SMATE: Semi-Supervised Spatio-Temporal Representation Learning on Multivariate Time Series

Jingwei Zuo, Karine Zeitouni and Yehia Taher
DAVID Lab, University of Versailles, Université Paris-Saclay, Versailles, France
Email: {jingwei.zuo, karine.zeitouni, yehia.taher}@uvsq.fr

***Abstract*—Learning from Multivariate Time Series (MTS) has attracted widespread attention in recent years. In particular, label shortage is a practical challenge for the classification task on MTS, considering its complex dimensional and sequential data structure. Unlike self-training and positive unlabeled learning that rely on distance-based classifiers, in this paper, we propose SMATE, a novel semi-supervised model for learning the interpretable Spatio-Temporal representation from weakly labeled MTS. We validate empirically the learned representation on 22 public datasets from the UEA MTS archive. We compare it with 13 state-of-the-art baseline methods for fully supervised tasks and four baselines for semi-supervised tasks. The results show the reliability and efficiency of our proposed method.**

***Index Terms*—Neural Networks, Multivariate Time Series, Semi-supervised Learning, Representation Learning**

I. Introduction

Most Multivariate Time Series (MTS) data, such as sensor readings, are labeled during the data collection process. The post-labeling of MTS is much more costly than classic data (e.g., image, text, etc.) due to the low interpretability over the real-valued sequence, leading to a considerable constraint for MTS classification in real-life scenarios.

Weakly supervised learning becomes an alternative option for the fully supervised algorithm by learning from the unlabeled samples. The previous studies on weak-label Time Series (TS) learning are usually based on self-training [1] or Positive Unlabeled Learning [2] with a carefully designed distance measure [3] or stopping criterion [4] to learn the pseudo-labels. Besides, they mostly focus on the Univariate Time Series (UTS) with the One-Nearest-Neighbor (1NN) classifier on raw data space, which is widely outpaced by today's advanced approaches [5], such as Deep Neural Networks (DNNs) [6] or ensemble methods [7].

From Univariate Time Series (UTS) to Multivariate Time Series (MTS), traditional methods usually combine the compact and effective features from different variables, such as Shapelet Ensemble [8]–[10], global discriminative patterns [11], or bag-of-patterns [12], [13]. However, the predefined features usually fail to capture MTS essentials: the temporal dependency and the interactions of multiple variables (i.e., *spatial interactions*, we use the term *spatial* in this paper for the variable axis). Recently, some deep learning-based methods were proposed to capture the MTS features with various network structures [14]–[17], showing promising performance on MTS classification tasks. However, the above-mentioned methods are mostly fully supervised, and rarely consider the label shortage issue when building the MTS classifier.

The recent research turns to Representation Learning [18] when handling weakly labeled MTS, which allows learning low dimensional embeddings in an unsupervised manner, such as using triplet loss [19] to form the embedding space, then even an SVM classifier is powerful enough on the learned representation [20]. However, existing techniques suffer from three major issues. First, the interactions between the MTS variables are generally computed on the entire 1-D series, ignoring the fact that the local spatial interactions at the sub-sequence level may evolve in the dynamic sequence, that is $spatial\ dynamics$. Second, the representation learned in a pure unsupervised manner depends mostly on the loss function selection. As no label information is utilized to learn the representation [19], there is a risk that it deviates from the true features, thus affecting the classifier performance. Third, they rather employ deep learning as a blind box and do not focus on the interpretability of the learned representation.

Therefore, to handle both the MTS complex structure and the label shortage problem, we propose SMATE, **S**emi-supervised Spatio-temporal representation learning on **M**ultiv**A**riate **T**ime **SE**ries. The auto-encoder based structure allows mapping the MTS samples from raw features space $\mathcal{X}$ to low dimensional embedding space $\mathcal{H}$. A Spatial Modeling Block combined with a multi-layer convolutional network captures the spatial dynamics, whereas a GRU-based structure extracts the temporal dynamic features. Thereby, SMATE is capable of compressing the essential Spatio-temporal characteristics of MTS samples into low-dimensional embeddings. On top of this embedding space $\mathcal{H}$, we propose a semi-supervised three-step regularization process to bring the model to learn class-separable representations, where both the labeled and unlabeled samples contribute to the model's optimization. This regularization process comes with the capability of visualization at each step, making SMATE interpretable.

We summarize the paper's main contributions as follows:

- **Spatio-temporal dynamic features in MTS:** We claim and demonstrate that the temporal dependency and the evolution of the spatial interactions (*spatial dynamics)* are important for building a reliable MTS embedding.
- **Weak supervision on learning representations:** With limited labeled data, SMATE can learn reliable class-separable MTS representations for downstream tasks, such as MTS classification (MTSC).

- **Interpretable MTS embedding learning:** SMATE allows for visual interpretability, not only from the class-separable representations but also in each step of the semi-supervised regularization process.
- **Extensive experiments on the MTS datasets:** The experiments are carried out on various MTS datasets from different application domains. The detailed evaluation with 13 supervised baselines and four semi-supervised work is provided, which shows the effectiveness and the efficiency of SMATE over the state of the art.

The rest of this paper starts with a review of the most related work. Then, we formulate the problems of the paper. Later, we present in detail our proposal SMATE, which is followed by the experiments on real-life datasets and the conclusion.

II. Related Work

We begin by discussing the related work on learning MTS representation with the main extension to MTS Classification (MTSC) tasks. Then, we briefly review the previous work on semi-supervised Time Series learning. Table I shows the comparison of the methods for learning MTS representation.

A. Multivariate Time Series Representation Learning

Firstly, we give two primary definitions.

Definition 1 (univariate time series): A univariate time series $\mathbf{x} \in \mathcal{R}^T$ is a sequence of real-valued numbers: $\mathbf{x}$=(x_1, x_2, ..., x_i, ..., x_T), where T is the sequence length.

Definition 2 (multivariate time series): A multivariate time series $\mathbf{x} \in \mathcal{R}^{T \times M}$ is a sequence of real-valued vectors: $\mathbf{x}$=(x_1, x_2, ..., x_i, ..., x_T), where $x_i \in \mathcal{R}^M$, M is the variable numbers.

A natural way to learn MTS representation is to extend the representation approaches developed earlier on Univariate Time Series (UTS) [21]. This is the case in [22] where the authors further explored Singular Value Decomposition (SVD) with multi-view learning to find the consistency and interactions between variables. Similarly, [8] [9] combine Shapelet representation from different variables to build an ensemble-like learner. Symbolic Representation for Multivariate Time Series (SMTS) [12] adopts the Bag-of-Patterns concept, considering all variables simultaneously and constructs a code-book to model the variable relations. Finally, WEASEL+MUSE [13] extend WEASEL [23] from UTS to MTS by creating a histogram of feature counts to capture the local and global changes in relationships between variables.

Different from the interpretable feature-based representations, various deep learning models are proposed to capture the variable (*i.e., spatial*) interactions in MTS. Multi-Channels Deep Convolutional Neural Networks (MC-DCNN) [15] extract firstly 1D-CNN features from each variable, then combine them with a Fully Connected (FC) Layer. Whereas the authors in [24] abandon the combination strategy but apply directly 1D-CNN to all variables. The 2D-CNN features with the cross-attention mechanism in CA-SFCN [25] enhanced the dependencies captured by 1D-CNN on both temporal and spatial axes. Besides, the recurrent models are widely applied to sequential data. A modified Gated Recurrent Unit (GRU) described in [16] models MTS with missing values, where each multivariate step is memorized into state units, then the recurrent structure captures the temporal dependencies. Another group of works [26], [27] adopt Graph Neural Networks (GNNs) to model the spatial interactions. However, they are generally designed for forecasting tasks (e.g., traffic forecasting), and most rely on external information (e.g., the road networks) between the variables. Last but not least, the hybrid LSTM-CNN structure is capable of extracting both local and long-term features. Various work such as the Squeeze-and-Excitation block in MLSTM-FCN [14] or the multi-view learning-like module in TapNet [28], enhanced the hybrid structure via modeling the spatial interactions. However, the interactions are generally computed at the sequence level, ignoring the fact that the local spatial interactions at the sub-sequence level may evolve in the dynamic sequence, i.e., *spatial dynamics*. Moreover, they are all fully supervised, requiring huge labels during the training process. Also, the learned representations are result-oriented (e.g., pursuing higher accuracy), with less focus on the interpretability, considered by the data mining community.

