Learning Meta-Distance for Sequences by Learning a Ground Metric via Virtual Sequence Regression

Bing Su, and Ying Wu, *Fellow, IEEE*

Abstract—Distance between sequences is structural by nature because it needs to establish the temporal alignments among the temporally correlated vectors in sequences with varying lengths. Generally, distances for sequences heavily depend on the ground metric between the vectors in sequences to infer the alignments and hence can be viewed as meta-distances upon the ground metric. Learning such meta-distance from multi-dimensional sequences is appealing but challenging. We propose to learn the meta-distance through learning a ground metric for the vectors in sequences. The learning samples are sequences of vectors for which how the ground metric between vectors induces the meta-distance is given. The objective is that the meta-distance induced by the learned ground metric produces large values for sequences from different classes and small values for those from the same class. We formulate the ground metric as a parameter of the meta-distance and regress each sequence to an associated pre-generated virtual sequence w.r.t. the meta-distance, where the virtual sequences for sequences of different classes are well-separated. We develop general iterative solutions to learn both the Mahalanobis metric and the deep metric induced by a neural network for any ground-metric-based sequence distance. Experiments on several sequence datasets demonstrate the effectiveness and efficiency of the proposed methods.

Index Terms—Metric learning, temporal alignment, virtual sequence regression, optimal transport.

1 Introduction

In many domains, the data are naturally in the form of multi-dimensional sequences. Pairwise distance measures between sequences serve as a proxy to manipulate the structured sequences so that any metric-based machine learning methods can be directly applied. The performances of metric-based algorithms such as the k-nearest neighbor classifier (k-NN) heavily depend on the quality of the distance measures. Therefore, learning distances for sequences from data is especially appealing.

Although metric learning has achieved a considerable maturity level both in practice and in theory [1], propagating these advances to sequence data is not trivial. This is because most existing metric learning methods are developed for static data which are in the form of "flat" feature vectors. An acquiescent assumption is that these vector data are independent and identically distributed, but the elements in sequences exhibit temporal relationships. Much less work has been devoted to metric learning for sequence data, and most of them actually encode each sequence into a vector and simply build the metric upon the vectors, which cannot capture the alignments or relationships among the vectors in sequences explicitly and may lose significant temporal information. Learning distances that operate directly on sequences is challenging, because such distances are naturally structural and combinatorial. Specifically, the major difficulties lie in two aspects.

First, different sequences vary in length, evolution speed, and local temporal duration. Different distance measures for sequences such as [2], [3] perform temporal alignments to eliminate the local temporal discrepancies. An illustrative alignment is shown in Fig. 1. Inferring the alignment depends on the metric between elements in sequences. For a specific sequence pair, their alignment cannot be inferred before the underlying metric is learned. Therefore, the objective of learning distances for sequences generally involves latent alignment structures when formulating the distances as a function of the unknown metric, and hence is difficult to manipulate and optimize.

Second, most metric learning methods employ the must-link/cannot-link constraints over positive/negative pairs [4], [5] or the relative constraints over triplets [6], [7]. The number of constraints is quadratic or cubic in the number of training samples, which easily becomes intractable when more training samples are available. One heuristic is to mine only a subset of the most informative constraints, but such mining is not trivial. Because of the complexity of measuring distances for sequences, the cost of constructing these constraints is larger and it can be computationally prohibitive to update the subset of constraints with the update of the metric during the optimization. Reducing the number of constraints is more crucial for sequence data.

In this paper, we propose a metric learning framework for sequence data to tackle these issues. We unify a wide range of distance measures for sequences into a formulation as a function of the *ground metric* for elements in sequences.

- *B. Su is with the Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China. Part of the work was done when B. Su was at Institute of Software, Chinese Academy of Sciences. E-mail: subingats@gmail.com.*
- *Y. Wu is with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, 60208. E-mail: yingwu@ece.northwestern.edu.*

Manuscript received April 19, 2005; revised August 26, 2015.

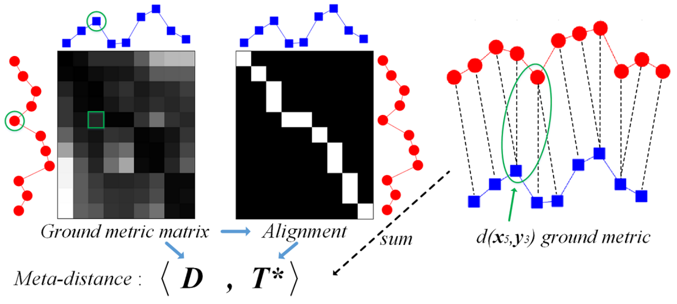


Fig. 1. For a sequence of 11 red points and a sequence of 9 blue points, given a ground metric d between the points, a ground metric matrix $\boldsymbol{D}$ stores all the pairwise distances between points with d, e.g., $\boldsymbol{D}_{5,3} = d(\boldsymbol{x}_5, \boldsymbol{y}_3)$ is the distance between the fifth red point and the third blue point with the ground metric. The optimal alignment matrix $\boldsymbol{T}^*$ can be inferred based on $\boldsymbol{D}$ according to some temporal constraints which differ in different distance measures. Each element of $\boldsymbol{T}^*$ indicates whether or the probability of aligning the corresponding two points. e.g., $\boldsymbol{T}^*_{5,3} = 0$ means that the fifth red point and the third blue point are not aligned. The distance between the two sequences equals $\langle \boldsymbol{T}, \boldsymbol{D} \rangle$ and hence depends on the ground metric. It can be viewed as a meta-distance upon the ground metric.

As shown in Fig. 1, the final distances are *meta-distances* built upon the ground metric by inferring the temporal alignments among the element pairs. Thanks to such parameterization, we show that various meta-distances for sequences are amenable to learn via learning a Mahalanobis distance [8] or a deep embedding function implemented by a neural network as the ground metric. More specifically, we treat the alignments as latent variables of the meta-distance function that takes the ground metric as an argument, since inferring them also depends on the ground metric. The formulation of the objective for learning the ground metric incorporates latent variables. We develop iterative alternating descent algorithms that achieve joint optimization of the Mahalanobis or deep metric and the latent alignments, which can be instantiated with any meta-distances using various alignment inference methods.

