FEAT: A GENERAL FRAMEWORK FOR FEATURE-AWARE MULTIVARIATE TIME-SERIES REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

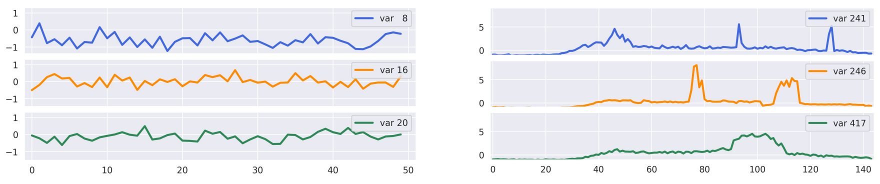
Multivariate time-series is complex and uncertain. The overall temporal patterns change dynamically over time, and each feature is often observed to have a unique pattern. Therefore, it is challenging to model a framework that can flexibly learn feature-specific unique patterns as well as dynamically changing temporal patterns simultaneously. We propose a general framework for FEature-Aware multivariate Time-series representation learning, called FEAT. Unlike previous methods that only focus on training the overall temporal dependencies, we focus on training feature-specific as well as feature-agnostic representations in a data-driven manner. Specifically, we introduce a feature-wise encoder to explicitly model the feature-specific information and design an element-wise gating layer that learns the influence of feature-specific patterns per dataset in general. FEAT outperforms the benchmark models in average accuracy on 29 UEA multivariate time-series classification datasets and in MSE and MAE on four multivariate time-series forecasting datasets.

1 INTRODUCTION

Deep learning methods have been applied to time-series data in various fields such as manufacturing, healthcare, finance, and transportation to enhance the performance of classification or forecasting tasks (Guo et al., 2019; Zhang et al., 2021; Essien & Giannetti, 2020). Approaches related to multivariate time-series data have received considerable attention for leveraging the abundant information of features to improve the performance of downstream tasks (Rasul et al., 2020; Zhang et al., 2020). However, effectively capturing the intrinsic complexity of multivariate time-series data is difficult (Duan et al., 2022; Chen et al., 2022; Liu et al., 2021; Karim et al., 2019), given the complex and uncertain characteristics of data in which features have varied changing patterns over time (Zhao et al., 2015; Du et al., 2020). Therefore, modeling a general representation learning framework to extract the significant information for multivariate time-series data is more challenging than for univariate time-series data (Han et al., 2018; Du et al., 2020).

Most time-series representation learning methods have the following two limitations. First, they mainly focus on training feature-agnostic temporal dependency. In other words, for multivariate time-series data, the unique patterns of each feature are not given sufficient attention. Figure 1a shows an example of a multivariate time-series sequence with similar temporal behaviors across the features, which we define as feature-agnostic patterns referring to Cirstea et al. (2022). By contrast, Figure 1b shows the case where the temporal behavior is different for each individual feature, which we call feature-specific patterns. Although both types of feature characteristics should be properly captured, current time-series representation learning methods tend to overlook the feature-specific patterns.

Second, few time-series representation learning methods are generally applicable to multivariate time-series data. We define *general* as *"to be able to perform robustly on various datasets from diverse domains with various tasks"*. However, most previous methods have focused on improving classification task via representation learning (Tonekaboni et al., 2021; Eldele et al., 2021). Franceschi et al. (2019) and Zerveas et al. (2021) conducted classification as well as regression tasks but only a few selected datasets were used for evaluations. Yue et al. (2022) proposed a universal



(a) Feature-agnostic patterns (b) Feature-specific patterns

Figure 1: (a) shows a multivariate time-series with feature-agnostic patterns where each feature has similar temporal behaviors. In contrast, (b) shows a sequence with feature-specific patterns. The former is a subset of features from *FingerMovements* dataset, while the latter is from *PEMS-SF* dataset in the UEA archive.

representation learning framework, TS2Vec, which achieved state-of-the-art performance on 29 multivariate time-series classification datasets and four multivariate time-series forecasting benchmarks. However, TS2Vec is designed to learn only feature-agnostic temporal representation. Specifically, as the multivariate time-series sequence is embedded along the temporal axis, the original feature dimension cannot be preserved any more, making it difficult to explicitly model feature-specific information (Yang et al., 2015).

To resolve these two limitations, we propose FEAT, a general framework for FEature-Aware multivariate Time-series representation learning, which can be applied to datasets in various domain and tasks. In contrast to the previous methods that focus on training feature-agnostic temporal patterns, we also focus on training feature-aware temporal representation by leveraging a feature-wise encoder. Furthermore, we propose an element-wise gating layer that flexibly learns the influence of feature-specific patterns of a given input sequence per dataset. The main contributions of this paper are summarized as follows:

- We propose FEAT, a general representation learning framework for multivariate time-series data that learns feature-specific patterns along with the feature-agnostic temporal patterns.
- We explicitly model the feature-specific information for the first time by introducing a feature encoder and present an element-wise gating layer that flexibly learns the influence of feature-specific patterns per timestamp in a data-driven manner.
- FEAT outperforms existing state-of-the-art representation learning methods on various benchmark datasets and tasks including classification (improvement of 1.9%p on average in terms of accuracy) and forecasting (improvement of 9.54% on average in terms of MSE).

2 Method

The overall architecture of FEAT is shown in Figure 2. We propose a novel self-supervised learning framework for multivariate time-series that utilizes feature-specific patterns in addition to temporal patterns. Specifically, FEAT learns representation for the first time in terms of three diversified perspectives: feature-specific patterns, feature-agnostic temporal patterns, and dependency between multiple feature-specific and temporal information. In the following subsections, we describe the architectural details of the proposed framework.

Notations $S = \{X_1, X_2, \ldots, X_N\}$ is a set of time-series data with N instances. $X_i \in \mathbb{R}^{T\times F}$, the i^{th} instance in the set S, is the input of our framework. T is the sequence length, and F is the number of features of the given input sequence. d_1 and d_2 are the dimension of timestamp-wise embedding and feature-wise embedding, respectively, and D stands for the final representation dimension. Note that we mark the notations and the output dimensions of each layer in Figure 2 to help understand the following explanations.

2.1 Input Embedding

Timestamp-wise embedding and feature-wise embedding We use two learnable input embedding matrices, $W_1 \in \mathbb{R}^{F\times d_1}$ and $W_2 \in \mathbb{R}^{T\times d_2}$, to learn the timestamp-wise embedding E_1 and the

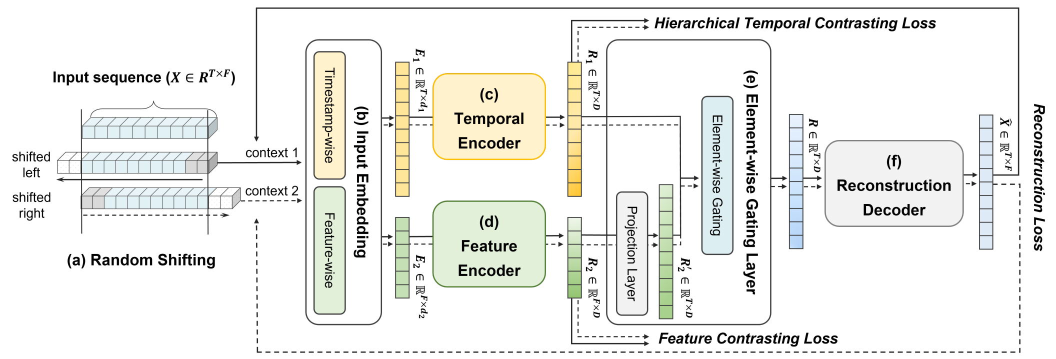

Input sequence ($X \in R^{T \times F}$)
shifted left
shifted right
context 1
context 2
(a) Random Shifting
Timestamp-wise
Feature-wise
(b) Input Embedding
$E_1 \in \mathbb{R}^{T \times d_1}$
$E_2 \in \mathbb{R}^{F \times d_2}$
(c) Temporal Encoder
(d) Feature Encoder
$R_1 \in \mathbb{R}^{T \times D}$
$R_2 \in \mathbb{R}^{F \times D}$
Hierarchical Temporal Contrasting Loss
Feature Contrasting Loss
Projection Layer
$R_2' \in \mathbb{R}^{T \times D}$
Element-wise Gating
(e) Element-wise Gating Layer
$R \in \mathbb{R}^{T \times D}$
(f) Reconstruction Decoder
$\hat{X} \in \mathbb{R}^{T \times F}$
Reconstruction Loss

Figure 2: Overall architecture of FEAT. The input time-series is randomly shifted along the horizontal axis to construct an augmented pair for contrastive learning. The temporal encoder learns feature-agnostic temporal representation through hierarchical temporal contrasting. The feature encoder learns feature-specific representation by feature contrasting, which is then aligned to the temporal axis by a projection layer to construct the feature-specific temporal representation. The final representation is derived by integrating feature-specific as well as feature-agnostic temporal representations by an element-wise gating layer. The decoder reconstructs the input time-series from the final representation for the auxiliary task. The dashed line indicates the stream of the augmented context pair by random shifting.

feature-wise embedding E_2 respectively, as shown in Figure 2b. The timestamp-wise embedding has a limitation in that the embedded vector of size $T \times d_1$ cannot preserve the dimension of F as the identity of each feature is entangled after the projection. This makes it difficult to model the feature-specific patterns in the upper layer. Therefore, we introduce a feature-wise input embedding matrix W_2 to utilize the feature-specific patterns explicitly. Without using any meta-information or manual preprocessing, we embed each feature by projecting the transposed raw input sequence X^T.

