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2022.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations (ICLR)*, 2023.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., and Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2020.

Xu, J., Wu, H., Wang, J., and Long, M. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations (ICLR)*, 2022.

Yang, L. and Hong, S. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *International Conference on Machine Learning (ICML)*, 2022.

Yang, Y., Zhang, C., Zhou, T., Wen, Q., and Sun, L. Dcdetector: Dual attention contrastive representation learning for time series anomaly detection. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2023.

Yoon, J., Zame, W. R., and van der Schaar, M. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 2018.

Yu, M. and Sun, S. Policy-based reinforcement learning for time series anomaly detection. *Engineering Applications of Artificial Intelligence*, 2020.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. Ts2vec: Towards universal representation of time series. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. A transformer-based framework for multivariate time series representation learning. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2021.

Zhang, X., Zhao, Z., Tsiligkaridis, T., and Zitnik, M. Self-supervised contrastive pre-training for time series via time-frequency consistency. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhang, Y. and Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., and Zhang, Q. Multivariate time-series anomaly detection via graph attention network. In *IEEE International Conference on Data Mining (ICDM)*, 2020.

Zhou, B., Liu, S., Hooi, B., Cheng, X., and Ye, J. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning (ICML)*, 2022.

Zhou, T., Niu, P., Wang, X., Sun, L., and Jin, R. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A. Details of Experiments

A.1. Datasets

We conduct experiments on the following eight datasets following Wu et al. (2023); Yang et al. (2023):

1) **ETTm1** records 7 crucial indicators of an electricity transformer, including high useful load and oil temperature, etc. Data points are recorded every 15 minutes and the entire dataset covers a period of two years. We use data from the first 20 months and split it into train/valid/test sets with a ratio of 0.6:0.2:0.2.
2) **Weather** records 21 meteorological indicators in Beutenberg, including air temperature and dewpoint, etc. Data points are recorded every 10 minutes and the entire dataset covers the whole 2020 year. Train/valid/test sets are split by a ratio of 0.7:0.1,0.2.
3) **Electricity** records hourly electricity consumption (in kW) of 321 clients from 2012 to 2014. Train/valid/test sets are split by a ratio of 0.7:0.1,0.2.
4) **Traffic** contains San Francisco Bay area freeways' road occupancy rates measured by 862 sensors from 2016.07 to 2018.06. Data points are recorded every hour and train/valid/test sets are split by a ratio of 0.7:0.1,0.2.
5) **SMD** records 38 indicators of a server machine from a large Internet company, including CPU load, network usage, etc. The entire dataset covers a period of 5 weeks.
6) **PSM** records 25 indicators of application server nodes at eBay, including CPU and memory utilization, etc. The training set covers 13 weeks, followed by 8 weeks for testing.
7) **SWaT** records indicators measured by 51 sensors of a modern industrial control system.
8) **GECCO** records 9 indicators about the drinking-water quality, including PH value and amount of chlorine dioxide, etc. Data points are recorded every minute.

The first four datasets are often used in forecasting and imputation tasks and our train/val/test splits are the same as Wu et al. (2021); Nie et al. (2023). The last four datasets are often used in anomaly detection tasks. Each of them is originally divided into two parts for training and testing. Only testing parts are labeled with ground truth anomaly annotations. Following Wu et al. (2023), we further split the original training set by a ratio of 0.8:0.2 for our training and validation. Table 4 shows the statistical characteristics of each dataset.

The first 7 datasets are publicly available at `https://github.com/thuml/Time-Series-Library` and GECCO is available at `https://github.com/DAMO-DI-ML/KDD2023-DCdetector`.

Table 4: Statistical characteristics of datasets used in experiments.

Dataset	Channels	Train Timestamps	Valid Timestamps	Test Timestamps	Field
ETTm1	7	34,560	11,520	11,520	electricity transformer indicators
Weather	21	36,887	5,270	10,539	meteorological weather indicators
Electricity	321	18,412	2,632	5,260	electricity consumption
Traffic	862	12,280	1,756	3,508	road occupancy rates
SMD	38	566,724	141,681	708,420	server machine indicators
PSM	25	105,984	26,497	87,841	application server indicators
SWaT	51	396,000	99,000	449,919	industrial control system indicators
GECCO	9	55,408	13,852	69,261	drinking-water quality

A.2. General-purpose and Pre-training Baselines

General-purpose Baseline We select TimesNet (Wu et al., 2023) as the general-purpose baseline. TimesNet discovers multiple periods of MTS by Fast Fourier Transform and then transforms time series into a set of 2D tensors according to these periods. These 2D tensors are processed by inception blocks for feature extraction. TimesNet conducts multiple tasks by changing the output layer and training criterion. The source code is available at `https://github.com/thuml/Time-Series-Library`.

Pre-training Baselines We select the following 2 pre-training methods for comparison:

1) **TS2Vec** (Yue et al., 2022) is a contrastive learning based pre-training method for MTS. It uses timestamp masking and random cropping to generate contexts and representations at the same timestamp in two contexts are viewed as positive pairs for contrastive learning. Max pooling is used to perform hierarchical contrastive learning at multiple scales. The source code is available at https://github.com/yuezhihan/ts2vec.

2) **SimMTM** (Dong et al., 2023) is a mask-reconstruction based pre-training method for MTS. It reconstructs the original time series with multiple masked series with different masks. A contrastive-style manifold constraint is further used to learn series-wise representation. The source code is available at https://github.com/thuml/SimMTM.

A.3. Experiments for Forecasting

Task-specific Baselines We select the following 4 task-specific baselines for forecasting:

1) **PatchTST** (Nie et al., 2023) is the current SOTA for MTS forecasting. It divides the input MTS into patches and uses a Transformer encoder with channel independence to process these patches. The encoder output is then flattened and linearly projected to desired lengths for final forecasting. The source code is available at https://github.com/yuqinie98/PatchTST.

2) **DLinear** (Zeng et al., 2023) is a light-weight MTS forecaster. It decomposes input MTS into trend and season components and applies a linear model on each component respectively. The source code is available at https://github.com/cure-lab/LTSF-Linear.

3) **Crossformer** (Zhang & Yan, 2023) is a Transformer that explicitly captures cross-channel dependency for MTS forecasting. It divides MTS into patches like PatchTST and proposes a two-stage attention mechanism to capture both temporal and cross-channel dependency. A hierarchical encoder-decoder is constructed for forecasting. The source code is available at https://github.com/Thinklab-SJTU/Crossformer.

4) **FEDformer** (Zhou et al., 2022) is a Transformer that uses seasonal-trend decomposition with frequency-enhanced blocks to capture temporal dependency for MTS forecasting. The source code is available at https://github.com/MAZiqing/FEDformer.

Setup We use the common protocol for MTS forecasting (Zhou et al., 2021; Wu et al., 2021; Nie et al., 2023): train/val/test sets are normalized with the mean and standard deviation of the training set. On ETTm1, Weather, Electricity and Traffic, we forecast the future $L_f = \{96, 192, 337, 720\}$ steps with the default past window length set to 336. Considering that FEDformer and TimesNet may prefer a shorter past window, we select past lengths from $\{96, 192, 336, 720\}$ via grid search for these two methods. On datasets containing anomaly points, i.e. SMD, PSM, SWaT and GECCO, we forecast the future $L_f = \{50, 100, 150, 200\}$ steps with the default past window length set to 400. Similarly, past lengths for FEDformer and TimesNet are chosen from $\{50, 100, 200, 400\}$ via grid search. All experiments are repeated 3 times and the averaged Mean Square Error (mSE) and Mean Absolute Error (mAE) are reported.