Another contribution of our work is the extension of the *regressive virtual metric learning (RVML)* [9] method for reducing the number of constraints. RVML requires a linear number of constraints by moving each sample to its corresponding pre-defined virtual point. Our method extends RVML in four ways: (1) RVML learns a metric for independent vector data. Our method learns meta-distances for sequence data by learning a ground metric for non-independent vectors in sequences. (2) RVML associates each sample with a virtual vector. Our method associates each sequence sample with a virtual sequence and provides three solutions to generate virtual sequences. (3) RVML is not combined with deep learning. Our method is extended to learn a non-linear deep metric as the ground metric. (4) RVML does not involve latent variables. Our method learns the ground metric and the latent alignments simultaneously.

This paper is an extension of the conference paper [10]. The major extensions include (1) the proposed method is extended to learn deep ground metrics by employing neural networks and experimentally compared with seven deep metric learning methods; (2) more virtual sequence generation methods are presented and evaluated; (3) the proposed method is extended to tackle zero-shot sequence classification; (4) More detailed discussions, illustrations, and analysis are presented.

2 Related Work

Differences with conventional metric learning. Most classical metric learning methods for vector data employ either the pair-based or the triplet-based constraints. The pair-based must-link/cannot-link side information was introduced in the seminal work of [4], and then widely used in a lot of methods such as *information-theoretic metric learning (ITML)* [5], regularized distance metric learning [11], and sparse distance metric learning [12]. Generally, the nearest neighbors based methods, such as neighbourhood component analysis [13], maximally collapsing metric learning [14], *large margin nearest neighbors (LMNN)* [7], [15], and *sparse compositional metric learning (SCML)* [16], used the triple-based constraints to force the distances of each instance to its target neighbors relatively smaller than those to impostors. RVML [9] introduced the virtual point based constraints. Propagating these advances for vector representations to sequence data is not trivial.

Differences with edit distance learning and kernel learning for sequences. In [17], [18], [19], the string edit distance was learned by learning the cost matrix for edit operations. The elements in sequences were symbols from a fixed finite alphabet and the edit operations for each sequence pair were fixed. In [20], weighted finite-state transducers based rational kernels [21] were learned to measure the similarities between sequences, where the elements were also restricted to a finite alphabet. It is difficult to apply these methods to unconstrained sequences, where the elements are continuous real vectors rather than discrete symbols and the number of all possible elements is infinite. In contrast, our method learns the Mahalanobis distance for real vectors and the latent alignments jointly.

Differences with existing metric learning methods for optimal transport (OT). In [22], the OT distance for histograms was learned by learning the ground metric based on side supervision on specific similarity coefficients of all histogram pairs, where the supporting points for all histograms were fixed. This method cannot be applied to unconstrained sequences because it directly learns a ground matrix containing all pairwise distances for the supporting points. In [23], the supervised word mover's distance (SWMD) learned OT distances for documents each consists of a set of unordered words by learning the ground metric, where the words are in a fixed finite dictionary and the weights for these fixed words were learned together. It minimized the leave-one-out kNN error by a gradient-based solution. In contrast, our method minimizes the regression-based loss by non-gradient descent optimization, and is applicable to unconstrained multidimensional sequences where the elements lie in a continuous space.

Differences with existing metric learning methods for sequences. Canonical Time Warping (CTW) [24], Generalized CTW [25], [26], and Deep CTW [27], [28] are unsupervised distances that map two sequences with two different transformations, respectively. The transformations are different for different sequence pairs. In contrast, our method is supervised and learns a common ground metric for all sequences. Temporal Transformer Network (TTN) [29] takes a sequence with a pre-defined length as input and predicts its warping function which is fixed when comparing with different sequences. In contrast, our method can handle sequences of different lengths. For different sequence pairs, the warping functions or alignments inferred by the meta-distance are different.

In [30], the ground-truth alignments were used for learning the metric. In contrast, ground-truth alignments are not available and our method learns the ground metric and the alignments jointly. In [31] and [32], Mahalanobis distances were learned as ground metrics to enhance the *dynamic time warping (DTW)* distance, where the DTW alignments for all sequence pairs were fixed by using the Euclidean metric. The solutions were sub-optimal since the alignments may change with the learned matrices. In contrast, our method achieves joint optimization for the metric and the latent alignments. In [33], LDMLT iteratively updated the ground Mahalanobis metric with the triplets constraints and updated the alignments by DTW to build dynamic triplets. However, the iterative solution is not guaranteed to converge because updating the alignments by DTW does not guarantee to decrease the objective of the logDet divergence based metric learning. In contrast, our method is guaranteed to converge, trains much faster, and is applicable to different sequence distances.

Differences with deep metric learning. Deep metric learning methods [34], [35], [36] are typically deep extensions of classical metric learning methods. Most of them also employ the pair-based or the triplet-based constraints and formulate the loss functions in terms of pairwise distances between embedding representations. Such loss functions include neighbourhood component analysis (NCA) loss [37], contrastive loss [38], triplet loss [39], hierarchical triplet loss [40], binomial deviance loss [34], etc. All possible pairs or triplets grow polynomially with the number of training samples. Random sampling is often less informative since the training may be dominated by redundant pairs or triplets. A lot of recent works focus on sampling, constructing, or weighting more informative pairs or triplets, such as lifted structured loss [35], N-pairs loss [41], semi-hard mining [42], and general pair weighting (GPW) [43]. These methods also assume that the training samples are independent and applying them to features in sequences can lose significant temporal information. In contrast, our method utilizes the temporal structures of sequences and the number of constraints grows only linearly with the number of training sequences.

Differences with recurrent neural network (RNN) based metric learning. Some works [44], [45] actually encoded the sequences into fixed-length vectors and build metrics upon vectors. In contrary, our method is applied to elements in sequences and the alignments can be explicitly inferred, which are crucial in some applications. Through the alignments, our method enables a fine and intuitive interpretation of the meta-distance. Moreover, our method can be applied before sequences are fed into those RNN-based methods to enhance the temporal relationships and the discriminative information.

3 A Unified Perspective on Distance Measures for Sequences

In this section, we present a unified formulation of the distance measures for sequences and establish the connections between the formulation and several distance measures.