2.2 Time-series Data Augmentation

Masking Input Embedding We randomly mask the input embedding as a data augmentation for contrastive learning inspired by TS2Vec (Yue et al., 2022). In our experiment, random binomial masking is applied to the timestamp-wise embedding and no masking is applied to the feature-wise embedding. The detailed experiment is presented in Appendix A.1. Note that the masking is only applied during training.

Random Shifting We also introduce random shifting (Figure 2a), which is applied to the raw input time-series to produce randomly augmented contexts. Yue et al. (2022) proposed random cropping to generate two overlapping segments in contexts with random lengths for contextual consistency. However, for the feature-wise embedding, the size of T in the transposed input with the size of $F \times T$ must be fixed. To mitigate this concern, we propose random shifting, which can maintain the effect of random cropping while keeping the dimension T fixed. As shown in Figure 2a, we shift the raw input time-series along the horizontal axis (where the shift size is randomly chosen as a hyper-parameter) and pad or trim the sequence to preserve the context length. Maximum shift size is used to impose constraints to prevent the distribution of each feature from the shifted context from getting distorted from the original distribution.

2.3 Layers

Encoder Layer We design two encoders, temporal encoder (Figure 2c) and feature encoder (Figure 2d), to extract feature-agnostic temporal representation and feature-specific representation, respectively. We use a dilated convolutional neural network as the temporal encoder to extract the feature-agnostic temporal representation R_1. Dilated convolutional neural networks have been actively used for time-series data because of their ability to capture multi-scale temporal dependency (Tonekaboni et al., 2021; Franceschi et al., 2019; Yue et al., 2022; Bai et al., 2018; Oord et al., 2016). The temporal encoder takes the timestamp-wise embedding as an input, and it focuses on learning dynamic feature-agnostic temporal dependencies rather than the specific patterns of each feature.

To complement the loss of feature-specific information, we design an additional MLP-based feature encoder network, which focuses on learning feature-wise specific representation R_2 by receiving feature-wise embedding.

Element-wise Gating Layer The main goal of time-series representation learning is to represent each timestamp of a given sequence in the hidden dimension space. This means that the length of input sequence and that of the final output representation should be the same. Concerns might arise if the output of the feature encoder (Figure 2d) and the output of temporal encoder (Figure 2c) have different lengths; the former is F while the latter is T in the above structure. To address such concerns, we first align the feature-specific representation R_2 along the temporal axis, learning feature-temporal dependency by a projection layer in Figure 2e. The aligned feature representation R_2' is regarded as feature-specific temporal representation while the representation R_1 from the temporal encoder is regarded as feature-agnostic temporal representation. Once these two types of temporal representations R_1 and R_2' are obtained, we aggregate them using an element-wise gating module in the layer shown in Figure 2e. We assume that each timestamp can be strongly influenced by inter-dependencies with other timestamps that are inherently shared between features, while some timestamps can be influenced by the representative patterns that uniquely appear in each feature (Li et al., 2021; Cirstea et al., 2022). Consequently, we adopt a gating mechanism to allow the model to learn the influence of each characteristic flexibly in a data-driven manner. Through this, the model can more adaptively learn each representation compared to a model with a simple aggregation function such as concatenation or mean. The final representation is $R = \{r_1, r_2, \dots, r_T\}$, where $r_t = a_t * R_{1(t)} + b_t * R'_{2(t)} \in \mathbb{R}^D$. a_t is the weight of t^{th} feature-agnostic temporal representation in R_1, and b_t is the weight of t^{th} feature-specific temporal representation in R_2'. The sum of scalar pair a_t and b_t is equal to 1. The sets of T gating weights, $A = \{a_1, \dots, a_T\}$ and $B = \{b_1, \dots, b_T\}$, are computed as follows:

$$[A; B] = softmax(g([R_1; R_2'])). \tag{1}$$

We use a linear projection layer for $g(\cdot)$. Detailed analysis of the gating module is given in Section 3.3.

All components of the encoder layer enable FEAT to model the representations into three different perspectives sophisticatedly – feature-agnostic temporal representation R_1, feature-specific representation R_2, feature-specific temporal representation R_2'. Unlike previous architectures, we first focus on learning the beneficial feature-level representation as well as how the features are dynamically aligned on the temporal axis. Specifically, through the proposed gating layer, the model can automatically control the usage of feature-specific information at each timestamp.

Decoder Layer We used a reconstruction decoder (Figure 2f) to help train the gating parameters by upstream gradients and learn a richer representation. An MLP decoder is employed, which is only for the auxiliary loss during training.

2.4 Objective Function

We design the objective function as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{contrast_t} + \mathcal{L}_{contrast_f} + \mathcal{L}_{recon}, \tag{2}$$

where $\mathcal{L}_{contrast_t}$ is the hierarchical temporal contrasting loss, $\mathcal{L}_{contrast_f}$ is the feature contrasting loss, and $\mathcal{L}_{recon}$ is the reconstruction loss. The hierarchical temporal contrasting loss $\mathcal{L}_{contrast_t}$, adopted from Yue et al. (2022), is calculated from the overlapping segments of randomly shifted pairs described in Section 2.2. The main strategy is to set the same timestamps from the two overlapping representations as positive pairs and set the different timestamps to be negative pairs. $\mathcal{L}_{contrast_t}$ can be formulated as follows:

$$\ell_{temp}^{(i,t)} = -\log \frac{\exp\left(r_{i,t} \cdot r'_{i,t}\right)}{\sum_{t' \in \Omega} \left(\exp\left(r_{i,t} \cdot r'_{i,t'}\right) + \mathbb{1}_{[t \neq t']} \exp\left(r_{i,t} \cdot r_{i,t'}\right)\right)}, \tag{3}$$

$$\ell_{inst}^{(i,t)} = -\log \frac{\exp\left(r_{i,t} \cdot r'_{i,t}\right)}{\sum_{j=1}^{B} \left(\exp\left(r_{i,t} \cdot r'_{j,t}\right) + \mathbb{1}_{[i \neq j]} \exp\left(r_{i,t} \cdot r_{j,t}\right)\right)}, \tag{4}$$

$$\mathcal{L}_{contrast_t} = \frac{1}{NT} \sum_i \sum_t \left(\ell_{temp}^{(i,t)} + \ell_{inst}^{(i,t)} \right). \tag{5}$$

$\ell_{temp}^{(i,t)}$ in Eq. (3) is the temporal contrasting loss, where $r_{i,t}$ and $r'_{i,t}$ are the i^{th} augmented pair of feature-agnostic temporal representation at timestamp t, Ω is the set of overlapped timestamps from each context pair, and $\mathbb{1}$ is the indicator function. It induces the network to represent the same timestamps from different contexts as similar and every different timestamp to be unique. Specifically, the model can focus more on learning position-agnostic temporal representation under the contextual consistency (Yue et al., 2022). In addition to temporal contrasting loss, instance-wise contrastive loss $\ell_{inst}^{(i,t)}$ of Eq. (4) is applied, where the positive pair is the same timestamp from the augmented instance while the negative pair is the same timestamp from different instances in batch B. The two losses from Eq. (3) and Eq. (4) are computed for every timestamp and instance in a batch as formulated in Eq. (5), and they are also computed on the max-pooled representations along the temporal axis by hierarchical procedures. This enables the model to learn the representation that reflects the multi-scale granularities through the hierarchical structure.

In addition to temporal contrasting, we propose the feature contrasting loss $\mathcal{L}_{contrast_f}$ to force the model to learn the unique representation of each feature. Assuming that the distribution of each feature sequence from the shifted pairs is similar to each other as described in Section 2.2, we take two augmented feature-specific representations of feature f, the output of the feature encoder in Figure 2d, as a positive pair. $\mathcal{L}_{contrast_f}$ can be formulated as follows:

$$\ell_{feat}^{(i,f)} = -\log \frac{\exp\left(r_{i,f} \cdot r'_{i,f}\right)}{\sum_{f\prime=1}^{F} \left(\exp\left(r_{i,f} \cdot r'_{i,f'}\right) + \mathbb{1}_{[f \neq f\prime]} \exp\left(r_{i,f} \cdot r_{i,f'}\right) \right)}, \tag{6}$$

$$\ell_{inst}^{(i,f)} = -\log \frac{\exp\left(r_{i,f} \cdot r'_{i,f}\right)}{\sum_{j=1}^{B} \left(\exp\left(r_{i,f} \cdot r'_{j,f}\right) + \mathbb{1}_{[i \neq j]} \exp\left(r_{i,f} \cdot r_{j,f}\right) \right)}, \tag{7}$$

$$\mathcal{L}_{contrast_f} = \frac{1}{NF} \sum_i \sum_f \left(\ell_{feat}^{(i,f)} + \ell_{inst}^{(i,f)} \right). \tag{8}$$

The main difference of $\mathcal{L}_{contrast_f}$ from $\mathcal{L}_{contrast_t}$ is that it is applied to the feature-specific representation $R_2 \in \mathbb{R}^{F \times D}$ and not applied hierarchically, where $r_{i,f}$ and $r'_{i,f}$ are the representations of feature f from the i^{th} augmented pair. The feature contrasting loss helps the model to represent each feature with its own patterns to a vector by referring to different feature patterns in the same instance as well as different feature patterns from other instances.