TABLE I: Existing methods for learning MTS Representation

	SMTS	MUSE	Shapelet	USRL	TapNet	MLSTM-FCN	CA-SFCN	1NN-DTW	**SMATE**
Temporal Dynamics	-	-	-	✓	✓	✓	✓	-	✓
Spatial Dynamics	-	-	-	-	-	-	✓	-	✓
Interpretable Representation	✓	✓	✓	-	✓	-	-	-	✓
Label Shortage	-	-	-	✓	✓	-	-	✓	✓

B. Semi-supervised Learning on Time Series

The pioneering work [1], [3] on Semi-supervised TS Learning are based on self-training or Positive Unlabeled Learning [2] with the Nearest-Neighbor (1NN) classifier and a carefully designed distance, such as DTW [1] or DTW-D [3], and optimized stopping criterion [4] for importing the pseudo-labels. Those work are followed by [29], [30] for wider contexts. Though not mentioned in their papers, the self-training framework is extensible from the UTS to the MTS setting by using an adapted distance, such as DTW$_I$ [31], DTW$_D$ [31] or DTW$_A$ [32]. However, under more complex scenarios nowadays, such as 30 UEA MTS datasets [33] collected from different domains, the distance-based classifiers show limited performance and are rather used as baselines by recent studies [5].

Learning meaningful MTS representations [18] in a weakly supervised setting draws much attention recently. Unsupervised Scalable Representation Learning (USRL) described in [19] combines causal dilated convolutions with triplet loss for contrastive learning. On the one hand, it learns better UTS representation than the traditional supervised CNNs [6]. On the other hand, a single SVM on the learned MTS representation offers higher accuracy than a DTW$_D$-based classifier [31]. Similarly, authors in [30] adopt Multi-Task Learning (MTL) to learn the self-supervised UTS features from an auxiliary forecasting task. The recent work Semi-TapNet [28] proposes an Attentional Prototype Network to learn from the unlabeled samples. However, the above-mentioned approaches do not explore thoroughly both the labeled samples and the *Spatio-temporal dynamics* in MTS.

III. Problem Formulation

In this section, we formulate firstly the *Spatio-temporal dynamics* learning problem in MTS. Then, we give a formal definition of a semi-supervised classification problem for MTS. Table II summarizes the notations used in the paper.

A. Spatio-temporal Representation for MTS

TABLE II: Notation

Notation	Description
$\mathcal{D}, \mathcal{D}_l, \mathcal{D}_u$	dataset, labeled portion, unlabeled portion
T, M, N	MTS length, variable numbers, dataset size
L, D	embedding length, embedding dimension size
$m, \mathcal{P}$	temporal window size, embedding pool size
$\mathbf{x}$, $\mathbf{h}$	MTS instance, latent embedding
$\mathbf{s}$	variable/spatial interaction
θ	general parameters to be optimized

The Spatio-temporal modeling of MTS requires considering both the temporal dependency $p(x_{t'}|x_t)$ $(t' > t)$ and the spatial interactions between the variables. Previous studies [14], [28] usually consider the spatial interactions at the sequence level: $\mathbf{s} = \{\mathbf{x}^i \bowtie \mathbf{x}^j\} \in \mathcal{R}^M$, where $\mathbf{x}^i, \mathbf{x}^j \in \mathcal{R}^{T\times 1}$, $\bowtie$ indicates the interactions between the variables. However, the local spatial interactions at the subsequence level $s_t = \{\mathbf{x}^i_{t-m/2,t+m/2} \bowtie \mathbf{x}^j_{t-m/2,t+m/2}\} \in \mathcal{R}^M$ may evolve in the dynamic sequence, where m is the window size. To illustrate, given the system status at time t in MTS, it is not only decided by the local value $x_t \in \mathcal{R}^M$ given a temporal status, but also by its neighbor values $\left[x_{t-m/2} : x_{t+m/2}\right] \in \mathcal{R}^{M\times m}$, which brings a spatial correlation matrix on temporal neighbors and spatial variables given a spatial status $s_t \in \mathcal{R}^M$. Unlike the work such as DCRNN [27] relying on external information (e.g., the road networks) to model the spatial dependency between the variables, we aim at exploring the inherent spatial/variable interactions only on the MTS data.

Therefore, given a sample $\mathbf{x} \in \mathcal{R}^{T\times M}$ in raw space $\mathcal{X}$, the Spatio-temporal representation learning for MTS is to learn a low-dimensional representation $\mathbf{h} \in \mathcal{R}^{L\times D}$ on embedding space $\mathcal{H}$, integrating both temporal dynamic $p(x_{t'}|x_t)$ and spatial dynamic $p(s_{t'}|s_t)$. The item *dynamic* refers to the unstable system status within the evolving multivariate sequential data.

B. Semi-Supervised Learning on MTS

Definition 3 (weak-label MTS dataset): A weakly labeled MTS dataset $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$ contains both labeled and unlabeled MTS samples:

$$\mathcal{D}_l = \{\mathbf{x}_i, y_i\}_{i=1}^{N*r},\ \mathcal{D}_u = \{\hat{\mathbf{x}}_i\}_{i=1}^{N*(1-r)}$$

where r $(0 \leq r \leq 1)$ indicates the *ratio* of the labeled samples in $\mathcal{D}$ of size N, y_i is the annotation of the labeled instance $\mathbf{x}_i$.

The semi-supervised MTS learning aims at training a classifier to predict successfully the label of a testing MTS sample, adopting the supervised training from $\mathcal{D}_l$ and further unsupervised adjustment/optimization from $\mathcal{D}_u$.

IV. Proposal: SMATE

In this section, we introduce SMATE, which captures the essential characteristics of Multivariate Time Series (MTS) and allows learning an interpretable representation space in a semi-supervised manner. This section is organized as follows. First, we introduce the global model structure of SMATE. Then, we describe how the Spatio-temporal representation is learned from the raw MTS data space. Finally, we give the joint model optimization, which coordinates the weak supervision and the embedding learned via a three-step regularization process.

A. Global Structure of SMATE

SMATE is based on an *asymmetric auto-encoder* structure, integrating three key components: Spatio-temporal dynamic encoder, sequential decoder, and semi-supervised three-step regularization of the embedding space.

Given the fact that extracting features from a high-dimensional space generally requires additional attention compared to restoring data from a low-dimensional space [18], the encoding and decoding process in SMATE adopts different weight matrix (i.e., *asymmetric auto-encoder*) to capture the inner structure of MTS data. Although recent work [34] does represent the temporal dynamics of MTS via a sequence to sequence (seq2seq) model, they do not encompass the complex Spatio-temporal structure of MTS. As shown in Figure 1, SMATE adopts a two-channel encoder exploring both spatial and temporal dynamics and embeds the input MTS into a low-dimensional representation space, where the embedded samples are sparsely distributed with the reconstruction-based optimization. On the unsupervised embedding space, a three-step regularization is applied to learn class-separable embeddings. The class centroids are regularized by labeled and unlabeled samples, showing interpretability over the representation space. Finally, the model is jointly optimized by the reconstruction objective and the regularization objective.

B. Spatial Modeling Block (SMB)

Firstly, we introduce a novel module, Spatial Modeling Block (SMB), to capture the spatial interactions at subsequence levels. As shown in Figure 2, SMB takes as input an MTS representation $h = \{h_i\}_{i=1}^T \in \mathcal{R}^{T\times d}$ $(d = M$ for the first block in spatial encoding channel), followed by a one-dimensional average pooling layer on each variable $h^j \in \mathcal{R}^{T\times 1}$, encoding the temporal neighbors into the horizontal system status $s_H(i)$=$avg(\left[h_{i-m/2} : h_{i+m/2}\right])$, where i is the time tick, m is the window size. Then the Fully Connected (FC) layers allow firstly the interaction of the horizontal system status s_H in the vertical direction via a low-dimensional compression $s_V \in \mathcal{R}^{T\times d'}$, then remapping it to the initial data space to decide the spatial interaction weights at each one-dimensional segment. We define the spatial interactions $\mathbf{s} = \{s_i\}_{i=1}^T$, where $s_i \in \mathcal{R}^d$, representing the interaction weights for the vector $h_i \in \mathcal{R}^d$. The output of SMB is described by h'=$h \odot \mathbf{s}$, with the calibrated weights for each 1-D TS segment, where $\odot$ is the element-wise multiplication.