A.4. Experiments for Imputation

Task-specific Baselines We select the following 4 task-specific baselines for Imputation:

1) **SAITS** (Du et al., 2023) is a self-attention-based method for MTS imputation. The model is optimized via two joint tasks: 1) missing series imputation to fill the missing values and 2) observed series reconstruction to help the model converge to the distribution of the dataset. The source code is available at https://github.com/WenjieDu/SAITS.

2) **GRIN** (Cini et al., 2022) proposes an encoder-decoder that utilizes RNN and GNN for MTS imputation. It uses a bidirectional architecture to process the input MTS. Furthermore, a two-stage imputation process is devised, where the first stage utilizes the features extracted GNN, followed by the second stage utilizing representations from RNN to refine the first-stage imputation. The source code is available at https://github.com/Graph-Machine-Learning-Group/grin.

3) **LI (Linear Interpolation)** is a traditional method for MTS imputation which fills the missing values by constructing a linear curve between adjacent observed data points. We use its Pandas implementation, which is available at https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html.

4) **SI (Spline Interpolation)** is a traditional method for MTS imputation which fills the missing values with piecewise low-degree polynomial curves rather than a single, high-degree polynomial curve. We also use its Pandas implementation, which is available at https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html.

Setup Considering that point-wise missing is easy to solve with traditional interpolation methods and missing patterns in the real world are often structured and appear consecutively (Tashiro et al., 2021), we conduct experiments on block-wise imputation: we set the observed window length for imputation to $L_{imp} = 600$ for all 8 datasets. On each dataset, we evaluate imputation performance at 4 different missing range levels: $\{0\% \sim 12.5\%, 12.5\% \sim 25\%, 25\% \sim 37.5\%, 37.5\% \sim 50\%\}$. To generate an instance with block-wise missing, we first sample a mask ratio from the corresponding range level for each channel. Subsequently, a randomly sampled consecutive block of this ratio is masked in each channel. Similar to forecasting, all experiments are repeated 3 times and the averaged Mean Square Error (mSE) and Mean Absolute Error (mAE) are reported.

A.5. Experiments for Anomaly Detection

Task-specific Baselines We select the following 4 task-specific baselines for Anomaly Detection:

1) **DCdetector** (Yang et al., 2023) is a contrastive representation-based attention model for MTS anomaly detection. The motivation is that it is easier to learn shared patterns among different views of normal points than abnormal ones. Thus, DCdetector learns permutation invariant representations and uses the representation discrepancy between two views as the anomaly score. The source code is available at https://github.com/DAMO-DI-ML/KDD2023-DCdetector.

2) **AnomalyTrans** (Xu et al., 2022) is an association discrepancy-based Transformer for MTS anomaly detection. The motivation is that it is difficult for abnormal points to build nontrivial associations with the whole series. Therefore, the discrepancy between the learned attention distribution and an adjacent-concentrated Gaussian distribution is used as the anomaly score. The source code is available at https://github.com/thuml/Anomaly-Transformer.

3) **iForest (Isolation Forest)** (Liu et al., 2012) is a traditional algorithm for anomaly detection. The main idea is that anomalies can be isolated more easily than normal points. By employing a tree-based structure with random feature selection and splits, anomalies tend to be isolated with shorter paths. Anomaly score is defined as the depth of the leaf containing this point, which is equivalent to the number of splittings required to isolate it. To adapt iForest for time series, we fold multiple adjacent timestamps around a center point to form a sample following Ma & Perkins (2003). We use the sklearn implementation of iForest, which is available at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

4) **OCSVM (One-Class Support Vector Machine)** (Schölkopf et al., 2001) is a traditional algorithm for anomaly detection. It constructs a hyperplane that encapsulates the majority normal data points in a high-dimensional space. Samples lying outside this hyperplane are considered potential anomalies and signed distance to the hyperplane is used as the anomaly score. Similar to iForest, we fold adjacent timestamps to adapt OCSVM to time series and use the sklearn implementation, which is available at https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html.

Setup We use the common protocol for MTS anomaly detection (Xu et al., 2022; Wu et al., 2023; Yang et al., 2023): models undergoes unsupervised learning on training and validation sets. The testing sets are divided into non-overlapped sub-series of length 100 for metric evaluation. All experiments are repeated 3 times and the averaged metrics are reported.

We evaluate two anomaly detection metrics: **F1-score** and **Average Precision (AP)** (Manning, 2009). F1-score is a commonly used metric for MTS anomaly detection (Xu et al., 2022; Wu et al., 2023; Yang et al., 2023) and it needs a threshold to transform anomaly scores into labels. Following Yang et al. (2023), the threshold is chosen by ensuring that δ fraction of points in training and validation sets surpass the specified threshold. Then, the selected threshold is used to label

the testing set. δ is set to 0.5% for SMD, 1% for PSM and SWaT, 2% for GECCO. Considering anomaly scores of different methods have different physical meanings and the F1-score is sensitive to the threshold selection process, we also evaluate AP as a more comprehensive metric. To evaluate AP, we threshold at every possible value and get M precision-recall pairs $\{(P_i, R_i)\}_{i=1}^{M}, \forall j : R_j \leq R_{j+1}$, and AP is computed as $AP = \sum_{i=2}^{M}(R_i - R_{i-1})P_i$. Note that the selected threshold for F1-score corresponds to a single pair in $\{(P_i, R_i)\}_{i=1}^{M}$.

The widely-used segment adjustment strategy (Shen et al., 2020; Xu et al., 2022; Yang et al., 2023; Wu et al., 2023) is used for F1-score and AP evaluation: if a single timestamp within a contiguous anomaly segment is accurately detected, the entire anomalies within that same segment are also deemed correctly identified. This strategy relies on the fact that, in time series analysis, human operators often prioritize the overall anomaly segment rather than the point-wise anomaly. Hence, triggering an alert at any point within a contiguous anomaly segment is deemed acceptable.

A.6. Implementation Details

Pre-training For all datasets, the encoder consists of 4 standard Transformer layers and the decoder consists of 1 standard Transformer layer. The dimension of hidden state d_{model} is set to 256 and the head number of multi-head attention is set to 4. For ETTm1, Weather, Electricity and Traffic, we set patch size, min and max patch number to $P = 12, N_{min} = 20, N_{max} = 200$; for datasets containing anomaly points, i.e. SMD, PSM, SWaT and GECCO, we set $P = 10, N_{min} = 5, N_{max} = 100$.

As for parameter optimization, the mask ratio is set to $\alpha = 0.5$. The batch size is set to 256. Adam optimizer with a constant learning rate of 1e-4 is used for optimization. The maximum training step is set to 500,000. The model undergoes validation every 5,000 steps on the validation set, and if the validation loss fails to decrease over 10 consecutive validations, the pre-training process is terminated early. The model with the lowest validation loss is retained.