Additionally, we employed the reconstruction error $\mathcal{L}_{recon}$ for the loss function of as an auxiliary task. An MSE score between the input sequence X and the reconstructed sequence $\hat{X}$ is calculated on the overlapped timestamps from the shifted pair.

3 EXPERIMENTS

We verified the effectiveness of our representation learning framework by evaluating the performance improvements in classification as well as forecasting tasks. To conduct the downstream tasks, we trained traditional machine learning models on the learned representations following Yue et al. (2022) and Franceschi et al. (2019). For a fair comparison, we fixed the representation dimension as 320 for all the baseline models for representation learning. The results of FEAT were calculated by averaging the results of five repeated runs, and we report the performances of the baseline models from their original papers and from TS2Vec. All the experiments were conducted on a single NVIDIA Titan RTX GPU. Details of the experimental settings and datasets are described in Appendix A.1 and Appendix A.2.

Table 1: Multivariate time-series classification results in terms of average accuracy and average rank with 95% confidence interval. Note that as there is no official result of *InsectWingbeat* from DTW in the UEA archive, we excluded it when computing the average accuracy and rank. The best and second-best results are highlighted in **bold** and underlined, respectively.

Dataset	**FEAT (Ours)**	**TS2Vec**	**T-Loss**	**TNC**	**TS-TCC**	**TST**	**DTW**
ArticularyWordRecognition	**0.991**	0.987	0.943	0.973	0.953	0.977	0.987
AtrialFibrillation	**0.293**	0.200	0.133	0.133	0.267	0.067	0.200
BasicMotions	**1.000**	0.975	**1.000**	0.975	**1.000**	0.975	0.975
CharacterTrajectories	0.993	**0.995**	0.993	0.967	0.985	0.975	0.989
Cricket	0.969	0.972	0.972	0.958	0.917	**1.000**	**1.000**
DuckDuckGeese	0.564	**0.680**	0.650	0.460	0.380	0.620	0.600
EigenWorms	0.811	**0.847**	0.840	0.840	0.779	0.748	0.618
Epilepsy	0.948	0.964	**0.971**	0.957	0.957	0.949	0.964
ERing	0.896	0.874	0.133	0.852	**0.904**	0.874	0.133
EthanolConcentration	0.322	0.308	0.205	0.297	0.285	0.262	**0.323**
FaceDetection	0.530	0.501	0.513	0.536	**0.544**	0.534	0.529
FingerMovements	0.488	0.480	**0.580**	0.470	0.460	0.560	0.530
HandMovementDirection	**0.378**	0.338	0.351	0.324	0.243	0.243	0.231
Handwriting	**0.542**	0.515	0.451	0.249	0.498	0.225	0.286
Heartbeat	0.746	0.683	0.741	0.746	**0.751**	0.746	0.717
JapaneseVowels	0.983	0.984	**0.989**	0.978	0.930	0.978	0.949
Libras	**0.889**	0.867	0.883	0.817	0.822	0.656	0.870
LSST	0.548	0.537	0.509	**0.595**	0.474	0.408	0.551
MotorImagery	0.562	0.510	0.580	0.500	**0.610**	0.500	0.500
NATOPS	0.921	**0.928**	0.917	0.911	0.822	0.850	0.883
PEMS-SF	**0.874**	0.682	0.676	0.699	0.734	0.740	0.711
PenDigits	**0.989**	**0.989**	0.981	0.979	0.974	0.560	0.977
PhonemeSpectra	0.216	0.233	0.222	0.207	**0.252**	0.085	0.151
RacketSports	**0.888**	0.855	0.855	0.776	0.816	0.809	0.803
SelfRegulationSCP1	**0.852**	0.812	0.843	0.799	0.823	0.754	0.775
SelfRegulationSCP2	0.562	**0.578**	0.539	0.550	0.533	0.550	0.539
SpokenArabicDigits	0.986	**0.988**	0.905	0.934	0.970	0.923	0.963
StandWalkJump	**0.533**	0.467	0.333	0.400	0.333	0.267	0.200
UWaveGestureLibrary	**0.929**	0.906	0.875	0.759	0.753	0.575	0.903
InsectWingbeat	0.462	0.466	0.156	**0.469**	0.264	0.105	-
On the first 29 datasets:							
Avg. Acc.	**0.731(+1.9%)**	0.712	0.675	0.677	0.682	0.635	0.650
Avg. Rank	**2.448**	3.034	3.759	4.690	4.328	5.172	4.569
95% Confidence Interval	[1.857, 3.040]	[2.360, 3.709]	[2.997, 4.520]	[4.144, 5.236]	[3.506, 5.149]	[4.499, 5.845]	[3.913, 5.225]

3.1 Multivariate Time-Series Classification

For the multivariate time-series classification task, experiments were conducted on 30 UEA archive[1] datasets (Bagnall et al., 2018). We compared the performance of FEAT against six well-known time-series representation learning methodologies: TS2Vec (Yue et al., 2022), T-Loss (Franceschi et al., 2019), TS-TCC (Eldele et al., 2021), TST (Zerveas et al., 2021), TNC (Tonekaboni et al., 2021), and a distance-based algorithm DTW (Chen et al., 2013). Following the same evaluation protocol as TS2Vec, we trained an SVM classifier with RBF kernel on a max-pooled representation along the temporal axis.

Table 1 presents the summarized results of the classification task. We can observe that FEAT resulted in the best average accuracy with 1.9%p improvement on 29 UEA datasets over the second-best model TS2Vec. We can also observe that the average accuracy is not a biased result on account of a part of datasets, in that the average rank also outperforms the other baselines. The results in Table 1 imply that feature-aware representation learning benefits the performance compared to methods that only learn feature-agnostic temporal dependencies. We also analyzed the efficiency of FEAT in terms of total training time in Appendix A.3.

3.2 Multivariate Time-Series Forecasting

For the multivariate time-series forecasting task, experiments were conducted on the four benchmark datasets: ETTh$_1$, ETTh$_2$, ETTm$_1$, and ECL. We compared the performance of FEAT against the representation learning model TS2Vec (Yue et al., 2022) and five end-to-end forecasting models (Zhou et al., 2021; Cao et al., 2020; Bai et al., 2018; Li et al., 2019; Lai et al., 2018). For a fair comparison, we followed the same protocol as TS2Vec and trained a ridge regression on the final

[1]https://www.timeseriesclassification.com/index.php

Table 2: Multivariate time-series forecasting results. The **bold** and underlined numbers indicate the best and second-best performances, respectively.