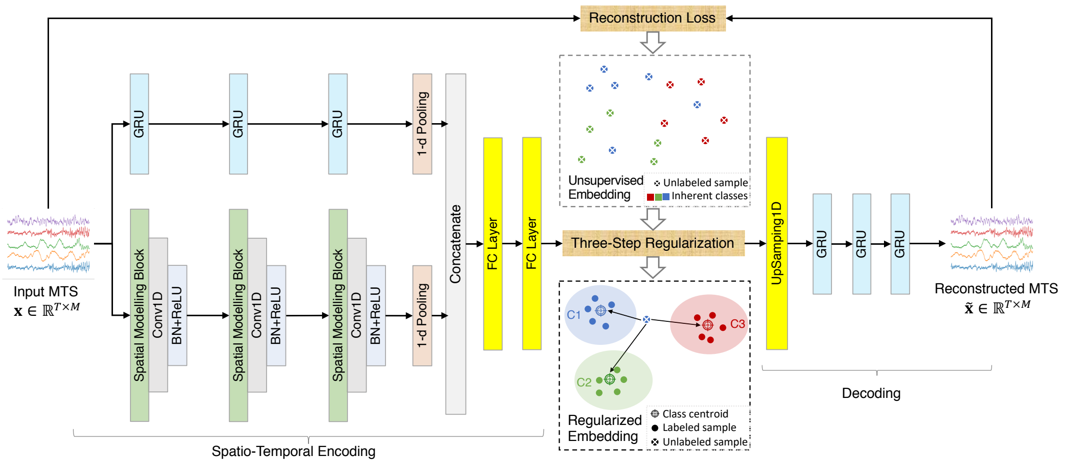


Fig. 1: Model Structure of SMATE

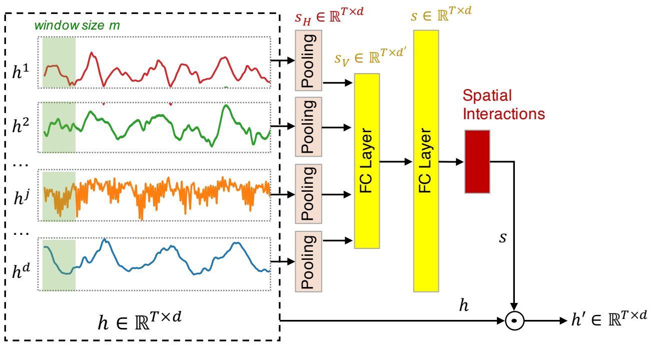

window size m
$s_H \in \mathbb{R}^{T \times d}$
$s_V \in \mathbb{R}^{T \times d'}$
$s \in \mathbb{R}^{T \times d}$
h^1
h^2
h^j
h^d
Pooling
FC Layer
Spatial Interactions
s
h
$h \in \mathbb{R}^{T \times d}$
$h' \in \mathbb{R}^{T \times d}$

Fig. 2: The Spatial Modeling Block (SMB)

C. *Spatio-Temporal Encoding on MTS*

Given $\mathbf{x} \in \mathcal{R}^{T \times M}$, the low-dimensional representation $\mathbf{h} \in \mathcal{R}^{L \times D}$ embeds the Spatio-temporal features of $\mathbf{x}$ by a neural network-based function $f_\theta(\mathbf{x})$. The low-dimensional embedding brings dramatic improvement on both the efficiency and accuracy for classification tasks [28], owning to the fact that the classifier is not distracted by the redundant information in raw data.

As shown in Figure 1, we adopt a two-channel structure to encode the spatial and temporal features in MTS, respectively. For the temporal channel, among different variants of the recurrent neural networks (RNN), we specifically consider Gated Recurrent Units (GRUs) [16] that mitigate the vanishing gradient problem. An update gate z_t and a reset gate r_t control the hidden state $h_t \in \mathcal{R}^{d_g}$ with the observation $x_t \in \mathcal{R}^M$ and the previous hidden state $h_{t-1} \in \mathcal{R}^{d_g}$, where d_g is the output dimension of GRUs. The update functions are as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{1}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{2}$$

$$h_t = (1-z_t) \odot h_{t-1} + z_t \odot \tanh(W_h x_t + U_h(h_{t-1} \odot r_t) + b_h) \tag{3}$$

where W_x, U_x, b_x $(x \in [r, z, h])$ are model parameters, σ is the sigmoid function, $\odot$ is the element-wise multiplication. Three GRUs are cascaded with a 1-D pooling layer to output $\mathbf{h}(\mathbf{T}) \in \mathcal{R}^{L \times d_g}$, where L=$T/\mathcal{P}$, $\mathcal{P}$ is the pool/sampling size[1].

[1]Note that the pool/window size m in SMB and $\mathcal{P}$ are different parameters.

For the spatial channel, we define the convolutional module:

$$\mathbf{h}'(l) = SMB(\mathbf{h}(l)) \tag{4}$$

$$\mathbf{h}(l+1) = ReLU(BN(W \otimes \mathbf{h}'(l) + b)) \tag{5}$$

where l $(0 \le l < 3)$ is the module number, $\mathbf{h}(0)$=$\mathbf{x} \in \mathcal{R}^{T \times M}$, $\mathbf{h}(l) \in \mathcal{R}^{T \times d_c}$, d_c is the filter number, W is the 1-D convolutional kernel of size m, $\otimes$ is the convolution operator. Within each of the three modules, the SMB firstly calibrates the interaction weights for each 1-D segment and outputs $\mathbf{h}' \in \mathcal{R}^{T \times d}$. Then a 1-D convolutional layer concatenated with Batch Normalization [35] and Rectified Linear Units (ReLU) [36] is deployed. The 1-D kernels match with the window size m in their neighboring SMB, as the convolution product $W \otimes \left[h'_{i-m/2} : h'_{i+m/2}\right]$ requires considering the spatial interactions captured by SMB within the same interval. Similar to the temporal channel, a 1-D pooling layer is applied after the last convolutional block to output $\mathbf{h}(\mathbf{S}) \in \mathcal{R}^{L \times d_c}$. Finally, we output the combined spatial and temporal features h_{concat} = $concat(\mathbf{h}(\mathbf{T}), \mathbf{h}(\mathbf{S})) \in \mathcal{R}^{L \times (d_g + d_c)}$ and apply two FC layers to get the Spatio-temporal embedding $\mathbf{h} \in \mathcal{R}^{L \times \mathbf{D}}$. The matrix representation allows the maximal preservation of the Spatio-temporal features and facilitating the MTS restoration. The detailed parameter settings can be found in Section V-A2.

D. *Joint Model Optimization*

As shown in Figure 1, since the representation learned via an autoencoder-based structure generally has a sparse distribution of class-specific samples [18], the unsupervised training derived from the reconstruction objective does not consider thoroughly the inner relation between class-specific samples but focus on the restoration performance from the embeddings. To address the issue, we propose a joint model optimization that integrates the temporal reconstruction and the three-step regularization objectives. Specifically, the joint optimization combines both the labeled and unlabeled samples to learn the class-specific clusters on the embedding space.

Firstly, we define the *temporal reconstruction loss* as:

$$L_R = \sum\nolimits_t \|x_t - \tilde{x}_t\|_2 \tag{6}$$

where $x_t, \tilde{x}_t \in \mathcal{R}^M$, corresponding to the observations in the raw and reconstructed MTS instances $\mathbf{x}$ and $\tilde{\mathbf{x}}$.

Then, we introduce the three-step regularization combining both labeled and unlabeled samples to foster the model adaptation of class-separable embeddings. The regularization approaches the embeddings within the class-specific clusters to the virtual class centroids, which are trained progressively. ***Step 1* -Supervised Centroids Initialization**: The class centroids are initialized by the class-specific embeddings. Given the labeled training set $\mathcal{D}_l = \{X^k\}_{k=1}^K$ where K is the class number, $X^k \in \mathcal{R}^{N_k \times T \times M}$ is a sample collection of class k, N_k is the sample number of class k. Then the embedding set H^k=$f_\theta(X^k) \in \mathcal{R}^{N_k \times L \times D}$ initializes the class centroid $\mathbf{c}_k$ by:

$$\mathbf{c}_k = mean(H^k), \quad \mathbf{c}_k \in \mathcal{R}^{L \times D} \tag{7}$$