Let Ω be a space and $d(\boldsymbol{M}) : \Omega \times \Omega \rightarrow \mathbb{R}$ be the metric on this space, which is parameterized by $\boldsymbol{M}$. Given two sequences $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{L_X}] \in \Omega^{L_X}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{L_Y}] \in \Omega^{L_Y}$ with lengths L_X and L_Y, respectively, whose elements $\boldsymbol{x}_i, i = 1, \cdots, L_X$ and $\boldsymbol{y}_j, j = 1, \cdots, L_Y$ are sampled in Ω, the distance between them can be formulated as

$$g_{\boldsymbol{M}}(\boldsymbol{X}, \boldsymbol{Y}) = \langle \boldsymbol{T}^*, \boldsymbol{D}(\boldsymbol{M}) \rangle , \tag{1}$$

where $\langle \boldsymbol{T}, \boldsymbol{D} \rangle = tr(\boldsymbol{T}^T \boldsymbol{D})$ is the Frobenius dot product. An illustrative example is shown in Fig. 1, where $L_X = 11$, $L_Y = 9$, and all the red and blue points lie in Ω.

$$\boldsymbol{D}(\boldsymbol{M}) := [d(\boldsymbol{M}, \boldsymbol{x}_i, \boldsymbol{y}_j)]_{ij} \in \mathbb{R}^{L_X \times L_Y} \tag{2}$$

is the cost matrix of all pairwise vector-wise distances between elements in $\boldsymbol{X}$ and $\boldsymbol{Y}$, whose element $\boldsymbol{D}(\boldsymbol{M})_{ij} = d(\boldsymbol{M}, \boldsymbol{x}_i, \boldsymbol{y}_j)$ is the distance between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ w.r.t. the metric $d(\boldsymbol{M})$. $\boldsymbol{T}^*$ is a matrix indicating the correspondence relationship, where $t^*_{i,j} = \boldsymbol{T}^*(i, j)$ actually measures whether or how the pair $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ corresponds to the same temporal position or structure. Ideally, only the differences between those elements within the same temporal positions reflect the differences between the entire sequences. However, due to the different sampling rates, the non-uniform evolution speeds of elements, local temporal distortions, etc, different sequences have different lengths and exhibit local temporal differences, so the i-th element in $\boldsymbol{X}$ and the j-th element in $\boldsymbol{Y}$ may not correspond to the same relative position. $\boldsymbol{T}^*$ is used to align the elements corresponding to the same temporal structure or position. Generally, the determination of $\boldsymbol{T}^*$ can be formulated as

$$\boldsymbol{T}^* = arg \min_{\boldsymbol{T} \in \boldsymbol{\Phi}} \langle \boldsymbol{T}, \boldsymbol{D}(\boldsymbol{M}) \rangle + \mathscr{R}(\boldsymbol{T}), \tag{3}$$

where $\boldsymbol{\Phi}$ is the feasible set of $\boldsymbol{T}$, which is a subset of $\mathbb{R}^{L_X \times L_Y}$ with some constraints, and $\mathscr{R}(\boldsymbol{T})$ is a regularization term on $\boldsymbol{T}$. The distance is symmetric if $\forall \boldsymbol{T} \in \boldsymbol{\Phi}, \boldsymbol{T}^T \in \boldsymbol{\Phi}$ and $\mathscr{R}(\boldsymbol{T}) = \mathscr{R}(\boldsymbol{T}^T)$. Different distance measures for sequences differ in the constraints imposed to the feasible set, the regularization term, and the optimization or inference method.

DTW [2]. DTW calculates an optimal alignment between two sequences with three constraints: boundary, continuity, and monotonicity. In the unified formulation, DTW restricts $\boldsymbol{T}$ to be a binary matrix, in which $t_{i,j} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ are aligned and $t_{ij} = 0$ otherwise. DTW instantiates the formulation (3) by setting:

$$\begin{array}{l} \mathscr{R}(\boldsymbol{T}) = 0; \\ \boldsymbol{\Phi} = \{\boldsymbol{T} \in \{0,1\}^{L_X \times L_Y} | \boldsymbol{T}_{1,1} = 1, \boldsymbol{T}_{L_X, L_Y} = 1; \\ \qquad \boldsymbol{T}\mathbf{1}_{L_Y} > \mathbf{0}_{L_X}, \boldsymbol{T}^T \mathbf{1}_{L_X} > \mathbf{0}_{L_Y}; \\ \qquad if\ t_{i,j} = 1, then\ t_{i-1,j+1} = 0, t_{i+1,j-1} = 0, \\ \qquad \forall 1 < i < L_X, 1 < j < L_Y \} \end{array} , \tag{4}$$

where $\mathbf{1}_b$ and $\mathbf{0}_b$ are the b-dimensional vectors with all one and zero elements, respectively, and ">" should be understood as element-wise. DTW solves Eq. (3) with constraints (4) via dynamic programming.

Variants of DTW. Most variants of DTW impose additional or relaxed constraints on the feasible set and therefore fit into our formulation. For example, in [46], additional locality constraints $\boldsymbol{T}\mathbf{1}_{L_Y} \leq a\mathbf{1}_{L_X}, \boldsymbol{T}^T\mathbf{1}_{L_X} \leq a\mathbf{1}_{L_Y}$ are imposed to restrict the amount of alignment; in [47], the continuity constraint is stricter by setting $\boldsymbol{T}\mathbf{1}_{L_Y} = \mathbf{1}_{L_X}$ or $\boldsymbol{T}^T\mathbf{1}_{L_X} = \mathbf{1}_{L_Y}$.