Fine-tuning For all 3 downstream tasks on all 8 datasets, we insert one TC layer between the frozen encoder and decoder for fine-tuning. d_{model} and the head number are the same as pre-training. Adam optimizer with a constant learning rate is used for optimization. The learning rate is set to 1e-5 for forecasting and anomaly detection and 5e-4 for imputation. The maximum training epoch is set to 20. If the validation loss fails to decrease over 3 consecutive validations, the training process is stopped early. The hyper-parameter r in Equation 9 for graph construction is set as $r = \min(10, \lceil 0.5C \rceil)$, i.e. $r = 4$ for ETTm1, $r = 5$ for GECCO and $r = 10$ for all other 6 datasets.

All deep learning methods, including our UP2ME, are implemented in PyTorch and run on 2 NVIDIA GeForce RTX 3090 GPUs with 24GB memory.

B. Full Results

Due to the space limitation in the main text, we place the full results of Section 3.1 here. mSE/mAE evaluation of forecasting on different prediction lengths is shown in Table 5; mSE/mAE evaluation of imputation on different missing levels is shown in Table 6; Precision, recall, F1-score and AP evaluation of anomaly detection is shown in Table 7.

Table 5: Full mSE/mAE evaluation of forecasting. The prediction length L_f is set to {96, 192, 336, 720} for the first four datasets and {50, 100, 150, 200} for the last four. Bold/underline indicates the best/second. Our method is marked in gray. OOM indicates out-of-memory.

Methods		Task-Specific								General		Pre-Training							
		PatchTST		DLinear		Crossformer		FEDformer		TimesNet		TS2Vec		SimMTM		UP2ME(IR)		UP2ME(FT)	
Metric		mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE
ETTm1	96	0.290	0.340	0.299	0.343	0.322	0.387	0.377	0.422	0.329	0.367	0.673	0.566	0.292	0.349	0.292	**0.330**	**0.287**	0.334
	192	0.330	0.369	0.335	0.365	0.391	0.422	0.403	0.438	0.382	0.400	0.732	0.623	0.329	0.373	0.337	**0.359**	**0.323**	0.361
	336	0.369	0.394	0.370	0.386	0.452	0.472	0.447	0.457	0.415	0.422	0.823	0.681	0.364	0.393	0.378	0.383	**0.351**	**0.381**
	720	0.422	0.426	0.427	0.423	0.607	0.562	0.480	0.472	0.444	0.443	0.917	0.735	0.414	0.422	0.435	**0.418**	**0.405**	0.419
	Avg	0.353	0.382	0.358	0.379	0.443	0.461	0.427	0.447	0.393	0.408	0.787	0.651	0.350	0.384	0.360	**0.372**	**0.341**	0.374
Weather	96	0.150	0.198	0.174	0.234	0.154	0.227	0.231	0.322	0.163	0.217	0.179	0.262	0.158	0.211	0.180	0.222	**0.144**	**0.192**
	192	0.195	0.241	0.216	0.274	0.198	0.270	0.283	0.353	0.212	0.259	0.225	0.305	0.200	0.249	0.231	0.265	**0.187**	**0.237**
	336	0.250	0.284	0.261	0.312	0.276	0.338	0.340	0.383	0.269	0.304	0.282	0.349	0.249	0.286	0.289	0.306	**0.237**	**0.277**
	720	0.319	0.334	0.326	0.367	0.329	0.362	0.389	0.404	0.344	0.353	0.360	0.403	0.319	0.336	0.366	0.360	**0.316**	**0.331**
	Avg	0.228	0.264	0.244	0.297	0.239	0.299	0.311	0.365	0.247	0.283	0.261	0.330	0.232	0.270	0.266	0.288	**0.221**	**0.260**
Electricity	96	0.130	0.223	0.141	0.238	0.156	0.264	0.211	0.326	0.165	0.269	0.366	0.446	OOM		0.130	0.220	**0.123**	**0.214**
	192	0.149	0.241	0.155	0.252	0.184	0.292	0.226	0.339	0.185	0.286	0.370	0.447			0.149	0.238	**0.143**	**0.233**
	336	0.166	0.260	0.170	0.269	0.213	0.312	0.243	0.354	0.197	0.299	0.379	0.452			0.169	0.257	**0.160**	**0.250**
	720	0.210	0.299	0.205	0.302	0.268	0.356	0.276	0.378	0.227	0.320	0.394	0.458			0.213	0.293	**0.193**	**0.282**
	Avg	0.164	0.255	0.168	0.265	0.205	0.306	0.239	0.349	0.194	0.293	0.377	0.451	OOM		0.165	0.252	**0.155**	**0.245**
Traffic	96	0.365	0.250	0.413	0.288	0.485	0.273	0.629	0.402	0.588	0.315	0.942	0.563	OOM		0.369	0.241	**0.358**	**0.235**
	192	0.383	0.258	0.425	0.293	0.506	0.282	0.635	0.397	0.614	0.327	0.940	0.558			0.392	0.251	**0.382**	**0.249**
	336	0.396	0.264	0.439	0.301	0.538	0.300	0.669	0.419	0.631	0.338	0.959	0.563			0.404	0.257	**0.393**	**0.254**
	720	0.435	0.287	0.468	0.319	0.583	0.315	0.698	0.432	0.655	0.348	1.051	0.592			0.439	0.278	**0.426**	**0.274**
	Avg	0.395	0.265	0.436	0.300	0.528	0.292	0.658	0.413	0.622	0.332	0.973	0.569	OOM		0.401	0.257	**0.390**	**0.253**
SMD	50	0.843	0.140	0.916	0.158	0.863	0.140	0.948	0.206	1.012	0.156	1.236	0.404	0.836	0.152	0.837	0.150	**0.818**	**0.132**
	100	0.880	0.165	0.971	0.189	0.901	0.165	0.999	0.227	0.961	0.186	1.270	0.418	0.876	0.177	0.899	0.189	**0.858**	**0.160**
	150	0.911	0.185	1.020	0.214	0.943	0.212	1.044	0.247	0.996	0.201	1.312	0.437	0.913	0.199	0.958	0.221	**0.890**	**0.181**
	200	0.940	0.205	1.059	0.236	0.991	0.223	1.088	0.266	1.012	0.208	1.338	0.448	0.949	0.219	1.003	0.243	**0.923**	**0.201**
	Avg	0.893	0.174	0.992	0.199	0.925	0.185	1.020	0.236	0.995	0.188	1.289	0.427	0.894	0.187	0.924	0.201	**0.872**	**0.169**
PSM	50	0.196	0.241	0.198	0.258	0.261	0.273	0.221	0.265	0.197	0.244	0.512	0.501	0.213	0.260	0.378	0.379	**0.191**	**0.237**
	100	0.274	0.293	0.271	0.316	0.353	0.323	0.292	0.314	0.267	0.293	0.658	0.567	0.281	0.306	0.575	0.490	**0.261**	**0.283**
	150	0.337	0.334	0.354	0.371	0.411	0.365	0.361	0.351	0.336	0.333	0.753	0.609	0.348	0.347	0.687	0.545	**0.325**	**0.323**
	200	0.403	0.374	0.433	0.418	0.482	0.399	0.432	0.390	0.406	0.373	0.825	0.641	0.418	0.387	0.766	0.583	**0.385**	**0.359**
	Avg	0.303	0.310	0.314	0.341	0.377	0.326	0.326	0.330	0.301	0.311	0.687	0.580	0.315	0.325	0.602	0.499	**0.290**	**0.300**
SWaT	50	0.132	0.043	0.196	0.102	0.156	0.082	0.163	0.078	0.139	**0.040**	11.366	1.525	0.170	0.064	0.168	0.056	**0.123**	**0.040**
	100	0.198	0.060	0.298	0.149	0.211	0.102	0.244	0.100	0.204	0.058	9.835	1.504	0.230	0.080	0.274	0.087	**0.186**	**0.056**
	150	0.247	0.075	0.422	0.198	0.261	0.115	0.300	0.110	0.260	0.072	9.547	1.566	0.275	0.094	0.331	0.105	**0.243**	**0.070**
	200	0.290	0.087	0.499	0.224	0.319	0.141	0.345	0.124	0.301	0.084	9.581	1.644	0.318	0.107	0.395	0.123	**0.285**	**0.082**
	Avg	0.217	0.066	0.354	0.168	0.237	0.110	0.263	0.103	0.226	0.063	10.082	1.559	0.248	0.138	0.292	0.093	**0.210**	**0.062**
GECCO	50	1.380	0.259	1.449	0.338	2.241	0.531	1.619	0.317	1.443	0.247	2.071	0.665	1.404	0.270	1.288	0.263	**1.225**	**0.232**
	100	1.572	0.308	1.712	0.464	2.564	0.622	1.769	0.357	1.607	0.302	2.395	0.747	1.585	0.322	1.450	0.322	**1.394**	**0.287**
	150	1.812	0.348	1.796	0.506	2.900	0.724	1.776	0.413	1.671	0.339	2.624	0.811	1.683	0.352	1.531	0.352	**1.489**	**0.326**
	200	1.860	0.374	1.853	0.522	2.845	0.751	1.778	0.446	1.902	0.367	2.787	0.862	1.788	0.380	1.635	0.376	**1.543**	**0.353**
	Avg	1.656	0.322	1.703	0.457	2.637	0.657	1.735	0.383	1.655	0.314	2.469	0.771	1.615	0.331	1.476	0.328	**1.413**	**0.299**