***Step 2* -Supervised Centroids Adjustment**: Once the centroids are initialized, we can make the supervised adjustment since the distance-based class probability allows to assess the contribution of individual samples on the centroid's decision. In other words, the centroid $\mathbf{c}_k$ is affected by larger contribution weights brought by nearby samples of class k. Let $\mathbf{x}_i^k \in \mathcal{R}^{T \times M}$ be a time series of class k, we define the weight of $\mathbf{x}_i^k$ to $\mathbf{c}_k$ as the inverse Euclidean Distance (ED) between the embedding $\mathbf{h}_i^k = f_\theta(\mathbf{x}_i^k) \in \mathcal{R}^{L \times D}$ and the centroid $\mathbf{c}_k$:

$$W_{k,i} = 1 - \frac{ED(\mathbf{h}_i^k, \mathbf{c}_k)}{\sum_{j=1}^K ED(\mathbf{h}_i^k, \mathbf{c}_j)} \tag{8}$$

Then the class centroid $\mathbf{c}_k$ can be adjusted accordingly by the labeled samples within the class-specific cluster:

$$\mathbf{c}_k = \sum\nolimits_{i=1}^{N_k} W_{k,i} \cdot \mathbf{h}_i^k, \quad \mathbf{h}_i^k \in H^k \tag{9}$$

***Step 3* -Unsupervised Centroids Adjustment**: Given the unlabeled samples $\mathcal{D}_u = \{\hat{\mathbf{x}}_i\}_{i=1}^{N*(1-r)}$, where r is the labeled data ratio. Apart from the optimization from the reconstruction objective, the unlabeled sample $\hat{\mathbf{x}}_i$ is capable of adjusting the centroid c_k via the propagated label from the distance-based class probability defined as:

$$\hat{p}_\theta(y = k|\hat{\mathbf{x}}_i) = 1 - \frac{ED(f_\theta(\hat{\mathbf{x}}_i), \mathbf{c}_k)}{\sum_{j=1}^K ED(f_\theta(\hat{\mathbf{x}}_i), \mathbf{c}_j)} \tag{10}$$

The unlabeled sample $\hat{\mathbf{x}}_i$ will be then integrated into the class-specific cluster with the highest probability. We can further adjust the class centroid $\mathbf{c}_k$ considering the unlabeled samples:

$$\mathbf{c}_k = \frac{N_k}{N_k + \hat{N}_k} \sum_{i=1}^{N_k} W_{k,i} \cdot \mathbf{h}_i^k + \frac{\hat{N}_k}{N_k + \hat{N}_k} \sum_{i=1}^{\hat{N}_k} \hat{p}_{k,i} \cdot \hat{\mathbf{h}}_i^k \tag{11}$$

where $\hat{\mathbf{h}}_i^k = f_\theta(\hat{\mathbf{x}}_i^k)$, $\hat{N}_k$ is the number of samples of class k in $\mathcal{D}_u$ with the propagated label.

The class centroids are initialized and adjusted by both labeled and unlabeled samples on the embedding space, from which we formalize the *regularization loss* derived from the labeled samples as follows:

$$L_{Reg}(\theta) = -\sum\nolimits_k logW_\theta(y = k|\mathbf{x}) \tag{12}$$

As both the reconstruction and regularization losses are normalized, we define the global optimization objective as:

$$\min_\theta(L_R + \lambda L_{Reg}) \tag{13}$$

where $\lambda \geq 0$ is a hyperparameter that balances the two losses. Importantly, L_{Reg} is included such that the embedding process not only serves to reduce the dimensions – it is actively conditioned to facilitate the encoder in learning class-separable embeddings. In practice, SMATE is not sensitive to λ (see Fig. 3 in Section V-B3); then for all the experiments, we set $\lambda = 1$.

V. Experiments

In this section, we evaluate the performance of the Spatio-temporal representation learned by SMATE. Firstly, we show the experimental setup, including the dataset information, hyperparameters' setting, baseline descriptions, and evaluation metrics. Then we evaluate the performance of the model with different baselines on both supervised and semi-supervised learning tasks. Finally, we analyze the Spatial Modeling Block regarding its ability to model the dimensional interactions in MTS. The model was trained using the Adam optimizer [37] on a single Tesla V100 GPU of 32 Go memory with CUDA 10.2. The authors are devoted to promoting reproducibility. Therefore, the source code, datasets, and instructions are publicly available[2].

A. Experimental setup

We evaluate the learned MTS representation on both classification and semi-supervised classification tasks. As SMATE allows learning class-separable representations, then an SVM classifier is powerful enough [20] to validate the learned representations. For the classification task, we firstly adopt the full labeled training set to learn the class-separable representations, then we train an SVM classifier with radial basis function kernel on the embedding space. For the semi-supervised aspect, we apply different portions of training labels to train the semi-supervised representation model, serving to learn the SVM classifier with the propagated labels from the distance-based class probability.

1) Datasets description: We evaluate our proposed method on the newly released UEA archive [33], including 30 MTS datasets from various application domains[3], which have a big difference in dimension size (2 $\sim$ 963), sample length (8 $\sim$ 3000) and the number of training samples (12 $\sim$ 7494). Readers can find information about the selected datasets sin Table III. We adopt the default train/test split of the archive. All 30 datasets are chosen for supervised analysis, whereas the datasets {*ArticularyWordR., Epilepsy, Heartbeat, SelfRegulationSCP1*} from four different domains are adopted for semi-supervised study.

[2]https://github.com/SMATE2021/SMATE

[3]The datasets can be found in www.timeseriesclassification.com

TABLE III: MTS dataset information

Domain	Dataset	Samples	Dim. (M)	Length (T)	Class
Human Activity	BasicMotions	40/40	6	100	4
	Cricket	108/72	6	1197	12
	Epilepsy	137/138	3	206	4
	ERing	30/270	4	65	6
	Handwriting	150/850	3	152	26
	Libras	180/180	2	45	15
	N/ATOPS	180/180	24	51	6
	RacketSports	151/152	152	30	4
	UWaveGestureLibrary	120/320	6	30	4
Motion	ArticularyWordR.	275/300	9	144	25
	CharacterTrajectories	1422/1436	3	182	20
	EigenWorms	128/131	6	17984	5
	PenDigits	7494/3498	2	8	10
ECG	AtrialFibrillation	15/15	2	640	3
	StandWalkJump	12/15	4	2500	3
EEG/ MEG	FaceDetection	5890/3524	144	62	2
	FingerMovements	316/100	28	50	2
	HandMovementDirection	160/74	10	400	4
	InsectWingbeat	30000/20000	200	30	10
	JapaneseVowels	270/270	12	29	9
	MotorImagery	278/100	64	3000	2
	SelfRegulationSCP1	268/293	6	896	2
	SelfRegulationSCP2	200/180	7	1152	2
Audio Spectra	DuckDuckGeese	50/50	1345	270	5
	Heartbeat	204/205	61	405	2
	Phoneme	3315/3353	11	217	39
	SpokenArabicDigits	6599/2199	13	93	10
Others	EthanolConcent.	261/263	3	1751	4
	LSST	2459/2466	6	36	14
	PEMS-SF	267/173	963	144	7

TABLE IV: Network Architecture of SMATE

Module	Layer	Type
Temporal Channel	1	GRU (128)
	2	GRU (128)
	3	GRU (128)
	4	AveragePooling1D($\mathcal{P}$, $\mathcal{P}$, 0)
Spatial Channel	1	SMB_1 + Conv1D(8,1,0) -128 filters + Batch Norm + ReLU
	2	SMB_2 + Conv1D(5,1,0) -256 filters + Batch Norm + ReLU
	3	SMB_3 + Conv1D(3,1,0) -128 filters + Batch Norm + ReLU
	4	AveragePooling1D($\mathcal{P}$, $\mathcal{P}$, 0)
FC	1	FC (128) + Batch Norm + LeakyReLU
	2	FC (128) + Batch Norm
SMB_1	1	AveragePooling1D(8, 1, 0)
	2	FC (d') + ReLU
	3	FC (M) + Sigmoid
SMB_2	1	AveragePooling1D (5, 1, 0)
	2	FC (8) + ReLU
	3	FC (128) + Sigmoid
SMB_3	1	AveragePooling1D (3, 1, 0)
	2	FC (16) + ReLU
	3	FC (256) + Sigmoid
Decoder	1	UpSampling1D ($\mathcal{P}$)
	2	GRU (128)
	3	GRU (128)
	4	GRU (M)

2) Hyperparameters Setting: **[Network Architecture]** As shown in Table IV, we set 3 GRU layers with a hidden dimension size of 128 for the Temporal Channel. Two 1-D Average Pooling layers are applied for Temporal and Spatial Channels, respectively, where we give the pool size, stride and padding in the bracket. We provide the kernel size, stride, and padding in the brackets of *Conv1D*. The SMBs are configured with consistent parameters of their neighbor *Conv1D*.