Optimal Transport (OT) [48]. Originally, OT measures the distance between distributions. A sequence can be viewed as an empirical probability by taking its elements as independent supporting points. In this way, although the temporal information is lost, OT can be applied to sequences. OT naturally has the form of Eq. (3), where

$$\begin{aligned}&\mathscr{R}(\boldsymbol{T}) = 0;\\ &\boldsymbol{\Phi} = \{\boldsymbol{T} \in \mathbb{R}_+^{L_X \times L_Y} | \boldsymbol{T}\mathbf{1}_{L_Y} = \tfrac{1}{L_X}\mathbf{1}_{L_X}, \boldsymbol{T}^T\mathbf{1}_{L_X} = \tfrac{1}{L_Y}\mathbf{1}_{L_Y}\}\end{aligned} \tag{5}$$

Solving the original OT is expensive. The Sinkhorn distance [49] smooths the OT problem by adding an entropy regularization term to $\boldsymbol{T}$, and the resulting optimum can be efficiently determined by Sinkhorn's fixed point iterations. It instantiates the formulation Eq. (3) by setting:

$$\begin{aligned}&\mathscr{R}(\boldsymbol{T}) = \lambda(\sum_{i=1}^{N}\sum_{j=1}^{M} t_{ij}\log t_{ij});\\ &\boldsymbol{\Phi} = \{\boldsymbol{T} \in \mathbb{R}_+^{L_X \times L_Y} | \boldsymbol{T}\mathbf{1}_{L_Y} = \tfrac{1}{L_X}\mathbf{1}_{L_X}, \boldsymbol{T}^T\mathbf{1}_{L_X} = \tfrac{1}{L_Y}\mathbf{1}_{L_Y}\}\end{aligned}, \tag{6}$$

where λ is a preset balancing coefficient.

Order-preserving Wasserstein Distance (OPW) [3], [50]. OPW casts sequence alignment as the OT problem. It imposes two regularization terms to the original OT problem to preserve the global temporal information. The first regularization favors $\boldsymbol{T}$ with large *inverse difference moment* which is calculated as

$$I(\boldsymbol{T}) = \sum_{i=1}^{L_X}\sum_{j=1}^{L_Y} \frac{t_{ij}}{(\frac{i}{L_X} - \frac{j}{L_Y})^2 + 1}, \tag{7}$$

The second regularization encourages the distribution of $\boldsymbol{T}$ to be similar to a prior distribution $\boldsymbol{P}$:

$$p_{ij} := \boldsymbol{P}(i,j) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\ell^2(i,j)}{2\sigma^2}}, \tag{8}$$

where $\ell(i,j) = \frac{|i/L_X - j/L_Y|}{\sqrt{1/{L_X}^2 + 1/{L_Y}^2}}$. Both regularization terms encourage alignments between elements with similar relative temporal positions and restrict the matching between elements that are far away temporally. OPW instantiates the formulation (3) by setting:

$$\begin{aligned}&\mathscr{R}(\boldsymbol{T}) = \lambda_1 I(\boldsymbol{T}) + \lambda_2 KL(\boldsymbol{T}||\boldsymbol{P});\\ &\boldsymbol{\Phi} = \{\boldsymbol{T} \in \mathbb{R}_+^{L_X \times L_Y} | \boldsymbol{T}\mathbf{1}_{L_Y} = \tfrac{1}{L_X}\mathbf{1}_{L_X}, \boldsymbol{T}^T\mathbf{1}_{L_X} = \tfrac{1}{L_Y}\mathbf{1}_{L_Y}\}\end{aligned}, \tag{9}$$

where λ_1 and λ_2 are preset balancing coefficients, and $KL(\boldsymbol{T}||\boldsymbol{P})$ is the Kullback-Leibler divergence. OPW solves Eq.(3) with constraints (9) by the Sinkhorn's matrix scaling algorithm. Each element t^*_{ij} in the learned $\boldsymbol{T}^*$ can be viewed as the probability of aligning $\boldsymbol{x}_i$ to $\boldsymbol{y}_j$.

We observe that these distances actually share the common formulation and can be considered as *meta-distances* built on $d(\boldsymbol{M})$, although they have different motivations. For these distances, the determination of $\boldsymbol{T}^*$ depends on the metric $d(\boldsymbol{M})$. In the literature [22], [51], the metric is called the *ground metric*. We follow this name to distinguish it with the meta-distance for sequences.

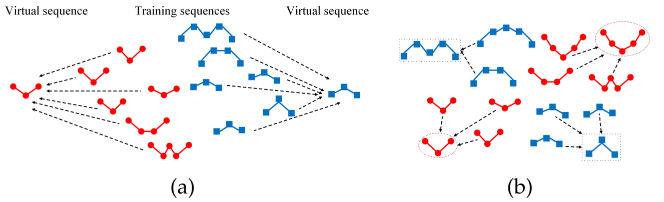


Fig. 2. (a) Temporal structure (TS) based virtual sequences. All training sequences from the same class (with the same color) are associated with the same virtual sequence, all components in all virtual sequences are orthogonal to each other so that the virtual sequences for different classes are well separated; (b) Large margin (LM) based virtual sequences. Training sequences that have large margins from other classes are selected as virtual sequences (bounded by dotted frames). The virtual sequence of a training sequence is set as the nearest selected training sequence from the same class.

4 Regressive Virtual Sequence Metric Learning

4.1 Problem

With the unified formulation (1) and (3), we view the meta-distance as a function of the ground metric parameterized by $\boldsymbol{M}$. The goal of our method is to learn a ground metric $\boldsymbol{M}$ resulting in a meta-distance $g_{\boldsymbol{M}}(\boldsymbol{X}, \boldsymbol{Y})$ (1), such that the meta-distances between sequences from different classes are large, and those between sequences from the same class are small. We learn a squared Mahalanobis-like distance [8] as the ground metric, i.e.,

$$d(\boldsymbol{M}, \boldsymbol{x}_i, \boldsymbol{y}_j) = (\boldsymbol{x}_i - \boldsymbol{y}_j)^T\boldsymbol{M}(\boldsymbol{x}_i - \boldsymbol{y}_j), \tag{10}$$

where $\boldsymbol{M}$ is a positive semi-definite matrix and can be decomposed as $\boldsymbol{M} = \boldsymbol{W}\boldsymbol{W}^T$, $\boldsymbol{W} \in \mathbb{R}^{b \times b'}$ and b' is greater than or equal to the rank of $\boldsymbol{M}$. This is equivalent to transform all elements $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ with a projection $\boldsymbol{W}$.