Table 6: Full mSE/mAE evaluation of imputation. Missing ratios are set into 4 levels: $\{0\% \sim 12.5\%, 12.5\% \sim 25\%, 25\% \sim 37.5\%, 37.5\% \sim 50\%\}$. Bold/underline indicates the best/second. Our method is marked in gray. OOM indicates out-of-memory.

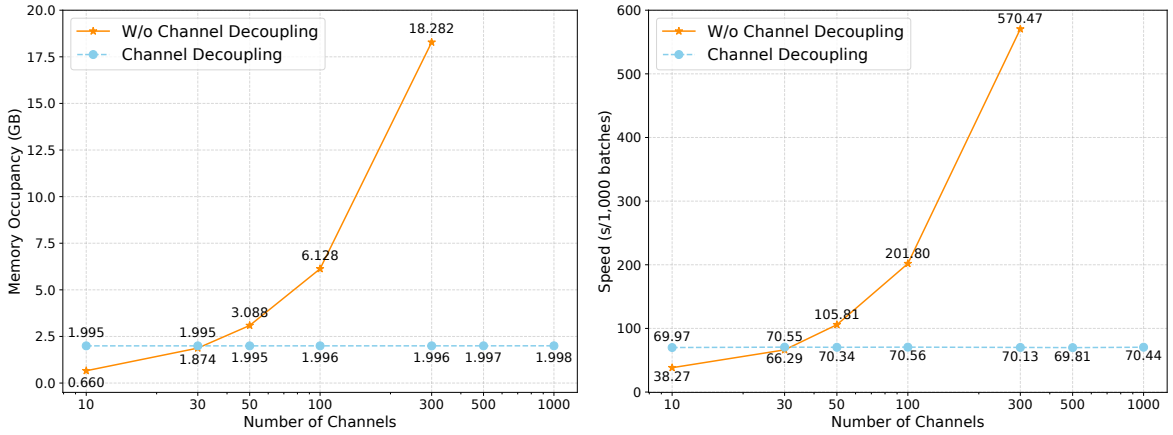
Methods		Task-Specific								General		Pre-Training							
		SAITS		GRIN		LI		SI		TimesNet		TS2Vec		SimMTM		UP2ME(IR)		UP2ME(FT)	
Metric		mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE	mSE	mAE
ETTm1	$0\% \sim 12.5\%$	0.090	0.186	0.286	0.385	0.606	0.454	2.138	1.086	0.094	0.201	0.142	0.249	0.262	0.338	0.213	0.277	**0.060**	**0.158**
	$12.5\% \sim 25\%$	0.131	0.227	0.538	0.534	0.971	0.599	2.072	1.067	0.146	0.250	0.254	0.349	0.323	0.374	0.264	0.311	**0.098**	**0.198**
	$25\% \sim 37.5\%$	0.236	0.314	0.526	0.518	1.028	0.622	2.015	1.051	0.194	0.288	0.311	0.384	0.349	0.395	0.295	0.332	**0.146**	**0.239**
	$37.5\% \sim 50\%$	0.349	0.385	0.620	0.545	1.042	0.631	1.959	1.036	0.256	0.327	0.393	0.432	0.377	0.414	0.328	0.352	**0.209**	**0.288**
	Avg	0.201	0.278	0.492	0.496	0.912	0.577	2.046	1.060	0.172	0.266	0.275	0.353	0.328	0.380	0.275	0.318	**0.128**	**0.221**
Weather	$0\% \sim 12.5\%$	0.067	0.115	0.243	0.322	0.111	0.124	1.043	0.750	0.085	0.141	0.082	0.147	0.127	0.184	0.102	0.135	**0.055**	**0.081**
	$12.5\% \sim 25\%$	0.094	0.151	0.224	0.306	0.178	0.199	1.026	0.744	0.093	0.142	0.110	0.186	0.142	0.197	0.136	0.180	**0.067**	**0.093**
	$25\% \sim 37.5\%$	0.116	0.180	0.157	0.239	0.198	0.222	1.006	0.738	0.121	0.174	0.139	0.221	0.168	0.224	0.165	0.211	**0.086**	**0.116**
	$37.5\% \sim 50\%$	0.134	0.194	0.304	0.359	0.228	0.248	0.989	0.732	0.154	0.206	0.168	0.249	0.248	0.304	0.195	0.238	**0.108**	**0.141**
	Avg	0.103	0.160	0.232	0.306	0.179	0.198	1.016	0.741	0.113	0.166	0.125	0.201	0.171	0.228	0.150	0.191	**0.079**	**0.108**
Electricity	$0\% \sim 12.5\%$	0.206	0.317	0.320	0.413	1.230	0.832	1.896	1.101	0.134	0.252	0.224	0.324	OOM		0.085	0.181	**0.080**	**0.174**
	$12.5\% \sim 25\%$	0.208	0.316	0.263	0.368	1.285	0.864	1.805	1.072	0.136	0.253	0.223	0.321			0.102	0.199	**0.091**	**0.187**
	$25\% \sim 37.5\%$	0.212	0.318	0.179	0.292	1.298	0.872	1.713	1.044	0.141	0.257	0.227	0.322			0.114	0.211	**0.102**	**0.199**
	$37.5\% \sim 50\%$	0.220	0.324	0.493	0.524	1.294	0.872	1.622	1.017	0.149	0.265	0.235	0.328			0.128	0.224	**0.114**	**0.211**
	Avg	0.211	0.319	0.313	0.399	1.277	0.860	1.759	1.058	0.140	0.256	0.227	0.324	OOM		0.107	0.204	**0.097**	**0.193**
Traffic	$0\% \sim 12.5\%$	0.565	0.309	0.482	0.275	2.183	0.974	2.888	1.140	0.492	0.271	0.553	0.300	OOM		0.307	0.209	**0.228**	**0.173**
	$12.5\% \sim 25\%$	0.567	0.307	0.499	0.282	2.276	1.000	2.759	1.109	0.505	0.273	0.551	0.296			0.334	0.221	**0.274**	**0.189**
	$25\% \sim 37.5\%$	0.573	0.310	0.504	0.290	2.287	1.004	2.631	1.076	0.513	0.277	0.558	0.297			0.346	0.227	**0.315**	**0.206**
	$37.5\% \sim 50\%$	0.586	0.317	0.551	0.319	2.269	1.000	2.504	1.044	0.521	0.283	0.572	0.301			0.366	0.237	**0.358**	**0.219**