However, as the datasets are collected from different domains with a big difference in *M*, *T*, and *N*, it is impractical to apply a unified parameter setting on all datasets. The kernel size m of the three convolution modules is set to (8,5,3), except the *PenDigits* dataset for which the kernels are set to (4,1,1) as it is infeasible to apply our default kernel size "8" into an 8-length time series. The hidden dimension size p' in SMB and pool size $\mathcal{P}$ are set as follows:

$$d' = \begin{cases} M/10, & 100 \leq M \\ M/4, & 10 \leq M < 100 \\ M, & M < 10 \end{cases}, \mathcal{P} = \begin{cases} T/20, & 1000 \leq T \\ T/10, & 50 \leq T < 1000 \\ T/4, & T < 50 \end{cases} \quad (14)$$

[Experiment Parameters] The Adam optimizer is set with the learning rate of 0.00001 and the default exponential decay rate in Keras. As there are limited training samples in most of the UEA datasets, it is not feasible to separate a validation set from the small size of training samples. For instance, the dataset "StandWalkJump" contains only 12 training samples for three classes. It is impractical to split the small training samples into training and validation sets. Therefore, for the datasets with less than 100 training samples, we define the stop condition based on the training loss. For the rest, the validation split is set to 0.2. We set the stop condition to hold when the difference of training/validation loss between epochs is less than a small threshold, 0.0001 for three consecutive steps.

B. Classification Performance Evaluation

We use the accuracy as the default metric for the supervised tasks, which is the default criterion in Time Series Classification work [5]. We also report the number of Win/Ties and the average rank [28] of different methods.

1) Comparison Methods: We compare the performance of a classification task with 13 benchmark approaches, including both classical data mining and recent deep learning methods. We adopt the default parameter settings described in each paper for testing. The methods are summarized as follows:

- Distance-based Nearest Neighbor (1NN) on non-normalized (*non-norm*) or normalized (*norm*) MTS [31]. **1NN-ED** (*non-norm & norm*): Euclidean Distance; **1NN-DTW**$_I$ (*non-norm & norm*): Sum of Dynamic Time Warping (DTW) distance [32] on each variable; **1NN-DTW**$_D$ (*non-norm & norm*): DTW distance applied directly on multi-variate vectors; **1NN-DTW**$_A$ (*norm*): Adaptive distance selected between DTW$_I$ and DTW$_D$ with higher accuracy at run time.
- Bag-of-patterns classifier. **WEASEL+MUSE** [13]: the logistic regression classifier on top of the bag of discriminative features that are extracted from different variables.
- Deep Learning-based classifier. **USRL** [19]: SVM classifier on the representation learned via unsupervised temporal encoding, the Contrastive Learning on Triplet Loss is adopted for adjusting the representation space; **TapNet** [28]: a Softmax function over MTS embeddings, which are learned from a set of variable combinations (i.e., multi-view on the variables); **MLSTM-FCN** [14]: a multi-layer perceptron (MLP) with Softmax function over the concatenated LSTM and CNN layers, capturing the spatial interactions between 1-D series; **CA-SFCN** [25]: Cross Attention Mechanism on both temporal and

spatial axes, working with Fully Convolutional Networks; $\mathbf{SMATE_{NR}}$: SMATE without supervised Regularization, instead, a *Softmax* layer is applied on the embedding.

2) Results Analysis: Table V shows the accuracy results comparison between our proposition and the 13 baselines mentioned above. We show as well the average rank and the number of Wins/Ties of each method. "N/A" indicates the model is not applicable due to memory overflow. Overall, SMATE defends its reliability with 11 Wins/Ties and the highest average rank of 3.85 among all the baselines. The current state-of-the-art deep learning method (TapNet, CA-SFCN) and the powerful data mining method (WEASEL+MUSE) have close ranks (4.73/5.45/4.66). CA-SFCN performs the best on five datasets but is not applicable on seven datasets due to memory overflow. WEASEL+MUSE performs among the best in Human Activity Recognition tasks (*BasicMotions*, *Criket*, *Epilepsy*), as the class-discriminative patterns can be directly extracted from the raw data space. Besides, the unsupervised representation learning method (USRL) performs much worse than SMATE with the same SVM classifier, confirming the reliability of our supervised regularization on the embedding space. Moreover, SMATE achieves the best performance among the baselines on all the datasets of EEG/MEG applications [33] (*FaceDetection*, *FingerMovements*, *HandMovementDirection*, *MotorImagery*, *SelfRegulationSCP1*, *SelfRegulationSCP2*), where the signals (i.e., variables) generally have strong and dynamic dependencies with each other. The spatial dynamic interactions could be essential characteristics that SMATE has successfully captured. $\mathrm{SMATE_{NR}}$ performs much worse than SMATE, proving that the supervised regularization process helps to build the class-separable embeddings.

However, SMATE produces visibly low accuracy on some datasets, e.g., 0.133 on *AtrialFibrillation*, 0.177 on *Phoneme*, on which the baselines perform poorly as well. This is probably caused by the original data source.

3) Parameter effects on model performance: We analyse the effects of window/kernel size m, pool size $\mathcal{P}$ and regularization weight λ on the example dataset *SelfRegulationSCP1*. In Fig. 3, we observe that a larger m brings higher model accuracy, as it creates a larger receptive field captured by both SMB and *Conv1D* block. Furthermore, the early convolutional modules are more sensitive to m as they keep close to the raw input features. A larger pool size $\mathcal{P}$ will reduce the embedding size, thus affecting the preserved embedding features and the training efficiency. SMATE is not sensitive to λ. It can be explained by the fact that a stable reconstruction process makes the model focus more on the regularization of the embedding space. When λ=0, no regularization is applied, leading to sparsely distributed embeddings with poor prediction performance.

C. Semi-supervised Classification Performance

For semi-supervised tasks, we evaluate the classifier's accuracy at different supervision levels by varying the labeled samples in the training set. We select the datasets {*ArticularyWordR., Epilepsy, Heartbeat, SelfRegulationSCP1*}

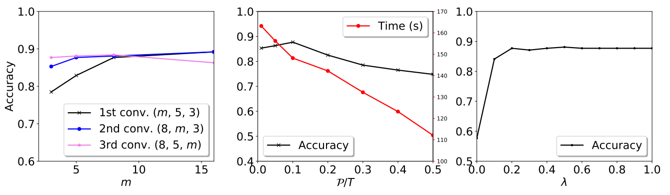


Fig. 3: Parameter effects: *left*) the kernel/window size m at each convolutional module; *mid*) The pool size $\mathcal{P}$ for building the embedding; *right*) the hyperparameter λ which weights the regularization loss.

from 4 different application domains. For comparison, we applied one classic model **1NN-DTW-D** [3] and three recently proposed semi-supervised deep learning models: **USRL** [19], **Semi-TapNet** [28] and **MTL** [30]. Since **1NN-DTW-D** and **MTL** are initially designed for UTS, we adapt them by:

- Adopting DTW_D[4] [31] as distance in **1NN-DTW-D**.
- Updating the MTS network optimization metrics in **MTL**.

741Figure 4 shows the classification accuracy at different supervision levels. In a Motion Recognition task (*ArticularyWorkR.*), from 10% labeled training set to fully labeled one, the accuracy of SMATE varies only by 0.046, compared to INN-DTW-D (0.264), USRL (0.286), Semi-TapNet (0.151) and MTL(0.225), showing that SMATE is capable of learning a class-separable representation under weak supervision and gives a better prediction than other classifiers under intense supervision. This conclusion is also demonstrated in EEG/MEG applications (*SelfRegulationSCP1*), with 10% labeled samples, SMATE is capable of obtaining a higher accuracy (0.781) than fully supervised 1NN-DTW-D (0.775), USRL (0.771), Semi-TapNet (0.739) and MTL (0.730). In Human Activity domain (*Epilepsy*), Semi-TapNet performs the worst with low supervised ratios, which can be explained by the fact that the limited labels restrain the intermediate-trained TapNet classifier for predicting the pseudo-labels. In an Audio Spectra task (*Heartbeat*), though the fully supervised accuracy of SMATE (0.741) is not as good as Semi-TapNet (0.751), the weakly supervised SMATE with 10% labeled samples performs the best among all semi-supervised models, indicating the reliability of the semi-supervised representation learned by SMATE.