Specially, let $\{\boldsymbol{X}^n, z^n\}_{n=1}^N$ be a set of N training sequences, where $\boldsymbol{X}^n = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{L^n}] \in \mathbb{R}^{b \times L^n}$ is the n-th sequence with length L^n. Different sequences may have different lengths. $\boldsymbol{x}_i, i = 1, \cdots, L^n$ are sampled in $\mathbb{R}^b$, and z^n is the class label of $\boldsymbol{X}^n$. We are interested in learning a meta-distance $g_{\boldsymbol{M}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'})$ with the form of Eq. (1) by learning $\boldsymbol{W}$ from the training set, such that the resulting $g_{\boldsymbol{M}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'}) = g_{\boldsymbol{I}}(\boldsymbol{W}^T\boldsymbol{X}^n, \boldsymbol{W}^T\boldsymbol{X}^{n'})$ captures the idiosyncrasy of sequence data and better separates sequences from different classes, where $g_{\boldsymbol{I}}$ means that $\boldsymbol{M} = \boldsymbol{I}$ when constructing Eq. (2): $\boldsymbol{D}_{\boldsymbol{I}}(\boldsymbol{W}) = [d(\boldsymbol{I}, \boldsymbol{W}^T\boldsymbol{x}_i, \boldsymbol{W}^T\boldsymbol{y}_j)]_{ij}$.

The difficulty largely lies in the fact that in Eq. (1), $\boldsymbol{T}^*$ is not fixed, but needs to be inferred by optimizing Eq.(3) for each sequence pair. The inference of $\boldsymbol{T}^*$ also heavily depends on $\boldsymbol{W}$. Once $\boldsymbol{W}$ changes, $\boldsymbol{T}^*$ for each sequence pair changes accordingly. Also, for any sequence pair, the corresponding optimal alignment $\boldsymbol{T}^*$ needs to be inferred individually. The cost of constructing a single must-link/cannot-link or relative constraint for sequence distance is much larger than for vector distance. Therefore, it can be computationally prohibitive to learn $\boldsymbol{W}$ with such constraints whose number is quadratic or cubic with the number of training sequences.

4.2 Objective and Optimization

RVML [9] introduces a new kind of constraints that moving each sample to its corresponding pre-defined virtual point. Compared with must-link/cannot-link and relative constraints, the number of such virtual point-based constraints is greatly reduced since it is linear with the number of samples. We extend RVML to sequence data by associating a virtual sequence instead of a virtual point for each sequence sample. Let $\boldsymbol{V}^n = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_{l^n}] \in \mathbb{R}^{b' \times l^n}$ be the virtual sequence related to $\boldsymbol{X}^n$. b' and l^n are the dimensionality and the number of elements in $\boldsymbol{V}^n$, respectively, which may not equal to those in $\boldsymbol{X}^n$. $\boldsymbol{V}^n$ is a function of $\boldsymbol{X}^n$ and z^n: $\boldsymbol{V}^n = f(\boldsymbol{X}^n, z^n)$. The setting of $\boldsymbol{V}^n$ can be very flexible, e.g., each $\boldsymbol{X}^n$ can be associated with a different $\boldsymbol{V}^n$, while all or a part of sequences from the same class can be associated with the same virtual sequence as shown in Fig. 2; a virtual sequence can be different from any training sequence as shown in Fig. 2(a), while for some $\boldsymbol{X}^n$, the associated $\boldsymbol{V}^n$ can be set to the training sequence itself or another training sequence as shown in Fig. 2(b). Generally, the virtual sequences for training sequences from different classes are set far away from each other.

We first assume that the virtual sequences for all training sequences have been obtained. The goal is to learn a transformation $\boldsymbol{W}$ by minimizing the meta-distances between the training sequences and their associated virtual sequences, i.e.,

$$\begin{aligned} &\min_{\boldsymbol{W}} \tfrac{1}{N} \sum_{n=1}^{N} g_{\boldsymbol{I}}(\boldsymbol{W}^T\boldsymbol{X}^n, \boldsymbol{V}^n) + \beta\|\boldsymbol{W}\|_{\mathcal{F}}^2 \\ &= \tfrac{1}{N} \sum_{n=1}^{N} \langle \boldsymbol{T}^{n*}, \boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W}) \rangle + \beta\|\boldsymbol{W}\|_{\mathcal{F}}^2 \\ &s.t.\ \boldsymbol{T}^{n*} = arg \min_{\boldsymbol{T} \in \boldsymbol{\Phi}} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W}) \rangle + \mathscr{R}(\boldsymbol{T}^n) \end{aligned} \quad (11)$$

where $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm and β is a hyperparameter that balances the two items.

The underlying $\boldsymbol{T}^{n*}, n = 1,\cdots,N$ for all training-virtual sequence pairs depend on the variable $\boldsymbol{W}$. We treat them as latent structures. In Eq.(11), if $\mathscr{R}(\boldsymbol{T})$ does not depend on $\boldsymbol{W}$, the inferences over $\boldsymbol{T}^{n*}, n = 1,\cdots,N$ in the constraints are actually minimizing the same objective as the optimization over $\boldsymbol{W}$. This allows us to jointly learn $\boldsymbol{W}$ and $\boldsymbol{T}^{n*}, n = 1,\cdots,N$ by optimizing the following objective:

$$\min_{\boldsymbol{W}, \boldsymbol{T}^n} \frac{1}{N} \sum_{n=1}^{N} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W}) \rangle + \beta\|\boldsymbol{W}\|_{\mathcal{F}}^2 + \mathscr{R}(\boldsymbol{T}^n). \quad (12)$$

The objective function Eq. (12) is not jointly convex on $\boldsymbol{W}$ and $\boldsymbol{T}^n, n = 1, \cdots, N$. We minimize it by alternatively updating the metric and the latent alignments. We first fix $\boldsymbol{T}^n, n = 1, \cdots, N$ and update $\boldsymbol{W}$. In this case, the regularization term $\mathscr{R}(\boldsymbol{T})$ can be discarded and the objective can be reformulated as

$$\begin{aligned} &\tfrac{1}{N} \sum_{n=1}^{N} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W}) \rangle + \beta\|\boldsymbol{W}\|_{\mathcal{F}}^2 \\ &= \tfrac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{L^n} \sum_{j=1}^{l^n} t_{ij}^n \|\boldsymbol{W}^T\boldsymbol{x}_i^n - \boldsymbol{v}_j^n\|_2^2 + \beta\|\boldsymbol{W}\|_{\mathcal{F}}^2 \end{aligned} . \quad (13)$$