	Avg	0.573	0.311	0.509	0.291	2.253	0.994	2.696	1.092	0.508	0.276	0.559	0.299	OOM		0.338	0.223	**0.294**	**0.197**
SMD	$0\% \sim 12.5\%$	0.790	0.167	1.038	0.281	1.194	0.138	3.328	0.878	0.845	0.155	1.147	0.331	0.812	0.127	**0.694**	0.120	0.729	**0.083**
	$12.5\% \sim 25\%$	0.850	0.207	1.252	0.426	1.270	0.157	3.226	0.873	0.844	0.164	1.183	0.360	0.828	0.151	0.864	0.150	**0.733**	**0.092**
	$25\% \sim 37.5\%$	0.893	0.221	1.366	0.463	1.372	0.174	3.180	0.868	0.884	0.184	1.241	0.386	0.847	0.163	0.877	0.182	**0.768**	**0.111**
	$37.5\% \sim 50\%$	0.928	0.234	0.938	0.215	1.319	0.199	3.127	0.864	0.934	0.210	1.296	0.412	0.882	0.179	0.921	0.212	**0.796**	**0.128**
	Avg	0.865	0.207	1.148	0.346	1.289	0.167	3.215	0.870	0.877	0.178	1.217	0.372	0.842	0.155	0.839	0.166	**0.756**	**0.103**
PSM	$0\% \sim 12.5\%$	0.543	0.419	0.422	0.308	0.171	0.210	3.083	1.147	0.255	0.298	1.463	0.716	0.152	0.215	0.171	0.220	**0.103**	**0.164**
	$12.5\% \sim 25\%$	0.568	0.432	0.493	0.351	0.210	0.244	3.043	1.136	0.276	0.313	1.481	0.725	0.203	0.255	0.279	0.305	**0.116**	**0.173**
	$25\% \sim 37.5\%$	0.615	0.458	1.656	0.818	0.257	0.273	3.001	1.126	0.312	0.337	1.507	0.738	0.254	0.290	0.405	0.380	**0.155**	**0.208**
	$37.5\% \sim 50\%$	0.677	0.490	0.712	0.469	0.290	0.301	2.975	1.119	0.365	0.371	1.536	0.753	0.293	0.318	0.504	0.433	**0.205**	0.245
	Avg	0.601	0.450	0.821	0.486	0.232	0.257	3.026	1.132	0.302	0.330	1.497	0.733	0.225	0.269	0.340	0.334	**0.144**	**0.197**
SWaT	$0\% \sim 12.5\%$	2.660	0.694	4.981	0.773	0.105	0.031	13.606	1.569	0.191	0.091	6.313	1.386	0.083	0.048	0.086	0.035	**0.065**	**0.031**
	$12.5\% \sim 25\%$	2.750	0.714	6.088	0.873	0.172	0.049	13.592	1.566	0.218	0.095	6.146	1.263	0.139	0.061	0.167	0.056	**0.109**	**0.041**
	$25\% \sim 37.5\%$	2.975	0.764	6.615	0.916	0.217	0.064	13.571	1.563	0.255	0.106	6.112	1.227	0.185	0.073	0.228	0.074	**0.141**	**0.048**
	$37.5\% \sim 50\%$	3.332	0.845	6.876	0.927	0.249	0.074	13.545	1.559	0.298	0.116	5.794	1.152	0.215	0.081	0.287	0.094	**0.171**	**0.059**
	Avg	2.929	0.754	6.140	0.872	0.186	0.055	13.579	1.564	0.240	0.102	6.091	1.257	0.155	0.066	0.192	0.064	**0.121**	**0.045**
GECCO	$0\% \sim 12.5\%$	3.564	1.088	5.954	1.369	**1.132**	**0.173**	11.913	2.077	1.761	0.398	6.101	1.397	1.453	0.275	1.136	0.196	1.438	0.256
	$12.5\% \sim 25\%$	4.107	1.217	5.618	1.329	1.670	**0.235**	11.750	2.071	1.692	0.406	6.057	1.430	1.395	0.294	**1.055**	0.282	1.321	0.266
	$25\% \sim 37.5\%$	3.765	1.119	4.548	1.155	1.767	**0.277**	11.609	2.045	1.825	0.418	6.072	1.393	1.597	0.331	1.735	0.343	**1.524**	0.301
	$37.5\% \sim 50\%$	3.875	1.142	4.246	1.010	1.969	0.313	11.608	2.061	1.835	0.427	6.029	1.385	1.616	0.356	1.693	0.375	**1.590**	0.339
	Avg	3.828	1.142	5.091	1.216	1.634	**0.250**	11.720	2.063	1.778	0.412	6.065	1.401	1.515	0.314	**1.405**	0.299	1.468	0.290

Table 7: Precision (P), Recall (R), F1-score, Average Precision (AP) evaluation of anomaly detection. Bold/underline indicates the best/second. Our method is marked in gray.