D. Visualization & Interpretation of the Representation Space

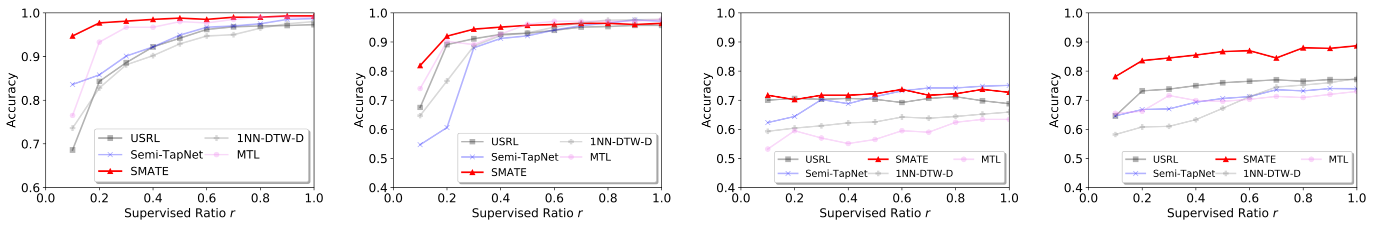
Apart from the thorough exploration of the weakly labeled samples, the representation space learned via SMATE shows good interpretability compared to the traditional Deep Learning models [14], [15], [24], [25] where the intermediate representations are not intuitively explainable.

We show in Figure 5 that t-SNE visualization of the representation space for the *Epilepsy* dataset, which contains four human activities: *Walking while gesturing*, *Walking slowly*, *Walking fast*, *Walking normally*. The 137 samples in the training set are projected to the representation space with 10%

[4]*DTW-D* and DTW_D are two different distance measures designed respectively for Univariate and Multivariate Time Series

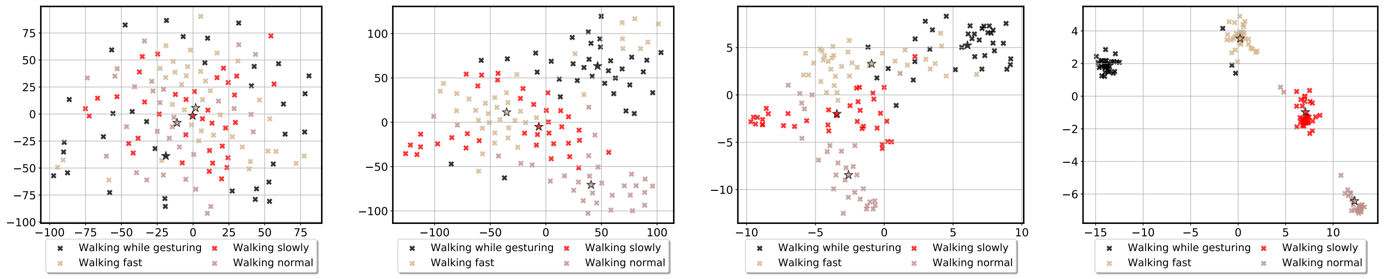
TABLE V: Performance Comparison for MTS classification over UEA MTS archive

Dataset	SMATE	SMATE$_{NR}$	USRL	TapNet	MLSTM -FCN	CA-SFCN	WEASEL +MUSE	1NN-ED	1NN-DTW$_I$	1NN-DTW$_D$	1NN-ED (*norm*)	1NN-DTW$_I$ (*norm*)	1NN-DTW$_D$ (*norm*)	1NN-DTW$_A$ (*norm*)
ArticularyWordR.	**0.993**	0.987	0.987	0.987	0.973	0.97	0.99	0.97	0.98	0.987	0.97	0.98	0.987	0.987
AtrialFibrillation	0.133	0.133	0.133	**0.333**	0.267	**0.333**	**0.333**	0.267	0.267	0.2	0.267	0.267	0.22	0.267
BasicMotions	**1**	**1**	**1**	**1**	0.95	**1**	**1**	0.675	**1**	0.975	0.676	**1**	0.975	**1**
CharacterTrajectories	0.984	**0.997**	0.994	**0.997**	0.985	0.988	0.99	0.964	0.969	0.99	0.964	0.969	0.989	0.989
Cricket	0.986	0.968	0.986	0.958	0.917	0.972	**1**	0.944	0.986	**1**	0.944	0.986	**1**	**1**
DuckDuckGeese	N/A	N/A	**0.675**	0.575	**0.675**	N/A	0.575	0.275	0.55	0.6	0.275	0.55	0.6	0.567
EigenWorms	N/A	N/A	0.878	0.489	0.504	N/A	**0.89**	0.55	0.603	0.618	0.549	N/A	0.619	N/A
Epilepsy	0.964	0.946	0.957	0.971	0.761	0.986	**1**	0.667	0.978	0.964	0.666	0.978	0.964	0.979
ERing	**0.981**	0.904	0.88	0.904	0.941	0.856	0.964	0.93	0.93	0.93	0.93	0.93	0.93	0.93
EthanolConcentration	0.399	0.373	0.236	0.323	0.373	0.323	**0.43**	0.293	0.304	0.323	0.293	N/A	0.323	0.316
FaceDetection	**0.647**	0.556	0.528	0.556	0.545	N/A	0.545	0.519	0.513	0.529	0.519	0.5	0.529	0.529
FingerMovements	**0.62**	0.55	0.54	0.53	0.58	0.59	0.49	0.55	0.52	0.53	0.55	0.52	0.53	0.509
HandMovementD.	**0.554**	0.365	0.27	0.378	0.365	0.324	0.365	0.279	0.306	0.231	0.278	0.306	0.231	0.224
Handwriting	0.421	0.335	0.533	0.357	0.286	0.322	0.605	0.371	0.509	**0.607**	0.2	0.316	0.286	0.601
Heartbeat	0.741	0.615	0.737	0.751	0.663	**0.756**	0.727	0.62	0.659	0.717	0.619	0.658	0.717	0.571
InsectWingbeat	N/A	N/A	0.16	**0.208**	0.167	N/A	N/A	0.128	N/A	0.115	0.128	N/A	N/A	N/A
JapaneseVowels	0.965	0.924	**0.989**	0.965	0.976	0.973	0.973	0.924	0.959	0.949	0.924	0.959	0.949	0.959
Libras	0.849	0.834	0.867	0.85	0.856	0.89	0.878	0.833	**0.894**	0.872	0.833	**0.894**	0.87	0.879
LSST	0.582	0.568	0.558	0.568	0.373	**0.674**	0.59	0.456	0.575	0.551	0.456	0.575	0.551	0.551
MotorImagery	**0.59**	**0.59**	0.54	**0.59**	0.51	N/A	0.51	0.39	N/A	0.5	0.51	N/A	0.5	0.5
N/ATOPS	0.922	0.87	0.944	0.939	0.889	**0.956**	0.87	0.86	0.85	0.883	0.85	0.85	0.883	0.883
PEMS-SF	**0.803**	0.744	0.688	0.751	0.699	N/A	N/A	0.705	0.734	0.711	0.705	0.734	0.711	0.73
PenDigits	0.98	0.98	**0.983**	0.98	0.978	0.975	0.948	0.973	0.939	0.977	0.973	0.939	0.977	0.977
Phoneme	0.177	0.19	**0.246**	0.175	0.11	0.19	0.19	0.104	0.151	0.151	0.104	0.151	0.151	0.151
RacketSports	0.849	0.816	0.862	0.868	0.803	0.875	**0.934**	0.868	0.842	0.803	0.868	0.842	0.803	0.858
SelfRegulationSCP1	**0.887**	0.874	0.771	0.739	0.874	0.734	0.71	0.771	0.765	0.775	0.771	0.765	0.775	0.786
SelfRegulationSCP2	**0.567**	0.539	**0.556**	0.55	0.472	N/A	0.46	0.483	0.533	0.539	0.483	0.533	0.539	0.539
SpokenArabicDigits	0.979	0.967	0.956	0.983	**0.99**	0.982	0.982	0.967	0.96	0.963	0.967	0.959	0.963	0.963
StandWalkJump	**0.533**	0.4	0.4	0.4	0.067	0.2	0.333	0.2	0.333	0.2	0.2	0.333	0.2	0.333
UWaveGestureLibrary	0.897	0.869	0.884	0.894	0.891	0.8	**0.916**	0.881	0.868	0.903	0.81	0.868	0.903	0.9
Avg. Rank	3.85	6.19	5.9	4.73	7.33	5.45	4.66	9.3	7.43	6.37	9.37	7.88	6.83	6.21
Wins (Ties)	**11**	**3**	6	5	2	5	8	0	2	2	0	2	1	2