Algorithm 1 RVSML

```
1: Input: A set of training sequences $\{\boldsymbol{X}^n\}_{n=1}^N$ and the
   associated virtual sequences $\{\boldsymbol{V}^n\}_{n=1}^N$
2: Output: the transformation $\mathbf{W}$
3: Initialize the alignment matrices $\boldsymbol{T}^n, n = 1, \cdots, N$ for
   all training-virtual sequence pairs.
4: while $\mathbf{W}$ has not converged do
5:   Update $\mathbf{W}$ by Eq. (13)
6:   for $n = 1, \cdots, N$ do
7:     Update $\boldsymbol{T}^n$ by optimizing Eq. (16)
8:   end for
9: end while
```

Minimizing Eq.(13) is a weighted regression problem, which admits a closed form solution:

$$\boldsymbol{W}^* = \boldsymbol{A}^{-1}(\sum_{n=1}^{N} \sum_{i=1}^{L^n} \sum_{j=1}^{l^n} t_{ij}^n \boldsymbol{x}_i^n \boldsymbol{v}_j^{nT}), \quad (14)$$

where

$$\boldsymbol{A} = \sum_{n=1}^{N} \sum_{i=1}^{L^n} \sum_{j=1}^{l^n} t_{ij}^n \boldsymbol{x}_i^n \boldsymbol{x}_i^{nT} + \beta N \boldsymbol{I}. \quad (15)$$

This solution can be simply derived by setting the derivative of Eq.(13) to 0.

We then update $\boldsymbol{T}^n, n = 1, \cdots, N$ by fixing $\boldsymbol{W}$. In this case, the matrix $\boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W})$ consisting of all pairwise squared Euclidean distances between $\boldsymbol{W}\boldsymbol{x}_i^n$ and $\boldsymbol{v}_j^n$ is also fixed, and the irrelevant regularization term $\|\boldsymbol{W}\|_{\mathcal{F}}^2$ can be discarded. We further observe that the optimizations of $\boldsymbol{T}^n$ for $n = 1, \cdots, N$ are independent. Therefore, we can solve them separately by applying the inference Eq.(3) to each training-virtual sequence pair:

$$\boldsymbol{T}^{n*} = arg \min_{\boldsymbol{T}^n \in \boldsymbol{\Phi}} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W}) \rangle + \mathscr{R}(\boldsymbol{T}^n). \quad (16)$$

The two updating procedures are repeated until convergence or reaching a maximum number of iterations. We call this framework *Regressive Virtual Sequence Metric Learning (RVSML)* and summarize it in Alg. 1.

Convergence. Both updating procedures of Alg. 1 decrease the value of the objective (12). 0 is a trivial lower bound of the objective (12). Therefore, Alg. 1 ensures the convergence to a local solution.

Instantiation and complexity. Alg. 1 can be applied to learn any meta-distance with the form Eq. (1) as discussed in Sec. 3. A specific meta-distance instantiates step.7 in Alg. 1, i.e., the inference of $\boldsymbol{T}^n$. For instance, for DTW, step.7 is performed by dynamic programming; for OPW, step.7 is performed by Sinkhorn's matrix scaling. As long as sufficient inference or optimization method for an instantiation of Eq. (16) is available, Alg. 1 can be efficiently performed. When instantiated by DTW and OPW, the complexity per iteration is $O(b^2b' + NmTb^2 + NmTbb')$, where m and T are the average lengths of virtual sequences and training sequences, respectively.

4.3 Links with Other Methods

Connection with RVML [9]. RVML can be viewed as a special case of the proposed RVSML. By regarding vector data as sequences with only one element and setting the length of all virtual sequences to 1, the alignment between any training-virtual sequence pair by any meta-distance is unique. Therefore, RVSML degenerates into RVML.

Connection to must-link/cannot-link constraints. Most classical metric learning methods employ pair-based or triplet-based constraints to achieve a large margin between similar and dissimilar sample pairs, i.e., the distance between the samples from the same class is below a threshold η_1, and the distance between those from different classes is above another threshold η_{-1}.

$$\begin{array}{l} g_{\boldsymbol{M}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'}) \leq \eta_1, for\ z^n = z^{n'} \\ g_{\boldsymbol{M}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'}) \geq \eta_{-1}, for\ z^n \neq z^{n'} \end{array}. \tag{17}$$

When the meta-distance $g_{\boldsymbol{W}}$ is a real metric, in the transformed space induced by RVSML, the distances between similar and dissimilar sequence pairs gain the following margins:

$$\begin{array}{l} \eta_1 = 2 \max\limits_{(\boldsymbol{X}^n, \boldsymbol{V}^n)} g_{\boldsymbol{I}}(\boldsymbol{W}^T \boldsymbol{X}^n, \boldsymbol{V}^n) \\ \eta_{-1} = \min\limits_{\boldsymbol{V}^n, \boldsymbol{V}^{n'}, \boldsymbol{V}^n \neq \boldsymbol{V}^{n'}} g_{\boldsymbol{I}}(\boldsymbol{V}^n, \boldsymbol{V}^{n'}) - \eta_1 \end{array}. \tag{18}$$

Although some well-known meta-distances such as DTW do not satisfy the triangle inequality, intuitively, dissimilar sequences are still pushed relatively far away because they are moved to different distant virtual sequences.

4.4 Virtual Sequences Generation

The virtual sequences can be generated using various approaches according to the desired properties of the metric, the prior knowledge on the data, etc. In this section, we develop three approaches.

Temporal-structure (TS) based virtual sequences. Intuitively, the evolution of a sequence pattern can be segmented into several ordered stages and each stage corresponds to a temporal structure, e.g., an action can be identified by a series of ordered key poses. If $\mathbf{W}$ is able to project the elements corresponding to different temporal structures to different clusters which are far away from each other, different sequence classes would become easier to distinguish.