		FEAT (Ours)		TS2Vec		Informer		StemGNN		TCN		LogTrans		LSTnet	
Dataset	**H**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh$_1$	24	**0.569**	**0.517**	0.599	0.534	0.577	0.549	0.614	0.571	0.767	0.612	0.686	0.604	1.293	0.901
	48	**0.612**	**0.547**	0.629	0.555	0.685	0.625	0.748	0.618	0.713	0.617	0.766	0.757	1.456	0.96
	168	0.751	0.636	0.755	0.636	0.931	0.752	**0.663**	**0.608**	0.995	0.738	1.002	0.846	1.997	1.214
	336	**0.887**	0.718	0.907	**0.717**	1.128	0.873	0.927	0.73	1.175	0.8	1.362	0.952	2.655	1.369
	720	**1.031**	0.796	1.048	**0.79**	1.215	0.896	-	-	1.453	1.311	1.397	1.291	2.143	1.38
ETTh$_2$	24	0.403	0.481	**0.398**	**0.461**	0.72	0.665	1.292	0.883	1.365	0.888	0.828	0.75	2.742	1.457
	48	0.608	0.602	**0.58**	**0.573**	1.457	1.001	1.099	0.847	1.395	0.96	1.806	1.034	3.567	1.687
	168	**1.686**	**1.011**	1.901	1.065	3.489	1.515	2.282	1.228	3.166	1.407	4.07	1.681	3.242	2.513
	336	**1.973**	**1.145**	2.304	1.215	2.723	1.34	3.086	1.351	3.256	1.481	3.875	1.763	2.544	2.591
	720	**2.21**	**1.238**	2.65	1.373	3.467	1.473	-	-	3.69	1.588	3.913	1.552	4.625	3.709
ETTm$_1$	24	0.37	0.395	0.443	0.436	**0.323**	**0.369**	0.62	0.57	0.324	0.374	0.419	0.412	1.968	1.17
	48	0.497	0.467	0.582	0.515	0.494	0.503	0.744	0.628	**0.477**	**0.45**	0.507	0.583	1.999	1.215
	96	**0.548**	**0.503**	0.622	0.549	0.678	0.614	0.709	0.624	0.636	0.602	0.768	0.792	2.762	1.542
	288	**0.636**	**0.563**	0.709	0.609	1.056	0.786	0.843	0.683	1.27	1.351	1.462	1.32	1.257	2.076
	672	**0.741**	**0.629**	0.786	0.655	1.192	0.926	-	-	1.381	1.467	1.669	1.461	1.917	2.941
ECL	24	**0.247**	**0.345**	0.287	0.374	0.312	0.387	0.439	0.388	0.305	0.384	0.297	0.374	0.356	0.419
	48	**0.266**	**0.361**	0.307	0.388	0.392	0.431	0.413	0.455	0.317	0.392	0.316	0.389	0.429	0.456
	168	**0.292**	**0.38**	0.332	0.407	0.515	0.509	0.506	0.518	0.358	0.423	0.426	0.466	0.372	0.425
	336	**0.309**	**0.394**	0.349	0.42	0.759	0.625	0.647	0.596	0.349	0.416	0.365	0.417	0.352	0.409
	720	**0.336**	0.413	0.375	0.438	0.969	0.788	-	-	0.447	0.486	0.344	**0.403**	0.38	0.443
Avg.		**0.749**	**0.607**	0.828	0.636	1.154	0.781	-	-	1.192	0.837	1.314	0.892	1.903	1.444

Table 3: Summary of ablation study results of multivariate time-series classification on 30 UEA datasets. We recorded the average accuracy of five repetitions with the standard deviation of total average accuracy per seed and the average standard deviation of accuracy per dataset. The second column denotes the amount of performance degradation compare to the best performance.

		Avg. Acc.		**Std. (per seed)**	**Avg. Std. (per dataset)**
	FEAT	**0.722**		0.004	0.014
Loss	w/o temporal contrasting	0.674	(-4.9%)	0.002	0.019
	w/o feature contrasting	0.711	(-1.1%)	0.002	0.019
	w/o temporal & feature contrasting	0.668	(-5.5%)	0.006	0.025
	w/o reconstruction	0.684	(-3.8%)	0.006	0.025
Module	MLP feature encoder				
	$\rightarrow$ Depth-wise dilated CNN	0.683	(-3.9%)	0.002	0.020
	Gating				
	$\rightarrow$ concatenation + projection	0.686	(-3.6%)	0.005	0.018
	$\rightarrow$ mean	0.698	(-2.4%)	0.007	0.018
	w/o random shifting	0.713	(-1.0%)	0.006	0.017
	w/o random masking	0.708	(-1.4%)	0.005	0.016

representation with 320 dimensions. Specifically, the regression model is trained to predict H future timestamps from the representation of the last timestamp of the given input sequence.

The evaluation results of multivariate time-series forecasting in terms of MSE and MAE are summarized in Table 2. FEAT generally outperforms both types of baselines models (time-series representation learning-based models and end-to-end forecasting models). FEAT is placed best or second-best in most dataset-time horizon cases. The average improvements of FEAT compared to TS2Vec are 9.54% in terms of MSE and 4.56% in terms of MAE. We provide additional analysis of training time and present forecasting plots in Appendix A.4.

3.3 Analysis

3.3.1 Ablation study

The following ablation studies were conducted to verify the effectiveness of each component in FEAT. First, we investigated the effect of each term in our objective function in Eq. (2). As summarized in Table 3, the model without the hierarchical temporal contrasting loss degraded the performance in 4.9%p. Furthermore, the performance without feature contrasting loss shows 1.1%p degradation. Without using either hierarchical temporal contrasting or feature contrasting loss, there

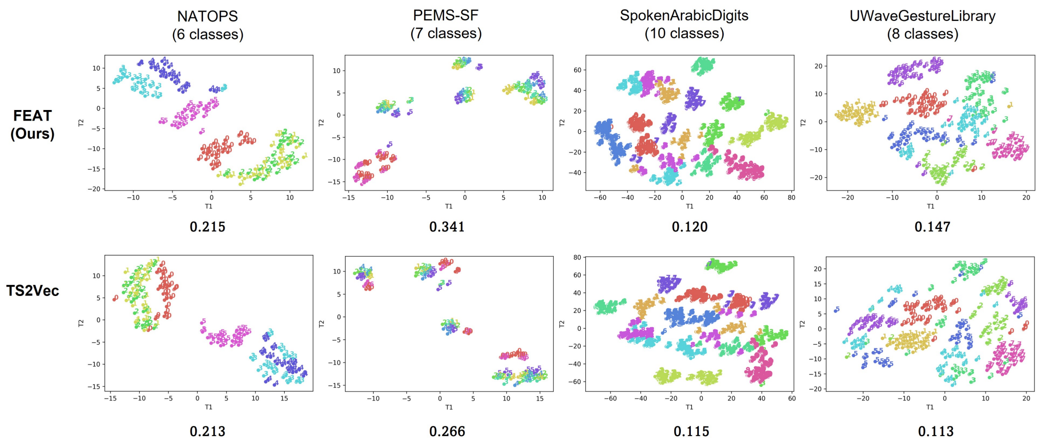
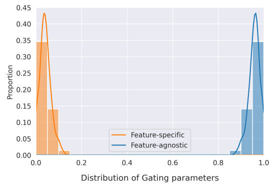
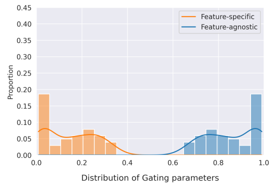


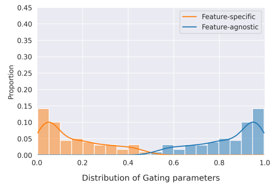
Figure 3: T-SNE visualization of representations from FEAT (first row) and TS2Vec (second row). Four datasets from UEA archive were plotted from the left column. The numbers located at the bottom of the plots are the Silhouette scores.



(a) PhonemeSpectra



(b) NATOPS



(c) UWaveGestureLibrary

Figure 4: Histograms for the distribution of element-wise gating parameters from three UEA datasets. The blue bins indicate the proportion of weights for the feature-agnostic temporal representation, and the orange bins are for the feature-specific temporal representation.

was 5.5%p performance degradation with larger standard deviations. It can be concluded that the hierarchical temporal contrasting loss and the feature contrasting loss both significantly contribute to the performance improvement. We also investigated the effect of the reconstruction loss by removing the reconstruction decoder. Consequently, substantial performance degradation occurred with larger standard deviations per seed and dataset. This implies that the decoder helps the model to learn more stable and effective representations by upstream gradient flows, as discussed in Section 2.3.

Second, we investigated the module-level performance changes. For demonstrating the effectiveness of the MLP-based feature encoder in FEAT (Figure 2d), we evaluated the performance by replacing MLP network with depth-wise dilated convolution, which can also explicitly model the feature-wise patterns. Consequently, 3.9%p performance degradation occurred. Furthermore, we validated the effectiveness of learnable element-wise gating layer in Figure 2e. When we replaced the gating layer with a simple aggregation module such as concatenation or mean, there were 3.6%p or 2.4%p respective performance degradations. In addition, FEAT without random masking and random shifting showed lower performances as well. Among the modules, the MLP-based feature encoder seems to be the most significant module in our feature-aware representation learning framework, followed by the element-wise gating module and the two augmentation techniques. Further analysis of how the feature encoder works is presented in Appendix A.5.1.

3.3.2 Qualitative study

We performed t-SNE visualization to demonstrate that the proposed framework FEAT generally learns the rich representation. Specifically, we compared FEAT with TS2Vec, which only focuses on learning feature-agnostic temporal dependency. The t-SNE plots in Figure 3 demonstrate that the instances belonging to the same class are clustered more densely as well as more clearly separated from the other classes compared to TS2Vec. We also calculated the Silhouette scores in Figure 3 by applying K-means clustering on learned representations with the number of clusters equal to the number of classes. Along with the t-SNE plots, the Silhouette scores support that the learned representations from FEAT were indeed clustered better than TS2Vec as the scores based on our representations were higher than TS2Vec. The overall t-SNE plots and Silhouette scores of all 30 UEA datasets are provided in Appendix A.5.2.