(a) ArticularyWordR. (Motion) **(b)** Epilepsy (Human Activity) **(c)** Heartbeat (Audio Spectra) **(d)** SelfRegulationSCP1 (EEG/MEG)

Fig. 4: Semi-supervised performance comparison on datasets from different domains



(a) Without any regularization **(b)** Regularization step 1: supervised initialization of the class centroids **(c)** Regularization step 2: supervised adjustment of the class centroids **(d)** Regularization step 3: unsupervised adjustment of the class centroids

Fig. 5: The g visualization of the representation space for the *Epilepsy* dataset. We set the supervised ratio to 0.1. The colors of the embeddings represent their inherent labels, which are not fully adopted for training. The class centroids are marked by $\star$. (a) The representation space obtained only by the reconstruction objective without any regularization. (b) Regularization step 1: 10% labeled samples in the training set are adopted to initialize the class centroids. (c) Regularization step 2: the same labeled samples are deployed to adjust the class centroids. (d) Regularization step 3: the rest unlabeled samples are deployed to adjust the class centroids.

labels adopted for training. We show respectively the space visualization at each regularization step. The results suggest that: **1)** the embeddings obtained only with the autoencoder's reconstruction objective are sparsely distributed, as the auto-encoder focuses more on the overall restoration from the embeddings, but less on the inner difference between the class-specific samples; **2)** with the regularization step 1, the class centroids initialized by the 10% labeled samples tend to assemble the embeddings with the same inherent labels. The assembling ability is further enhanced by the supervised adjustment in regularization step 2. **3)** the unlabeled samples are thoroughly explored in regularization step 3 to foster the class-specific clusters, which allow building a simple but reliable classifier such as SVM. **4)** the representation space is interpretable for not only the effect of the weak supervision but also the classification results. For instance, in Figure 5d, three samples of *Walking while gesturing* stay close to the class centroid of *Walking fast*, which may lead to the misclassification of certain samples in the two classes. To improve the classifier, more labels of the two classes in the training set can be added.

E. Performance of Spatial Modeling Block (SMB)

To validate the Spatial Modeling Block (SMB), firstly, we compare the classification accuracy of SMATE with or without integrating SMB on the 27 datasets that SMATE has successfully executed. Then we rebuild SMATE by re-placing SMB with the following modules in the state-of-the-art work which learn the variable relationships of MTS: **Random Dimension Permutation (RDP)** in TapNet [28] and **Squeez-and-Excitation (SE)** in MLSTM-FCN [14]. Briefly, SMATE-SMB achieves [17 Wins|8 Ties|2 Losses] to SMATE-NonSMB, indicating that SMB contributes to a better MTS representation.

In Figure 6, we give a one-to-one comparison between SMB and SE/RDP on the 27 datasets. We find that SMATE performs better than other modules on modeling the spatial interactions: [14 Wins|8 Ties|5 Losses] to SE, [12 Wins|9 Ties|6 Losses] to RDP. RDP performs relatively better than SE, as a set of grouped variables produced by RDP provides various MTS views, allowing exploring the interactions between the subsets of all variables more thoroughly. However, extra parameters for variable groups are introduced. SE is a parameter-free module but considers each variable has a unique and stable state when interacting with others, which ignores the dynamic features in time series. SMB answers both the questions of the parameter-free settings and the dynamic interactions. The results show that capturing the spatial dynamic interactions at the sub-sequence level performs better than modeling the variable interactions at the sequence level [14], [28].

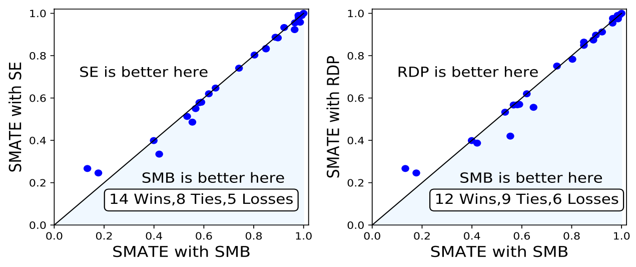


Fig. 6: Accuracy performance comparison between *left*) SMB & SE, *right*) SMB & RDP, which are are integrated separately into SMATE.

F. Efficiency Analysis

As the classical data mining classifiers do not benefit from the GPU's acceleration, it is unfair to compare them with the deep learning models on different hardware. Here, we compare SMATE with the deep learning models {MLSTM-FCN [14], CA-SFCN [25], USRL [19], and TapNet [28]}. Table VI shows the models' parameter numbers on *ArticularyWordR.*. Nevertheless, they do not represent the models' efficiency.

TABLE VI: Parameter numbers of deep learning models on MTSC

ArticularyWordR.	SMATE	MLSTM-FCN	CA-SFCN	USRL	TapNet
Param. number	564585	597865	685965	368655	635285

Figure 7 shows the computational efficiency concerning the different factors: (a) **the number of training epochs**, we take the dataset *ArticularyWordRecognition* with (N_{train}, M, T) = (275, 9, 144); (b) **the TS length**, we select *EthanolConcentration* with (N_{train}, M, **T**) = (261, 3, **1751**), and do random re-samplings over raw sequences; (c) **the number of TS instance** in the training set, we choose *LSST* with ($\mathbf{N_{train}}$, M, T) = (**2459**, 6, 36) and randomly re-sample the TS instances; (d) **the number of variables**, we select *PEMS-SF* with (N_{train}, **M**, T) = (267, **963**, 144) and randomly re-sample the TS variables. With the models' default parameter settings mentioned in their papers, we set 2000 training steps for USRL, 3000 training epochs for others during the tests on the factors (b)(c)(d).

Figure 7 shows that the training time of the deep learning models tends to be linear in the four factors. More specifically, the results suggest that: **1)** SMATE is generally much more efficient on short TS (with $T <\sim 500$), and more costly than MLSTM-FCN on long TS (see Figure 7b). As an MTS instance $x \in \mathcal{R}^{T\times M}$ with larger T brings a sequence embedding $h \in \mathcal{R}^{L\times D}$ with a larger L in SMATE, the regularization on a lager embedding space becomes more costly. **2)** SMATE is generally more efficient than the competitors, but tends to be more sensitive to the variable numbers **M** than USRL and TapNet (see Figure 7d). To explain this, first, Triplet Loss adopted in USRL requires intensive distance computations between the embeddings, which offsets the effect of larger input space. This can be also demonstrated in Figure 7b & 7c, where USRL is not sensitive to the TS length **T**, but highly sensitive to the number of TS instances $\mathbf{N_{train}}$. Second, the efficiency of SMATE is greatly affected by the input and output space of the auto-encoder, which is more sensitive to the variable numbers than the Random Dimension Permutation (RDP) block in TapNet, which works only on the input space.

Overall, USRL [19] is an order slower than SMATE due to its mini-batch optimization strategy and huge distance computations required by Triplet Loss. CA-SFCN [25] performs less than SMATE because of its costly cross attention mechanism on both temporal and spatial axes. However, MLSTM-FCN [14] and TapNet [28] outperform SMATE for long TS length ($\mathbf{T} > 500$) or huge variable numbers ($\mathbf{M} > 300$).

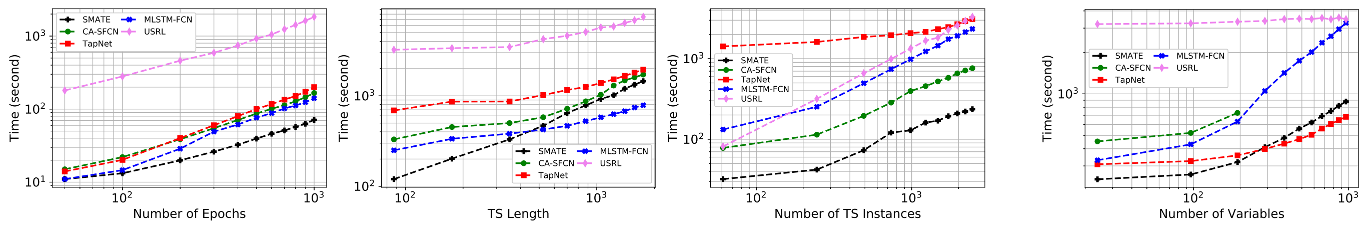
G. Discussion

Our approach has several advantages. First, owing to the Spatio-Temporal dynamic encoder, SMATE allows exploring more thoroughly the essential characteristics of MTS. TapNet [28] and MLSTM-FCN [14] generally consider the correlation between the entire 1-D series, while USRL [19] processes indifferently the MTS and UTS, they all ignore the fact that the interactions between 1-D segments may evolve in the dynamic sequence, which is especially important in certain domains (e.g., EEG/MEG applications).