Following this intuition, as shown in Fig. 2(a), we construct a virtual sequence for each class, which consists of vectors w.r.t. the ordered basic temporal structures shared by this class. Let m be the number of temporal structures per class. There are Cm temporal structures for all C classes. We define the vector for the u-th temporal structure as a unit vector $\boldsymbol{e}_u \in \mathbb{R}^{Cm}$, in which only the u-th attribute is 1 and all other attributes are 0. Therefore, the virtual sequence for the c-th class is $\boldsymbol{V}_c^T = [\mathbf{0}^{m\times m}, \cdots, \mathbf{0}^{m\times m}, \boldsymbol{I}^{m\times m}, \mathbf{0}^{m\times m}, \cdots, \mathbf{0}^{m\times m}] \in \mathbb{R}^{Cm\times m}$, where only the c-th block square matrix is the identity matrix and all other $C-1$ blocks are the null matrices, i.e., $f(\boldsymbol{X}^n, z^n) = \boldsymbol{V}_{z^n} = [\boldsymbol{e}_{(z^n-1)m+1}, \cdots, \boldsymbol{e}_{(z^n-1)m+m}]$. In this way, we generate C virtual sequences each consists of m unit vectors. All unit vectors in all virtual sequences are orthogonal and the active attribute for each vector is attempted to be discriminative for one temporal structure. Each component of a virtual sequence aims at representing a temporal structure of the related class. By making all components orthogonal to each other, all temporal structures and all virtual sequences are well separated. The dimensionality Cm of the unit vector may be different from the dimensionality b of the elements in the original training sequences. When $Cm < b$, the learned $\mathbf{W}$ also achieves dimensionality reduction for sequence data.

This generation approach has low complexity and is independent of the training sequences. The virtual sequences are directly generated without extra computation. Therefore, in our experiments, we use this approach to generate virtual sequences unless otherwise specified.

Large-margin-based virtual sequences. In this approach, we construct a virtual sequence for a training sequence based on the relative location of the training sequence w.r.t. other sequences. Special attention should be pay to those sequences distributed near the boundaries among different classes. As shown in Fig. 2(b), if we push any sequence near the boundaries to another sequence far away from the boundaries, the margins among different sequence classes would become larger.

Specifically, given a meta-distance measure with the squared Euclidean ground metric, for each training sequence $\boldsymbol{X}^n$, we define its smallest margin as $M_n^s = g_n^{sb} - g_n^{sw}$, where g_n^{sw} and g_n^{sb} are the meta-distances from $\boldsymbol{X}^n$ to the nearest training sequence from the same class and the nearest sequence from other classes, respectively. We also calculate the average pair-wise meta-distance g_n^{aw} between $\boldsymbol{X}^n$ and other sequences from the same class, and the average meta-distance g_n^{ab} between $\boldsymbol{X}^n$ and sequences from other classes. We define the average margin of $\boldsymbol{X}^n$ as $M_n^a = g_n^{ab} - g_n^{aw}$. For each class, we select the training sequences whose M_n^a and M_n^a are both positive as candidates. We sort the candidates according to their average margins in descending order. The top candidate is first selected into the target set of this class. For each ordered candidate, we calculate its meta-distances to all sequences in the current target set. If the smallest meta-distance is larger than a threshold, this candidate is also added to the target set. The threshold is set to half the mean of all pairwise meta-distances between sequences from all classes to increase the diversity among the selected target sequences. After all candidates are processed in order, the sequences in the final target set are considered to have large margins with other classes and hence serve as the target sequences of this class. The virtual sequence for a training sequence is selected as the nearest target sequence of the same class.

Barycenter-based virtual sequences. In this approach, the virtual sequence of all training sequences of a class is constructed as the barycenter of this class. The barycenter is also a sequence with pre-set length and its calculation depends on the meta-distance. For N^c training sequences $\boldsymbol{X}_k, k = 1, \cdots, N^c$ of the c-th class, given a meta-distance $g_{\boldsymbol{I}}(\cdot,\cdot)$ with the squared Euclidean ground metric, their barycenter $\boldsymbol{U}^c$ is defined as

$$\boldsymbol{U}^c = arg\min_{\boldsymbol{U}^c} \sum_{k=1}^{N^c} \frac{1}{N^c} g_{\boldsymbol{I}}(\boldsymbol{U}^c, \boldsymbol{X}_k). \tag{19}$$

$\boldsymbol{U}^c$ can be viewed as lying near the center of the distribution of sequences of this class. If sequences from different classes are pushed towards their centers respectively, the margins

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.3010568, IEEE Transactions on Pattern Analysis and Machine Intelligence

among different classes are enlarged. We employ the modified DTW barycenter algorithm in [52], [53] and the OPW barycenter algorithm in [54] to calculate the barycenter when the meta-distance is instantiated by DTW and OPW, respectively. When the distribution of each class is multi-modal and the variations of sequence samples are large, we can group the training sequences of each class into several clusters and construct the barycenter-based virtual sequences by viewing clusters as subclasses.

Semantic-based virtual sequences. The proposed RVSML can be extended to tackle the zero-shot sequence classification problem. We are given a set of training sequences $\{\boldsymbol{X}^n, z^n\}_{n=1}^N$ from seen classes and a sentence or phrase describing each seen class, respectively. For each class, we represent its language description by a semantic sequence of vectors, where each vector is the 300-dimensional pre-trained Word2Vec [55] embedding for a word in the description. We use this semantic sequence as the virtual sequence for all training sequences of this class. Let $\boldsymbol{V}^z$ denote the semantic sequence for the z-th class, the virtual sequence for $\boldsymbol{X}^n$ is $\boldsymbol{V}^{z^n}$. We employ the proposed RVSML instantiated by a meta-distance to learn a transformation from the virtual space to the semantic space.

At test time, the goal is to classify test sequences from unseen classes, given only the sentence or phrase descriptions of these new unseen classes. We represent these descriptions by semantic sequences. For a test sequence, we use the transformation learned by RVSML to map it into the semantic space. We calculate the meta-distances from the transformed test sequence to semantic sequences of all unseen classes. The test sequence is classified into the class with the smallest meta-distance.

By default, RVSML employs TS-based virtual sequences. For ease of distinction, we denote RVSML with LM-based virtual sequences, barycenter-based virtual sequences, and semantic-based virtual sequences for zero-shot learning by RVSML-LM, RVSML-BC, and RVSML-ZS respectively.