Furthermore, we analyzed the element-wise gating layer in Figure 2e to demonstrate that it can flexibly learn the influence of feature-specific patterns on the various datasets in general. In Figure 4a, the weights for feature-agnostic representation are distributed above 0.8. This means that the temporal dependency highly influences the final representation. Indeed, *PhonemeSpectra* has similar patterns between features along the temporal axis. By contrast, in Figure 4b and Figure 4c, plotted from datasets having diverse patterns among the features, the distributions appear to be more spread out than in Figure 4a. Although the feature-agnostic temporal representation has a more significant influence than the feature-specific temporal representation because of the fundamental characteristic of time-series data, Figure 4 shows that the gating layer can automatically capture the importance of feature-specific patterns in a data-driven manner.

4 Related Work

As large amounts of unlabeled data have been accumulated, extensive self-supervised learning methods have been studied to extract rich representation leveraging these data (Chen et al., 2020; Gao et al., 2021; Devlin et al., 2018). In particular, in natural language processing and computer vision, many representation learning methods have been studied by introducing various pretext tasks or techniques like contrasting and augmentation (Chen et al., 2020; Gidaris et al., 2018; He et al., 2022; Oord et al., 2018). In contrast, studies on time-series representation learning are limited.

A few methodologies of representation learning on time-series data have been proposed to effectively extract rich representations from complex temporal patterns (Yue et al., 2022; Tonekaboni et al., 2021; Franceschi et al., 2019; Eldele et al., 2021; Zerveas et al., 2021; Woo et al., 2022). Yue et al. (2022) proposes a universal framework that learns contextual representations by hierarchical contrastive learning over two overlapping segments from random contexts. Tonekaboni et al. (2021) proposes a framework for learning generalizable time-series representations for non-stationary time-series via defining neighborhoods considering stationarity properties to design a debiased contrastive objective. Franceschi et al. (2019) employs triplet loss to learn scalable representations for multivariate time-series. Eldele et al. (2021) introduces a contrastive learning framework using a tough cross-view prediction task on the augmented sequences. Zerveas et al. (2021) employs a transformer-based architecture for multivariate time-series representation learning using a reconstruction loss. Woo et al. (2022) proposes contrastive learning methods to learn disentangled seasonal-trend representations for forecasting. Most of the aforementioned works propose methods based on temporal dependency, and none have focused on learning multivariate time-series representations through utilizing feature-specific information. We propose a general framework that learns representations with feature-specific information. Notably, our framework can capture the feature-specific information in a data-driven manner for the various datasets, which does not require any additional data preprocessing or expert knowledges.

5 Conclusion

This paper presents a general framework for feature-aware multivariate time-series representation learning, called FEAT. This framework is designed to model the feature-wise specific patterns and to leverage this information for learning rich representation. The evaluation of our framework was conducted on classification as well as forecasting tasks, and the results demonstrate its effectiveness. FEAT achieved the best performances on 30 classification datasets and four forecasting datasets on average among the existing time-series representation learning models. Furthermore, we show that the learnable gating layer in FEAT is flexible enough to learn the feature-specific patterns depending on the datasets. As our framework explicitly models each feature and aligns it to the timestamps, future work can focus on utilizing external features that can enrich the output representations. FEAT can also take advantage of its reconstruction structure in anomaly detection tasks.

REFERENCES

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

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

Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33:17766–17778, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo EAPA Batista. Dtw-d: time series semi-supervised learning from a single example. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 383–391, 2013.

Yijiang Chen, Xiangdong Zhou, Zhen Xing, Zhidan Liu, and Minyang Xu. Cass: A channel-aware self-supervised representation learning framework for multivariate time series classification. In *International Conference on Database Systems for Advanced Applications*, pp. 375–390. Springer, 2022.

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting–full version. *arXiv preprint arXiv:2204.13767*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Shengdong Du, Tianrui Li, Yan Yang, and Shi-Jinn Horng. Multivariate time series forecasting via attention-based encoder–decoder framework. *Neurocomputing*, 388:269–279, 2020.

Ziheng Duan, Haoyan Xu, Yueyang Wang, Yida Huang, Anni Ren, Zhongbin Xu, Yizhou Sun, and Wei Wang. Multivariate time-series classification with hierarchical variational graph pooling. *Neural Networks*, 154:481–490, 2022.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112*, 2021.

Aniekan Essien and Cinzia Giannetti. A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders. *IEEE Transactions on Industrial Informatics*, 16(9): 6069–6078, 2020.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 922–929, 2019.

Min Han, Shoubo Feng, CL Philip Chen, Meiling Xu, and Tie Qiu. Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction. *IEEE Transactions on Knowledge and Data Engineering*, 31(9):1809–1821, 2018.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Zhihan Li, Youjian Zhao, Jiaqi Han, Ya Su, Rui Jiao, Xidao Wen, and Dan Pei. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3220–3230, 2021.

Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*, 2021.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*, 2020.

Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*, 2021.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*, 2022.

Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8980–8987, 2022.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124, 2021.

Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. Graph-guided network for irregularly sampled multivariate time series. *arXiv preprint arXiv:2110.05357*, 2021.

Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6845–6852, 2020.

Jianwei Zhao, Zhihui Wang, Feilong Cao, and Dianhui Wang. A local learning algorithm for random weights networks. *Knowledge-Based Systems*, 74:159–166, 2015.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021.

A Appendix

A.1 Reproduction Details

Table 4: Ablation over different feature masking ratios of multivariate time-series classification on 30 UEA datasets. We recorded the average accuracy of five repetitions with the standard deviation of total average accuracy per seed and the average standard deviation of accuracy per dataset. Random ratio means that the masking ratio is chosen at random under the maximum ratio given in the second column. Fixed ratio means that the given proportion of feature subset is masked.

		Avg. Acc.	**Std. (per seed)**	**Avg. Std. (per dataset)**
	FEAT	<u>0.722</u>	0.004	0.014
random ratio	10%	0.718	0.004	0.016
	30%	0.717	0.003	0.017
	50%	0.717	0.004	0.015
fixed ratio	10%	**0.723**	0.003	0.021
	30%	<u>0.722</u>	0.006	0.019
	50%	0.719	0.002	0.020

We used the default fixed set of hyper-parameters following TS2Vec (Yue et al., 2022) regardless of the dataset or the downstream task. The only difference is that the number of optimization iterations was increased approximately 1.5 times on average. The batch size B is set to 8. The learning rate is 0.001. The final representation dimension D is set to 320, and the dimensions of timestamp-wise and feature-wise embedding, d_1 and d_2, were set to 64. The hidden dimension for encoders and element-wise gating layer was set to 320, equal to D. In addition, we fixed the maximum shift size as three for random shifting. Random binomial masking was applied to the timestamp-wise embedding but no masking for feature-wise embedding was applied by the ablation results in Table 4. According to Table 4, no significant improvements in terms of average accuracy or standard deviation were observed at every condition. The average accuracy decreases when the feature masking ratio increases, showing that each feature embedding contains helpful information.

We used ten blocks consisting of two dilated CNN layers for the temporal encoder with kernel size three and dilation size 2^l for the l^{th} block. A two-layer MLP network for feature encoder as well as reconstruction decoder was employed. A single linear layer was used for projecting feature-specific representation to the temporal axis. These architectural settings were fixed to all datasets and downstream tasks.

Table 5: Hyper-parameter sensitivity analysis results.

Batch Size			**Learning Rate**				**Representation Dim.**				**Recon. Loss**	
4	**8(*)**	**16**	**0.005**	**0.001(*)**	**0.0005**	**0.0001**	**64**	**128**	**320(*)**	**512**	**L1**	**L2(*)**
0.719	**0.722**	**0.722**	0.708	**0.722**	<u>0.720</u>	0.711	0.703	0.710	**0.722**	<u>0.721</u>	**0.725**	<u>0.722</u>

Additionally, we explored the hyper-parameter sensitivity by evaluating each hyper-parameter's effectiveness. In Table 5, we evaluated the model performance in terms of variation in batch size, learning rate, representation dimension, and reconstruction loss function. The numbers marked with * are the fixed hyper-parameters used for all our experiments. Considering the overall performance on 30 classification datasets, no harmful degradation occurs around the default settings. Based on this sensitivity analysis, we found the FEAT is not very sensitive to a slight change of hyper-parameters, which can be another advantage of practical implementation.

A.2 Dataset

UEA archive datasets UEA archive provides various types of datasets such as Human Activity, Audio Spectra, Electrocardiogram (ECG), Electroencephalogram (EEG), and Magnetoencephalography (MEG). The datasets are labeled per observed sequence.

Electricity Transformer Temperature (ETT) dataset ETT dataset[2] contains data collected by an electricity transformer over two years. In this study, we use three distinct datasets - ETTh$_1$, ETTh$_2$ and ETTm$_1$. ETTh$_1$ and ETTh$_2$ are 1-hour-level datasets observed from different counties in China, while ETTm$_1$ is a 15-minute-level dataset. As they have different granularities, we can also validate the robustness of our learned representations on the various granularities. The train/validation/test for the ETT datasets is 12/4/4 months.

Electricity Consuming Load (ECL) dataset ECL[3] is the electricity consumption dataset of 321 clients. It is preprocessed into 1-hour-level dataset following Zhou et al. (2021). The train/validation/test for the ECL dataset is 60%/20%/20%.