Methods		Task-Specific				General	Pre-Training			
		DCdetector	AnomalyTrans	iForest	OCSVM	TimesNet	TS2Vec	SimMTM	UP2ME(IR)	UP2ME(FT)
SMD	P	87.21	**88.21**	38.80	87.87	80.71	65.12	80.61	80.22	82.85
	R	79.49	93.92	**93.94**	54.44	85.63	86.02	85.56	85.32	83.78
	F1	84.40	**90.98**	54.92	67.23	83.09	74.13	83.01	82.69	83.31
	AP	82.75	93.49	80.65	73.42	90.59	87.79	**93.91**	93.90	93.58
PSM	P	97.21	97.32	96.22	99.00	99.16	84.82	**99.27**	98.94	99.02
	R	**97.79**	97.41	86.00	76.85	83.28	90.24	88.14	93.33	95.38
	F1	**97.50**	97.37	90.82	86.53	90.53	87.44	93.37	96.05	97.16
	AP	98.73	98.80	96.61	96.86	99.70	96.48	99.73	99.75	**99.76**
SWaT	P	**93.28**	90.49	23.20	47.63	89.12	15.27	91.18	92.06	91.98
	R	**100.00**	**100.00**	95.86	87.18	92.60	92.42	87.47	93.74	95.81
	F1	**96.52**	95.01	37.36	61.61	90.83	26.20	89.29	92.89	93.85
	AP	**99.57**	98.92	91.46	88.55	97.37	82.12	97.38	97.83	98.07
GECCO	P	22.40	24.92	20.09	**93.33**	52.44	10.73	72.19	41.67	50.57
	R	54.25	55.75	38.36	36.44	41.68	38.36	35.20	**84.93**	**84.93**
	F1	31.71	34.45	26.37	52.41	46.45	16.77	47.32	55.91	**63.39**
	AP	31.87	48.54	38.92	62.08	**68.53**	37.76	63.30	65.47	65.09

C. Additional Experiments about Computational Overhead

Ablation of Channel Decoupling in Pre-training. Figure 5 shows the memory occupancy and time cost of pre-training with and without channel decoupling against the number of channels C. We can see that without channel decoupling, both memory occupancy and time cost increase linearly w.r.t C and the pre-training process encounters the out-of-memory (OOM) problem on one NVIDIA Quadro RTX 8000 GPU with 48GB memory when $C > 300$, even with a small batch size 8. Incontrast, with channel decoupling, the memory occupancy and time cost are irrelative to C, enabling UP2ME to scale effectively to high-dimensional datasets, such as Electricity ($C = 321$) and Traffic ($C = 862$).



(a) (b)

Figure 5: Efficieny evaluation of channel decoupling in pre-training. (a) Pre-training memory occupancy with and without channel decoupling against the number of channels C on synthetic datasets with different numbers of channels. (b) Pre-training time cost with and without channel decoupling against the number of channels. Experiments are conducted on a single NVIDIA Quadro RTX 8000 GPU with 48GB memory. The batch size is set to 256 for channel decoupling and 8 for without channel decoupling. The x-axis is in the log scale.

Hyper-parameter r in Graph Construction of Fine-tuning. Figure 6 illustrates the memory occupancy of fine-tuning against hyper-parameter r in graph construction. The memory occupancy increases rapidly when $r > 10$, reaching the OOM scenario when $r > 20$. This underscores the challenge of employing fully connected graphs for high-dimensional datasets and emphasizes the necessity of our sparse graph construction.

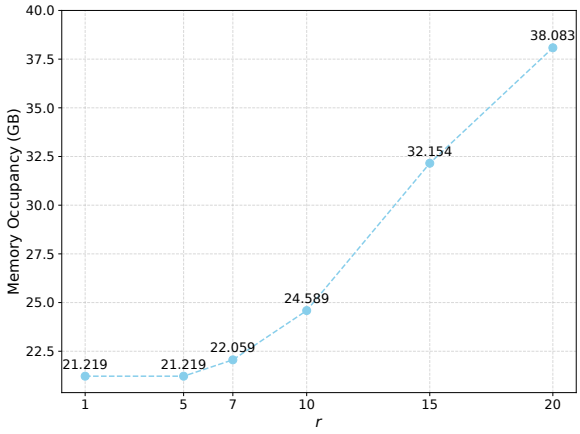


Figure 6: Fine-tuning memory occupancy against hyper-parameter r in graph construction on Electricity ($C = 321$).

D. Visualization

D.1. Mask-Reconstruction

Figure 7 shows some mask-reconstruction cases using UP2ME. The pre-trained UP2ME can leverage complex temporal dependency to reconstruct time series of different lengths from different channels.