Second, SMATE explores thoroughly the unlabeled samples, which contributes not only to the autoencoder's reconstruction objective, but also to the regularization process on the embedding space. While Semi-TapNet [28] considers unlabeled data only with the pseudo-labels predicted by intermediate-trained classifiers, which is less reliable when there are limited labels. USRL [19] trains the representation without any supervision, which shows less advantage for the classification task.

Third, the representation space learned via SMATE is interpretable for showing the effect of the three-step regularization process and explaining the classification results, which allows taking further actions to improve the classifier.

Finally, SMATE allows an efficient representation learning and classification for MTS. On the one hand, from the distance-based approaches (e.g. 1NN-based classifiers [31]) to the bag-of-patterns classifier (e.g., WEASEL+MUSE [13]),



(a) *ArticularyWordRecognition*: (N_{train},M,T) = (275,9,144) **(b)** *EthanolConcentration*: (N_{train},M,$\mathbf{T}$) = (261,3,**1751**) **(c)** *LSST*: ($\mathbf{N_{train}}$,M,T) = (**2459**,6,36) **(d)** *PEMS-SF*: (N_{train},$\mathbf{M}$,T) = (267,**963**,144)

Fig. 7: Training time on four datasets with regard to: (a) the number of training epochs; (b) the TS length $\mathbf{T}$; (c) the number of TS instance $\mathbf{N_{train}}$; (d) the number of variables $\mathbf{M}$. We did not report the results of CA-SFCN [25] for $M > 200$ in (d) due to the memory overflow.

the classic data mining methods always show a high time complexity [5]. On the other hand, the deep learning-based approaches show no significant efficiency advantage to SMATE due to their larger parameter space to optimize.

VI. Conclusion

In this paper, we proposed SMATE, to learn the Spatial-temporal representation on weakly-labeled multivariate time series. Inside the autoencoder-based structure, the Spatial-temporal encoder maps the temporal dynamic features and the spatial dynamic interactions into a low dimensional embedding space. A semi-supervised three-step regularization process is proposed to compel the model in learning class-separable representation. The weak supervision on the embedding space allows building a reliable classifier, which is extremely valuable in real-life scenarios with label shortage issues. The results show that the evolving variable interactions (i.e., *spatial dynamics*) play an essential role in modeling multivariate time series. Moreover, SMATE allows for visual interpretability in both the learned representation and the semi-supervised representation learning process.

A recent experimental study on MTCS models has been conducted in [5]. It concludes that MTSC is still *at an earlier stage of development than univariate TSC*. For instance, existing approaches do not consider typical features of real-world data, such as missing values and unequal length time series. Hence, our future work will be oriented towards extending SMATE to support multivariate time series with missing values and unequal length. Further, we intend to improve the model by, e.g., incorporating insights from CoDATS [38] with time series domain adaptation, and from ROCKET [39] with random convolutional kernels for feature extraction.

Acknowledgements

This research was supported by DATAIA convergence institute as part of the *Programme d'Investissement d'Avenir* (ANR-17-CONV-0003) operated by DAVID Lab, UVSQ, Université Paris-Saclay. The authors would like to thank Anthony Bagnall and his team for providing the community with valuable datasets and source codes in the UEA & UCR Time Series Classification Repository. The authors would also like to thank Eamonn Keogh for his careful review and remarks on the preliminary version of this paper, as well as Nicoleta Preda and Zaineb Chelly for their suggestions.

References

[1] L. Wei and E. Keogh, "Semi-supervised time series classification," in *Proc. ACM SIGKDD'06*.

[2] M. N. Nguyen, X. L. Li, and S. K. Ng, "Positive unlabeled learning for time series classification," in *IJCAI*, 2011.

[3] Y. Chen, B. Hu, E. Keogh, and G. E. A. P. A. Batista, "DTW-D: Time Series Semi-Supervised Learning from a Single Example," in *KDD'13*.

[4] C. A. Ratanamahatana and D. Wanichsan, "Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification," *Soft. Eng., Arti. Intel., Net. & Para./Distri. Comp.*, 2008.

[5] R. A. Pasos, F. Michael, L. James, M. Matthew, and B. Anthony, "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances." *DMKD*, 2020.

[6] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *IJCNN*, 2017.

[7] J. Lines, S. Taylor, and A. Bagnall, "HIVE-COTE: The Hierarchical Vote Collective of Transformation-based Ensembles for Time Series Classification," in *IEEE ICDM*, 2016.

[8] M. S. Cetin, A. Mueen, and V. D. Calhoun, "Shapelet ensemble for multi-dimensional time series," in *SDM*, 2015.

[9] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Fast classification of univariate and multivariate time series through shapelet discovery," *Knowledge and Information Systems*, vol. 49, no. 2, pp. 429–454, 2016.

[10] R. Mousheimish, Y. Taher, and K. Zeitouni, "Automatic Learning of Predictive CEP Rules: Bridging the Gap between Data Mining and Complex Event Processing," in *DEBS*, 2017.

[11] A. Dorle, F. Li, W. Song, and S. Li, "Learning Discriminative Virtual Sequences for Time Series Classification," in *CIKM*, 2020.

[12] M. Gokce Baydogan, G. Runger, M. G. Baydogan, and G. Runger, "Learning a symbolic representation for multivariate time series classification," *Data Min Knowl Disc*, vol. 29, pp. 400–422, 2015.

[13] P. Schäfer and U. Leser, "Multivariate Time Series Classification with WEASEL+MUSE," Tech. Rep., 2017.

[14] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, 2019.

[15] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *WAIM'14*.

[16] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," *Sci. Rep.'18*.

[17] Y. Bai, L. Wang, Z. Tao, S. Li, and Y. Fu, "Correlative Channel-Aware Fusion for Multi-View Time Series Classification," in *AAAI'21*.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE TPAMI'13*, pp. 1798–1828.

[19] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised Scalable Representation Learning for Multivariate Time Series," in *NeurIPS*, 2019.

[20] L. Wu, I. En-Hsu, Y. J. Yi, F. Xu, Q. Lei, and M. J. Witbrock, "Random Warping Series: A Random Features Method for Time-Series Embedding," in *AISTATS*, 2018.

[21] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *DMKD*, 2013.

[22] S. Li, Y. Li, and Y. Fu, "Multi-View Time Series Classification: A Discriminative Bilinear Projection Approach," in *CIKM*, 2016.

[23] P. Schäfer and U. Leser, “Fast and Accurate Time Series Classification with WEASEL,” in *CIKM*, 2017.

[24] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, “Deep convolutional neural networks on multichannel time series for human activity recognition,” in *IJCAI*, 2015.

[25] Y. Hao and H. Cao, “A New Attention Mechanism to Classify Multivariate Time Series,” in *IJCAI*, 2020.

[26] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks,” in *KDD*, 2020.

[27] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting,” in *ICLR*, 2018.

[28] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, “TapNet: Multivariate Time Series Classification with Attentional Prototypical Network,” in *AAAI’20*.

[29] H. Wang, Q. Zhang, J. Wu, S. Pan, and Y. Chen, “Time series feature learning with labeled and unlabeled data,” *Pattern Recognition*, 2019.

[30] S. Jawed, J. Grabocka, and L. Schmidt-Thieme, “Self-Supervised Learning for Semi-Supervised Time Series Classification,” in *PAKDD’20*.

[31] M. Shokoohi-Yekta, J. Wang, and E. Keogh, “On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case,” in *SDM*, 2015.

[32] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, “Generalizing DTW to the multi-dimensional case requires an adaptive approach HHS Public Access,” *Data Min Knowl Discov*, vol. 31, no. 1, pp. 1–31, 2017.

[33] A. B. Hoang, A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, “The UEA multivariate time series classification archive,” Tech. Rep., 2018.

[34] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, “Learning Representations for Time Series Clustering,” in *NeurIPS*, 2019.

[35] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *ICML’15*.

[36] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” in *ICML*, 2010.

[37] D. P. Kingma and J. Lei Ba, “Adam: A Method for Stochastic Optimization,” in *ICLR*, 2015.

[38] G. Wilson, J. R. Doppa, and D. Cook, “Multi-source deep domain adaptation with weak supervision for time-series sensor data,” in *KDD’20*.

[39] A. Dempster, F. Petitjean, and G. I. Webb, “ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels,” *DMKD*, 2020.