A.3 Supplementary analysis for multivariate time-series classification

Table 6: Total training time of FEAT, TS2Vec, and TS-TCC on 30 UEA datasets. The **bold** and <u>underlined</u> numbers indicate the best and second-best performances, respectively.

	Training Time (hours)		
	FEAT (Ours)	**TS2Vec**	**TS-TCC**
Training representation	<u>0.62</u>	**0.28**	3.15
Training classifier	**0.17**	<u>0.18</u>	1.12
Total	<u>0.79</u>	**0.46**	4.27

For further analysis, we compared the total training time in Table 6 on 30 UEA datasets of the top three best models in terms of the average accuracy. We reimplemented TS-TCC[4] and TS2Vec[5] based on the code from their official repositories. We followed their default settings, except for unifying the representation dimension and evaluation protocol for a fair comparison. As FEAT is a more sophisticated model than TS2Vec owing to its feature encoder and reconstruction decoder, TS2Vec records the shortest time on training representation. However, FEAT achieves the shortest time for training the classifier on the learned representation. It can be understood that although a longer time is required to learn time-series representation through our proposed framework, sufficiently rich representation is learnt so that the following classifier can converge earlier than the other methods. In comparison to TS-TCC, which is the transformer-based representation learning framework, the proposed FEAT, the Dilated CNN and MLP-based model, is found to be much lighter. FEAT is approximately 5.08 times faster on training representation and 6.59 times faster on converging SVM classifier compared to TS-TCC.

A.4 Supplementary analysis for multivariate time-series forecasting

Table 7 summarizes the time for training and inference of the top three best models on the ETTm$_1$ dataset in Table 2. We reimplemented Informer[6] and TS2Vec from the official code from their official repositories. We followed the default setting for Informer and TS2Vec except for the number of epochs; the number of epochs in TS2Vec was set to that of FEAT. Like the analysis mentioned in Appendix A.3, TS2Vec resulted in the shortest time in training and inference on the representation learning framework. In contrast, Informer, a transformer-based end-to-end forecasting model, resulted in the longest time in training as well as inference phases. Although FEAT resulted in a longer total running time than TS2Vec, the difference is not as significant as that between FEAT and Informer. Notably, FEAT is 7.22 times more efficient in total training time and 4.30 times faster in total inference time for all horizons on average compared to Informer.

[2]https://github.com/zhouhaoyi/ETDataset.

[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[4]https://github.com/emadeldeen24/TS-TCC

[5]https://github.com/yuezhihan/ts2vec

[6]https://github.com/zhouhaoyi/Informer2020

Table 7: Training and inference time (seconds) of FEAT, TS2Vec, and Informer on the ETTm$_1$ dataset. For the representation learning models, FEAT and TS2Vec, we report separate elapsed time for learning time-series representations and (with + signs) training a linear regressor on top of the learned representations. Inference time is the total elapsed time for encoding representations from the frozen network and regression predictions.

Phase	**H**	**FEAT**	**TS2Vec**	**Informer**
Training	**24**	101.66 + 2.74	86.74 + 2.88	724.4
	48	101.66 + 3.35	86.74 + 4.17	212.6
	96	101.66 + 4.56	86.74 + 5.04	712.88
	288	101.66 + 9.77	86.74 + 9.82	1039.56
	672	101.66 + 19.34	86.74 + 19.16	1342.78
Inference	**24**	5.26 + 0.02	4.02 + 0.01	20.72
	48	5.26 + 0.02	4.02 + 0.02	4.85
	96	5.26 + 0.03	4.02 + 0.03	19.94
	288	5.26 + 0.08	4.02 + 0.08	29.95
	672	5.26 + 0.15	4.02 + 0.13	39.43

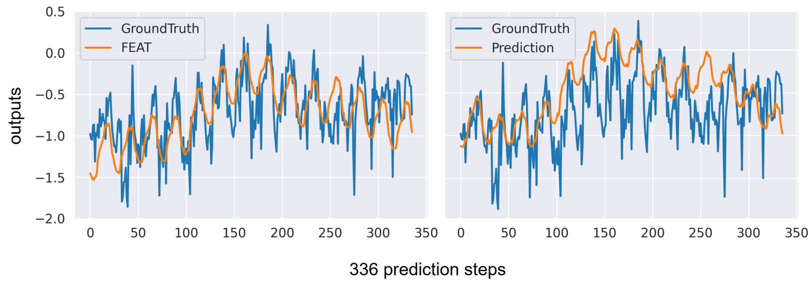


Figure 5: Prediction slice (H=336) of FEAT (left column) and TS2Vec (right column) on ETTh$_2$. Each prediction was plotted from the best epoch model. The blue lines stand for the ground truth while the orange lines stand for the prediction.

Table 8: Summary of ablation study results of multivariate time-series forecasting on 3 ETT datasets. We recorded the average MSE and MAE scores of five repetitions. The second and fourth columns denote the amount of performance degradation compared to each metric's best performance.

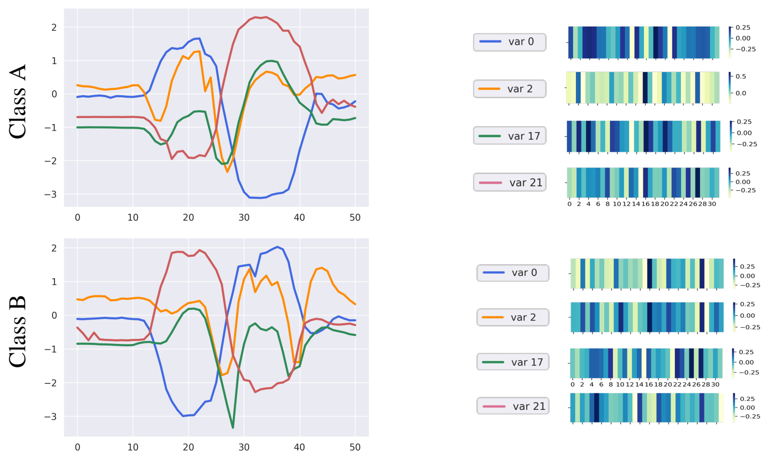
		Avg. MSE		**Avg. MAE**	
	FEAT	**0.901**		**0.683**	
Loss	w/o temporal contrasting	1.165	(+0.264)	0.787	(+0.104)
	w/o feature contrasting	0.909	(+0.008)	0.688	(+0.005)
	w/o temporal & feature contrasting	1.168	(+0.267)	0.786	(+0.103)
	w/o reconstruction	0.978	(+0.077)	0.708	(+0.025)
Module	MLP feature encoder				
	$\rightarrow$ Depth-wise dilated CNN	1.017	(+0.116)	0.715	(+0.032)
	Gating				
	$\rightarrow$ concatenation + projection	0.944	(+0.043)	0.700	(+0.017)
	$\rightarrow$ mean	0.943	(+0.042)	0.699	(+0.016)
	w/o random shifting	0.910	(+0.009)	0.686	(+0.003)
	w/o random masking	0.981	(+0.080)	0.703	(+0.020)

Figure 5 shows a slice of the target values and forecasting results of FEAT and TS2Vec on the ETTh$_2$ test dataset with a length of 336. It shows that the predictions from our proposed model are better fitted to the ground truths with trends and seasonal patterns rather than the TS2Vec representation learning framework.

In addition, Table 8 summarizes ablation study results on multivariate times-series forecasting task. It was performed on three ETT datasets (ETTh$_1$, ETTh$_2$, and ETTm$_1$), and the results are the averaged MSE and MAE scores of the five repetitions at all forecasting horizons. In terms of both metrics, all ablations exhibit degradation, and the full model achieved the best, which supports that each component of FEAT contributes to performance improvement.

A.5 Supplementary Figure

A.5.1 Feature-specific representation



(a) Original data (b) Feature-specific representation

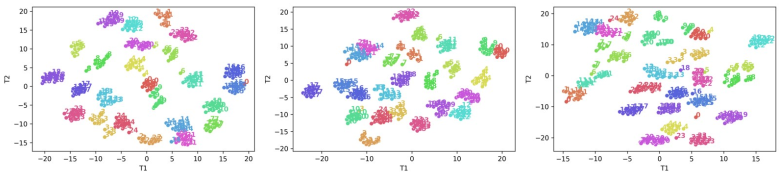
Figure 6: Heatmap visualizations of feature-specific representation from FEAT. Instances with different classes were sampled from *NATOPS* dataset and displayed by row. Sampled subset features of each class are plotted from the original time-series data in the first column, while the feature-specific representations by FEAT are displayed in the second column. Note that we undersampled feature-specific representation dimensions from 320 to 32.