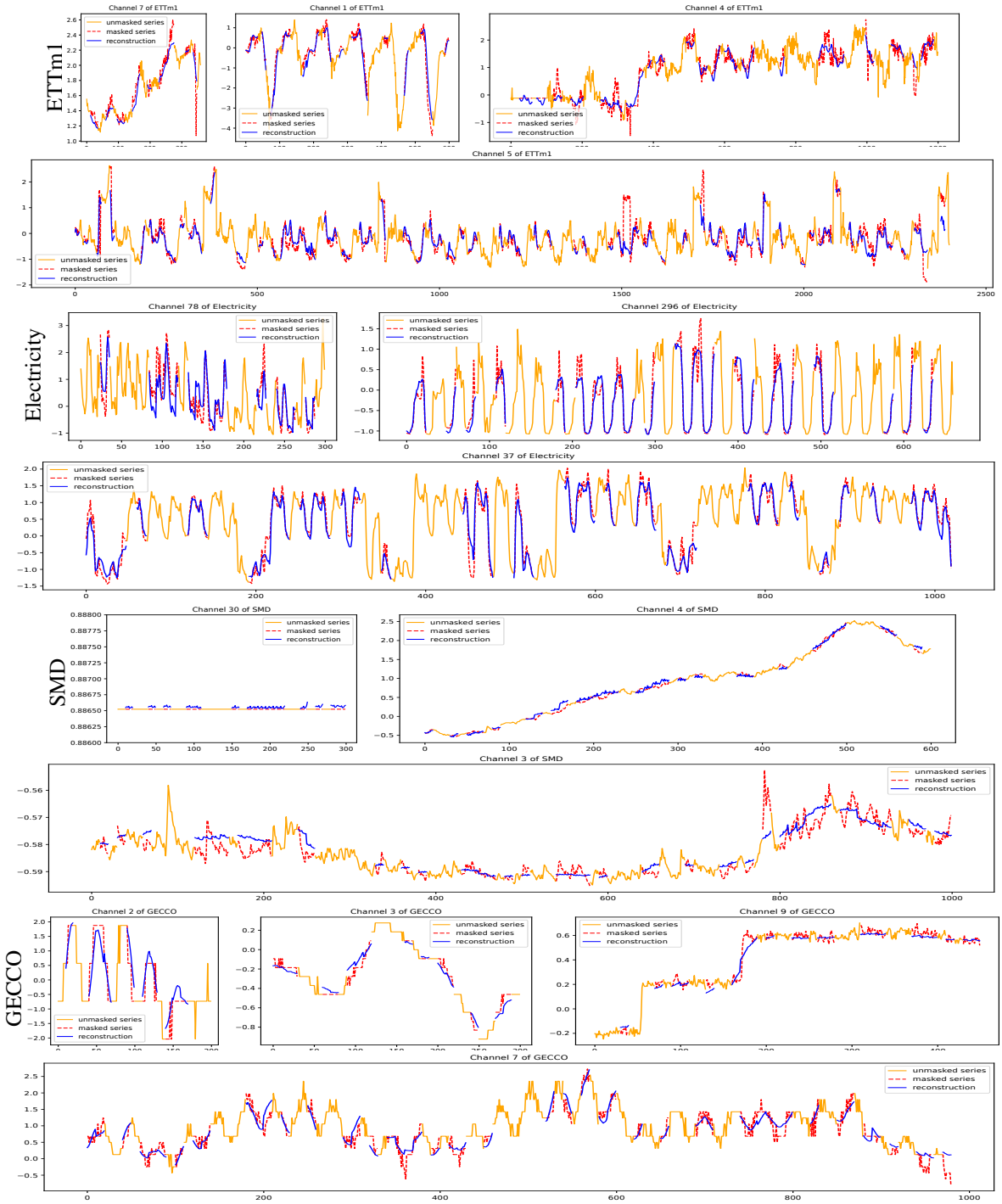
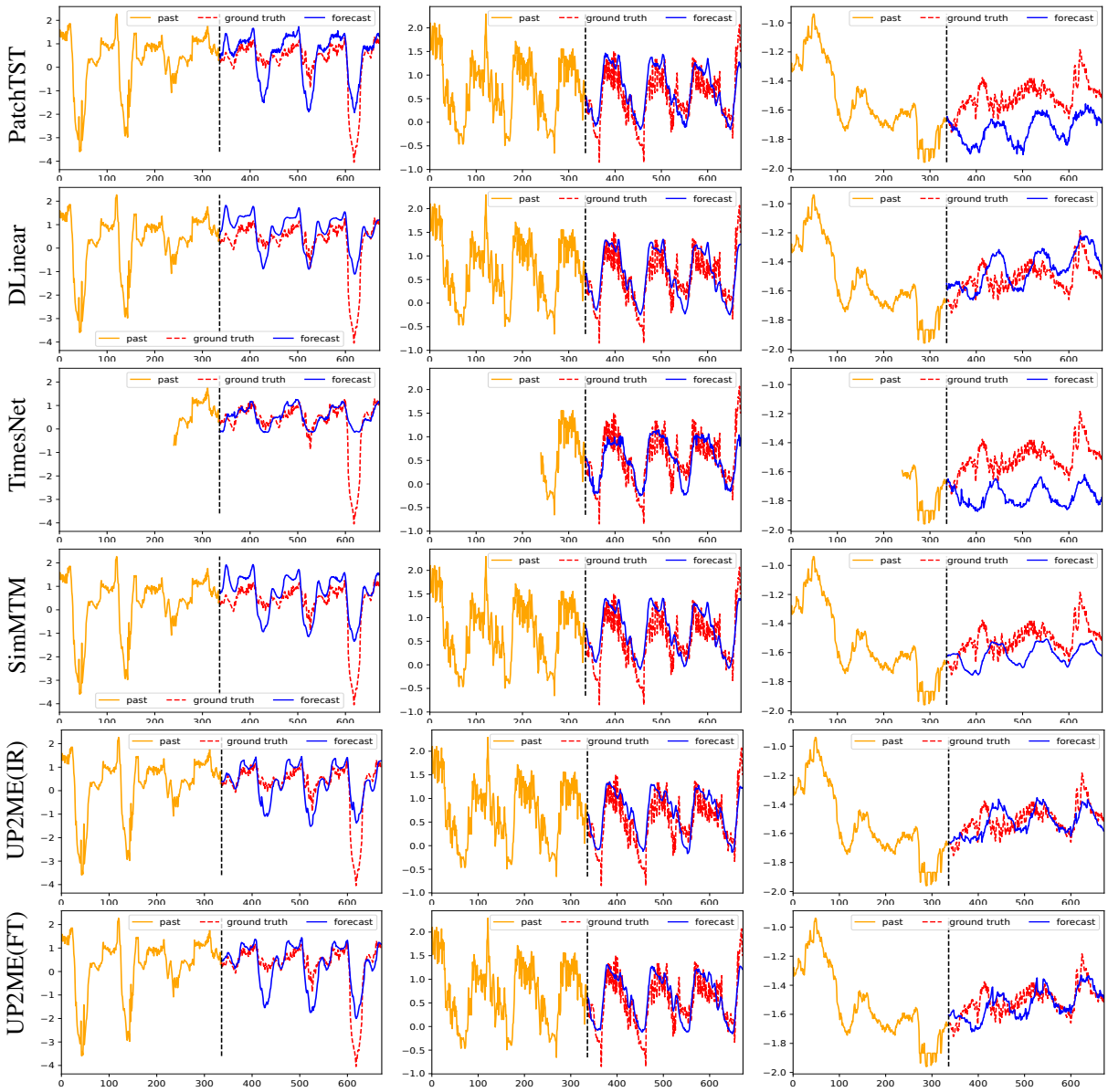


Figure 7: Mask-reconstruction cases using UP2ME. Every two rows represent the same dataset. The orange/red/blue curves stand for the unmasked/masked/reconstructed series. Cases of different channels with different lengths on the same dataset are generated by the same pre-trained model.

D.2. Forecasting

Figure 8 shows the forecasting cases of three channels in the ETTm1 dataset. For channels #1 and #2, all methods successfully predict the periodic pattern. For channel #7, PatchTST, TimesNet and SimMTM fail to predict the additional increasing trend. And predictions of our UP2ME(IR) and UP2ME(FT) are closer to the ground truth compared with DLinear.