Figure 6 presents the heatmap visualizations of feature-specific representation from FEAT. For comparison, we selected the sample sequences with different classes having different feature patterns. Specifically, as shown in Figure 6a, these two sequences have similar temporal patterns for the orange and green line, but opposite patterns for the red and blue line. We verified that our feature encoder can learn these unique characteristics by visualizing the feature-specific representations in Figure 6b. Feature contrasting, as well as feature encoder, helps FEAT to model the feature-specific patterns explicitly. Specifically, the two opposite patterns of the red and blue line in Figure 6a were represented in opposite manners for each class.

A.5.2 T-SNE PLOTS WITH THE SILHOUETTE SCORES FOR 30 UEA DATASETS

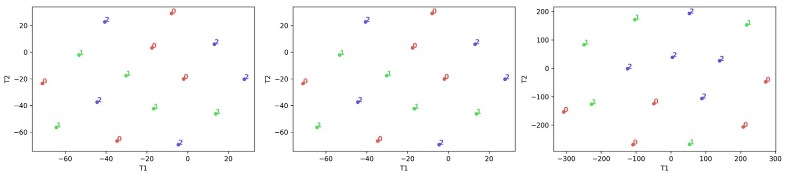
FEAT (Ours) **TS2Vec** **Original**

ArticularyWordRecognition
(25 classes)



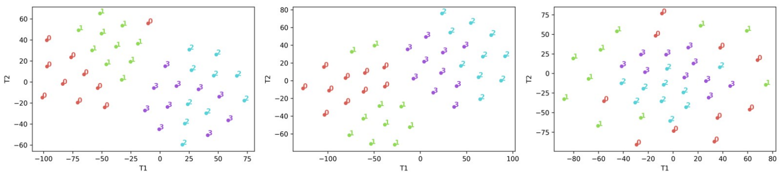
0.227 **0.190** **0.220**

AtrialFibrillation
(3 classes)



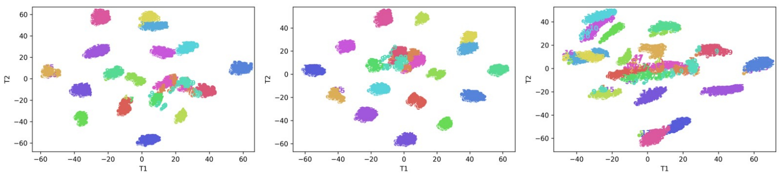
0.398 **0.392** **0.267**

BasicMotions
(4 classes)



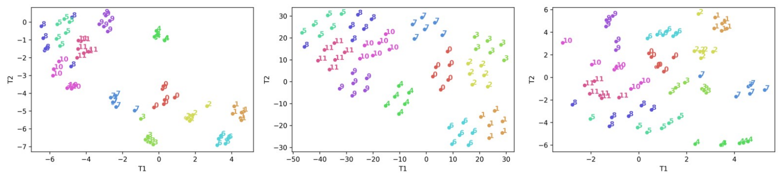
0.537 **0.427** **0.224**

CharacterTrajectories
(20 classes)



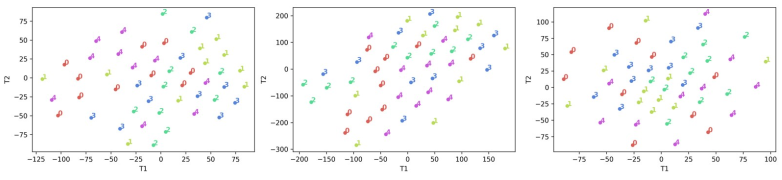
0.311 **0.317** **0.243**

Cricket
(12 classes)



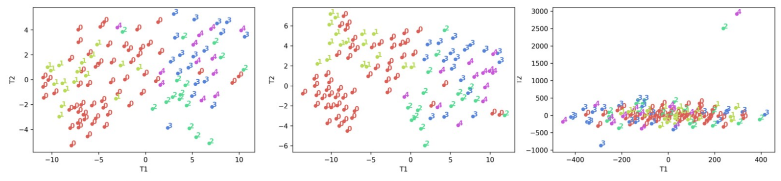
0.272 **0.264** **0.119**

DuckDuckGeese
(5 classes)



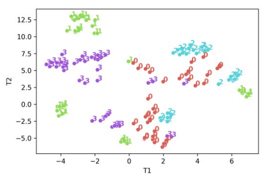
0.222 **0.342** **0.544**

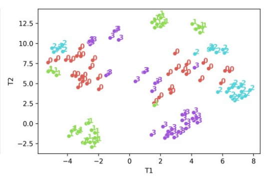
EigenWorms
(5 classes)

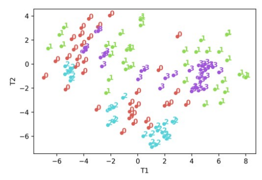


0.294 **0.240** **0.040**

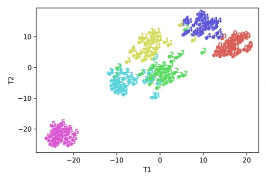
Epilepsy
(4 classes)

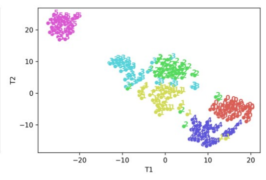

0.277

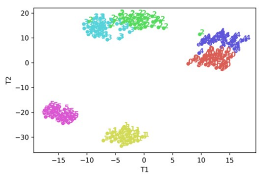

0.299


0.149

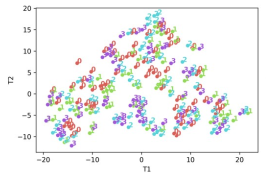
ERing
(6 classes)

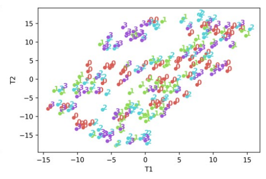

0.184

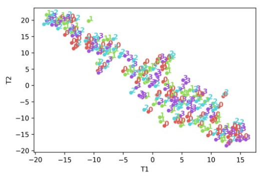

0.177


0.244

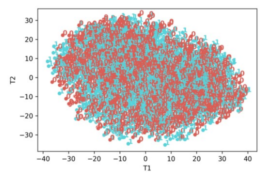
EthanolConcentration
(4 classes)

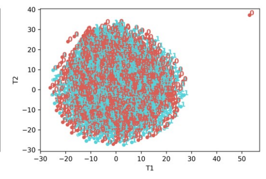

0.194

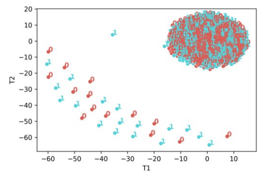

0.184


0.290

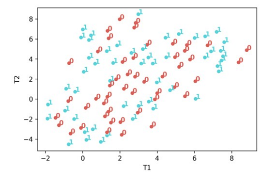
FaceDetection
(2 classes)

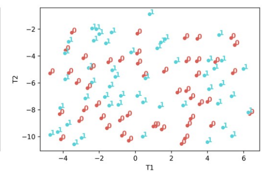

0.904

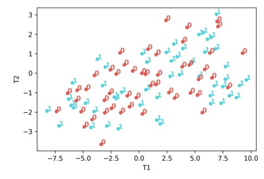

0.929


0.866

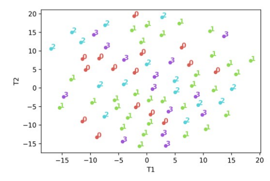
FingerMovements
(2 classes)

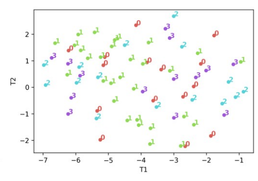

0.202

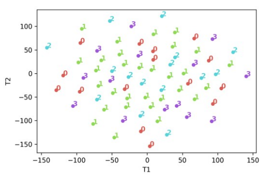

0.114


0.348

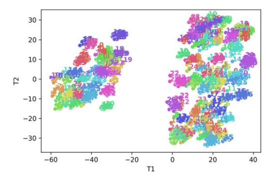
HandMovementDirection
(4 classes)

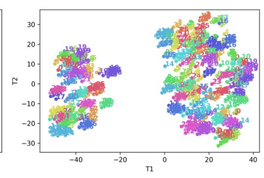

0.036

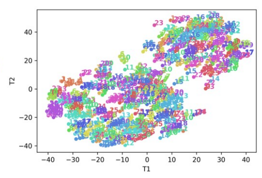

0.029


0.033

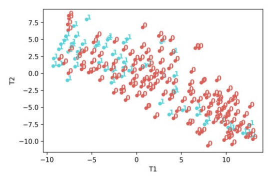
Handwriting
(26 classes)

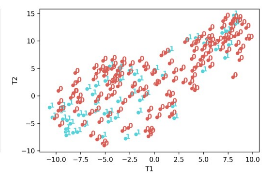

0.072

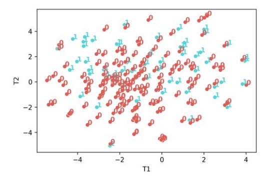

0.070


0.062

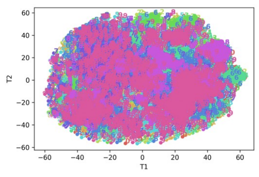
Heartbeat
(2 classes)