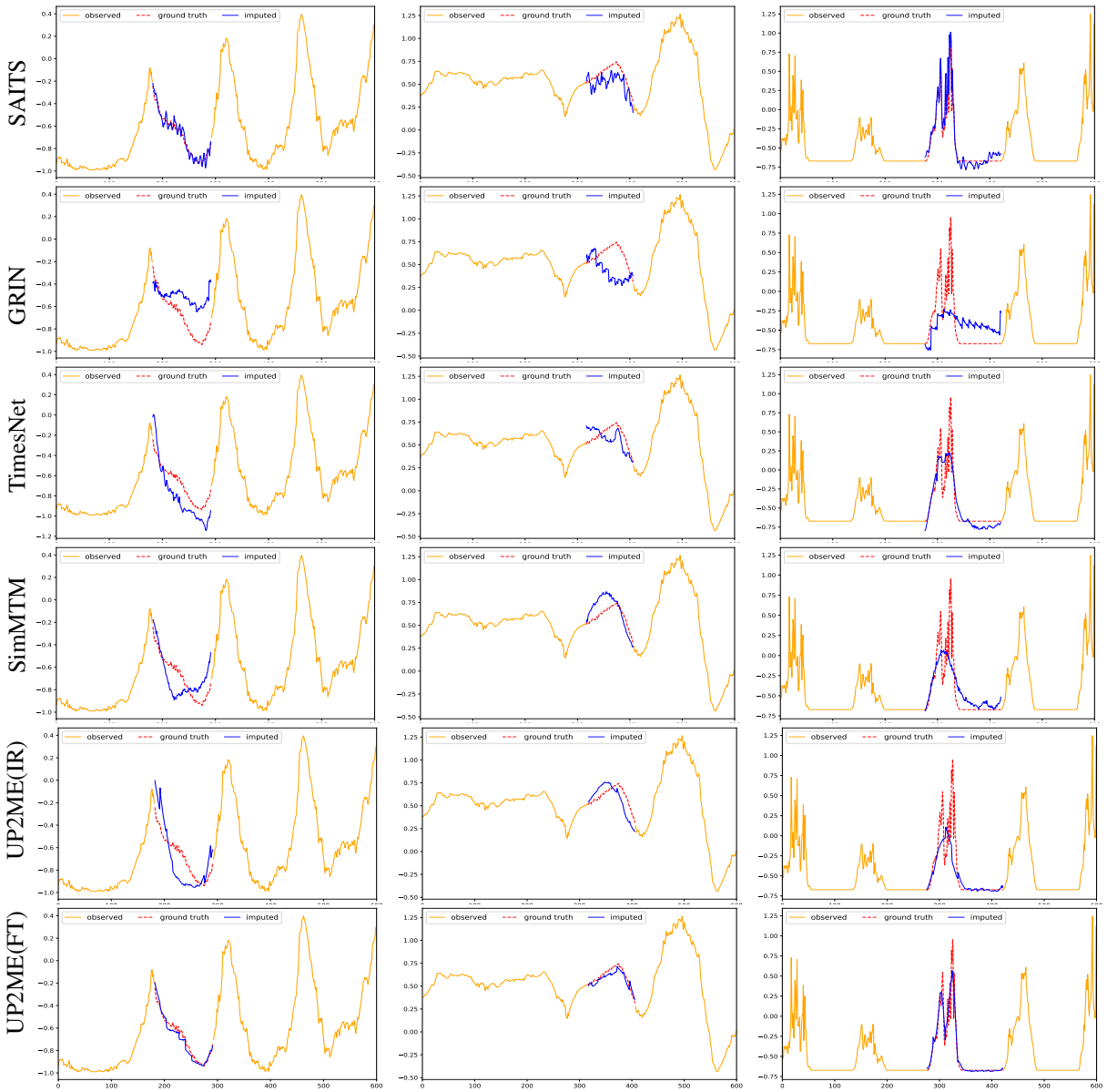


(a) Channel #1 (b) Channel #2 (c) Channel #7

Figure 8: Forecasting cases for 3 channels in the ETTm1 dataset. The prediction length is set to 336; all methods except TimesNet utilize the past 336 points for forecasting, while TimesNet employs the past 96 points, as experiments indicate its preference for a shorter window for improved results. The orange/red/blue curves represent past/future/predicted time series. Each row corresponds to one method, and each column corresponds to one channel.

D.3. Imputation

Figure 9 shows the imputation cases of three channels in the Weather dataset. Our UP2ME(IR) rivals the most competitive task-specific models. Additionally, UP2ME(FT) achieves remarkably faithful imputation compared to the ground truth. Moreover, the outputs of our UP2ME are more continuous with less oscillation.



(a) Channel #7 (b) Channel #10 (c) Channel #16

Figure 9: Imputation cases for 3 channels in the Weather dataset. The observed window length is set to 600, with the missing ratio of $12.5\% \sim 25\%$. The orange/red/blue curves represent observed/ground-truth/imputed series. Each row corresponds to one method, and each column corresponds to one channel.

D.4. Anomaly Detection

Figure 10 illustrates anomaly detection cases on the GECCO dataset. While DCdetector fails to identify the abnormal segment, AnomalyTrans detects two points within the segment. TimesNet, SimMTM, and UP2ME(IR) recognize anomalies at the end of the abnormal segment. After fine-tuning, UP2ME(FT) successfully identifies the entire abnormal segment.

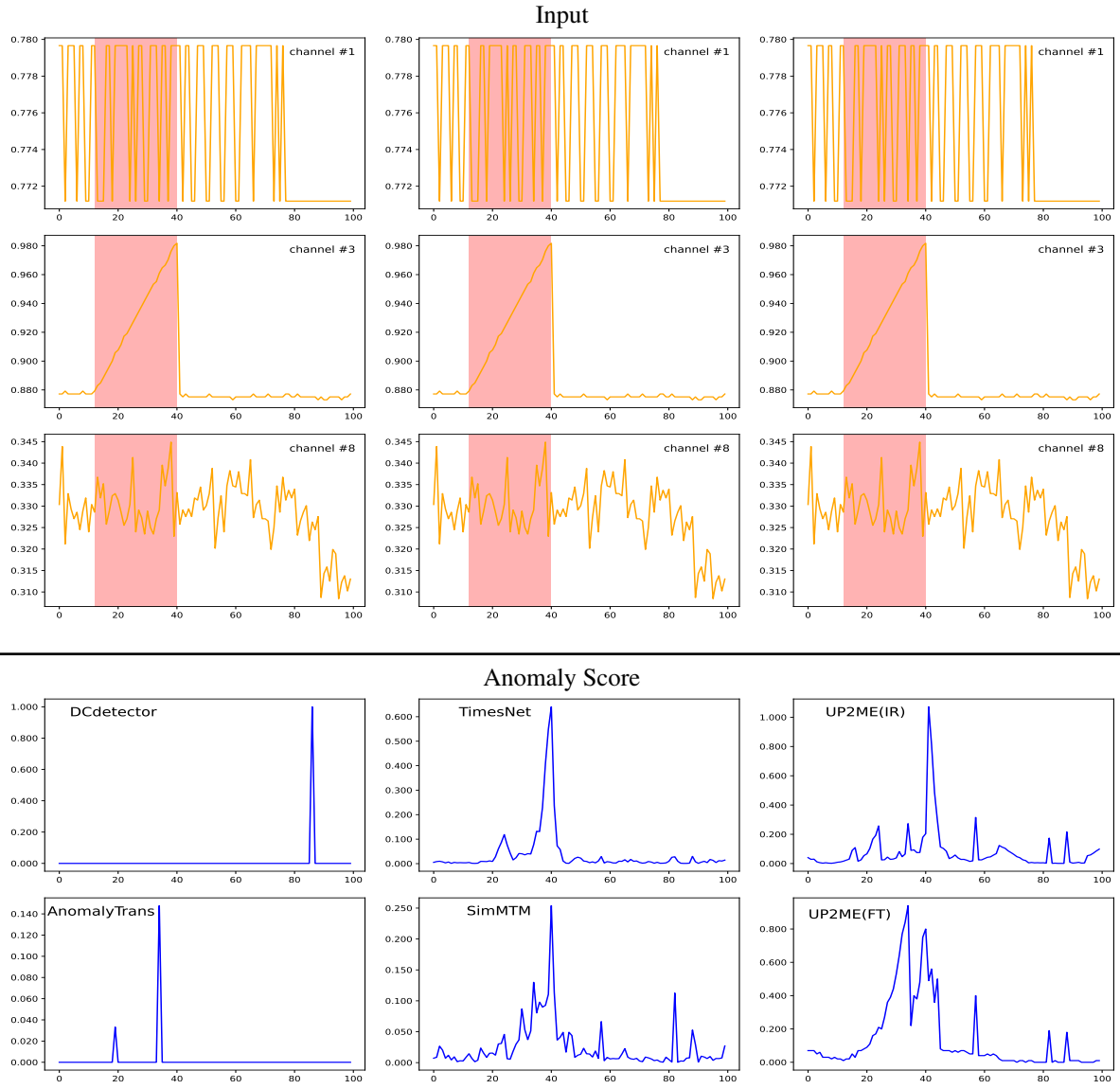


Figure 10: Anomaly detection cases on the GECCO dataset. The first three rows illustrate three channels of the input instance, with each row corresponding to one channel. Input values are repeated three times along the column axis to align with the anomaly scores below for better visualization. Ground truth anomaly timestamps are highlighted in red. The last two rows display anomaly scores generated by different methods.