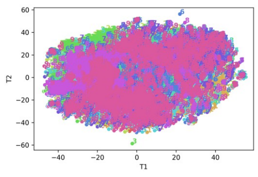

0.946

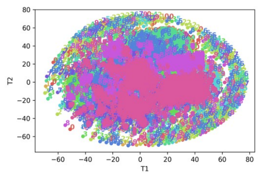

0.959


0.950

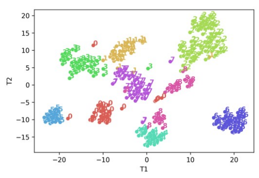
InsectWingbeat
(10 classes)

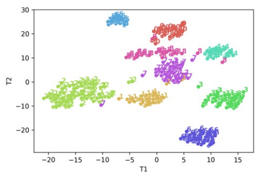

0.520

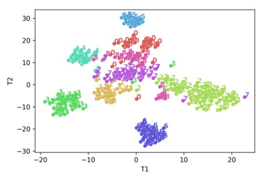

0.301


0.609

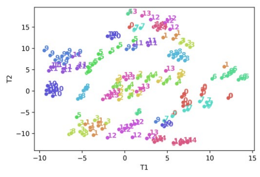
JapaneseVowels
(9 classes)

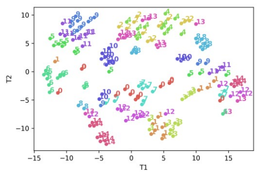

0.181

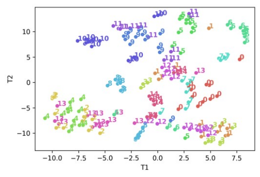

0.236


0.172

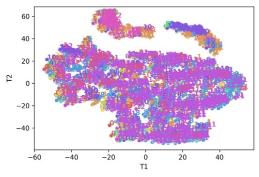
Libras
(15 classes)

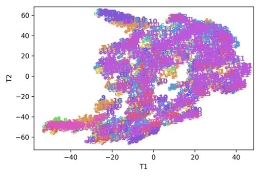

0.228

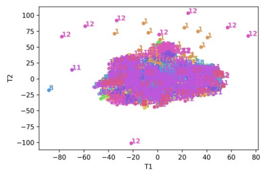

0.202


0.253

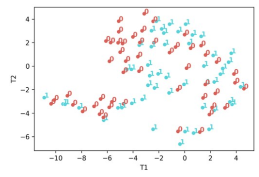
LSST
(14 classes)

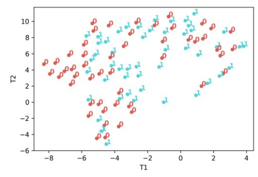

0.572

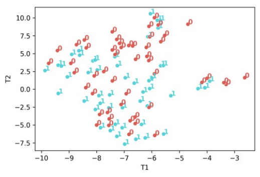

0.576


0.923

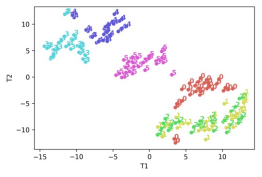
MotorImagery
(2 classes)

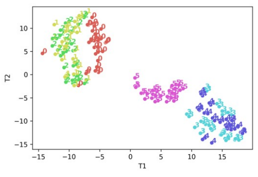

0.893

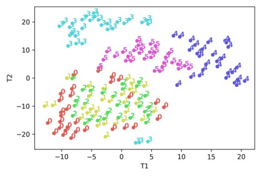

0.901


0.841

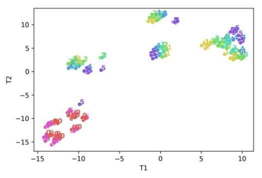
NATOPS
(6 classes)

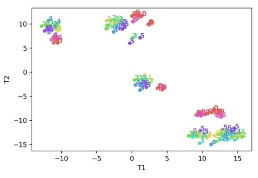

0.215

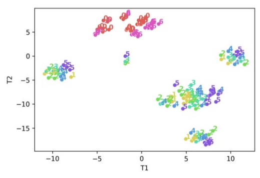

0.213


0.145

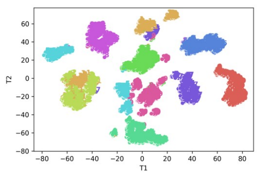
PEMS-SF
(7 classes)

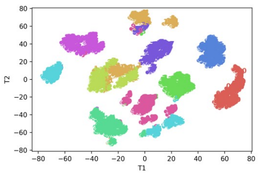

0.341

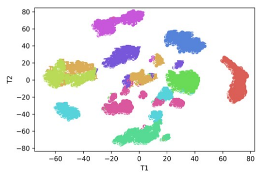

0.266


0.266

PenDigits
(10 classes)


0.260


0.262


0.331

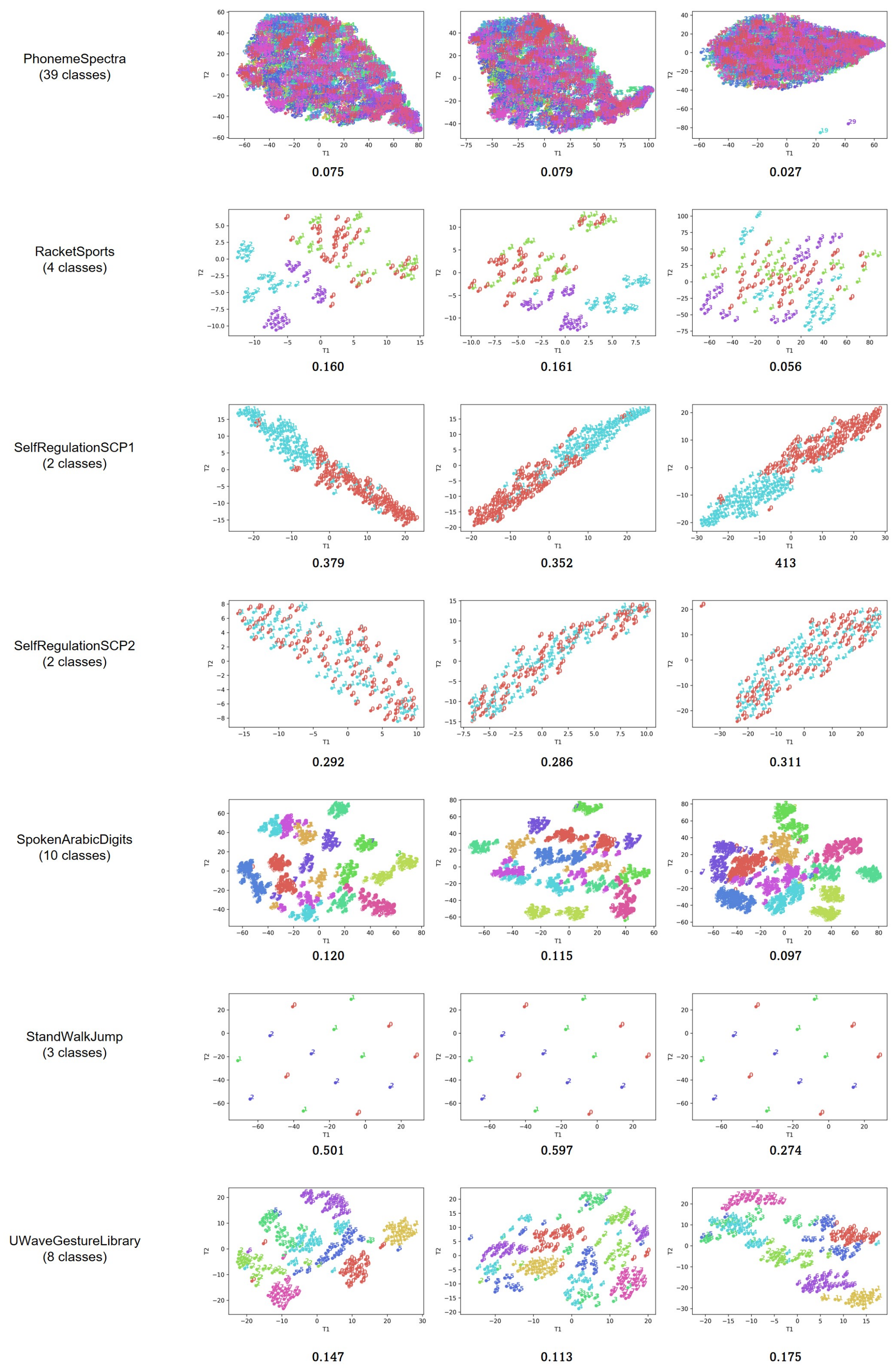


Figure 7: T-SNE visualization of representations from FEAT (first column) and TS2Vec (second column) and original data (third column). Thirty datasets from the UEA archive were plotted from the first row. The numbers located at the bottom of the plots are the Silhouette scores. The silhouette scores are calculated with the Euclidean distance metric, where the number of clusters equals the number of classes.