5 Deep Regressive Virtual Sequence Metric Learning

A linear transformation of the ground metric may not be able to properly regress the training sequences to the specified virtual sequences, because such latent-structure-involved regression may be complex and highly non-linear. With the success of deep learning, the proposed RVSML can also take the advantage of deep neural networks to learn a nonlinear mapping from the original space to an embedding space, where the transformed sequences are better pushed to the corresponding virtual sequences by using the squared Euclidean distance as the ground metric. We denote the deep extension of RVSML by Deep-RVSML.

The model architecture of Deep-RVSML is shown in Fig. 3. For a given sequence $\boldsymbol{X}^n = [\boldsymbol{x}_1^n, \cdots, \boldsymbol{x}_{L^n}^n]$, all its elements $\boldsymbol{x}_i \in \mathbb{R}^b, i = 1, \cdots, L^n$ are input to a deep encoder network, respectively. In this paper, the encoder network is composed of three fully connected layers and a linear output layer. Each hidden layer contains 1024 neurons followed by rectified linear unit (ReLu) activation. The number of nodes in the output layer equals the dimension b' of elements in the virtual sequences. The output of the encoder network for embedding $\boldsymbol{x}_i$ is denoted by $h(\boldsymbol{x}_i, \theta)$, where h represents the function implemented by the network and θ represents the set of parameters of the network. As a result, the input sequence is transformed into an encoded sequence $h(\boldsymbol{X}^n, \theta) = [h(\boldsymbol{x}_1, \theta), \cdots, h(\boldsymbol{x}_{L^n}, \theta)]$. Each training sequence $\boldsymbol{X}^n$ is associated with a virtual sequence $\boldsymbol{V}^n = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_{l^n}] \in \mathbb{R}^{b' \times l^n}$. The objective is to minimize the meta-distance between the encoded training sequences and the corresponding virtual sequences.

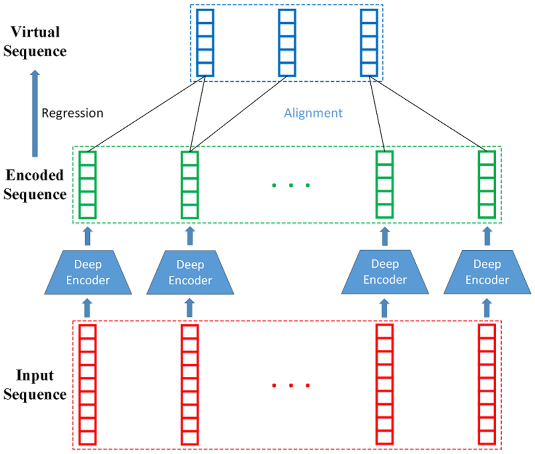


Fig. 3. The model architecture of Deep-RVSML.

$$\begin{aligned} &\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} g_{\boldsymbol{I}}(h(\boldsymbol{X}^n, \theta), \boldsymbol{V}^n) = \frac{1}{N} \sum_{n=1}^{N} \langle \boldsymbol{T}^{n*}, \boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta) \rangle \\ &s.t.\ \boldsymbol{T}^{n*} = arg \min_{\boldsymbol{T} \in \boldsymbol{\Phi}} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta) \rangle + \mathscr{R}(\boldsymbol{T}^n) \end{aligned} \tag{20}$$

where $\boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta)$ denotes the matrix of all the pairwise Euclidean distances between the embedding representations in $h(\boldsymbol{X}^n, \theta)$ and the elements in $\boldsymbol{V}^n$. The optimization of Eq. (20) follows the similar alternating procedures with the linear RVSML. When the parameters of the deep encoder network are fixed, the procedure for updating the alignments remain the same. Specifically, after the embedding representations are obtained by the network, $\boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta)$ can be calculated straightforwardly. The alignments between any encoded sequence and its corresponding virtual sequence can be inferred as follows:

$$\boldsymbol{T}^{n*} = arg \min_{\boldsymbol{T} \in \boldsymbol{\Phi}} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta) \rangle + \mathscr{R}(\boldsymbol{T}^n) \tag{21}$$

which is solved by the specified meta-distance instance such as DTW and OPW as in Eq. (3).

When the alignments are fixed, the objective (20) can be formulated as follows:

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \langle \boldsymbol{T}^{n*}, \boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta) \rangle = \frac{1}{N} \sum_{n,i,j} t_{i,j}^{n*} \left\| h(\boldsymbol{x}_i^n, \theta) - \boldsymbol{v}_j^n \right\|_2^2 \tag{22}$$

The alignments decouple the temporal relations of elements in sequences by assigning weights on different element pairs. In this way, each training-virtual sequence pair is decomposed into $L^n l^n$ independent vector pairs with different weights. Therefore, Eq. (22) is a standard weighted regression problem and can be optimized in a standard manner. Specifically, to update the parameters of the network, we calculate the gradient of Eq. (22) w.r.t. θ and employ the back propagation algorithm. In this paper, we employ the Adam optimizer to train the network.

The two procedures are alternated until convergence. In this way, the alignments and the network are jointly learned. For a test sequence, we only need to input all its elements into the trained network to obtain the encoded sequence.

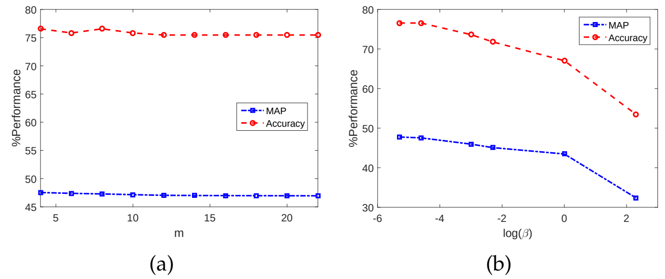
6 Experimental Results

6.1 Experimental setup

Datasets. MSR Action3D dataset [56] contains 567 depth video sequences from 20 action classes. We follow the splits in [57], [58] to divide the dataset into training and testing sets. **MSR Daily Activity3D dataset** [57] consists of 320 Kinect daily activity sequences from 16 activity classes. We follow the splits in [57], [58] to divide the dataset into training and test sets. **ChaLearn Gesture dataset** [59] consists of Kinect video sequences from 20 gesture types. The dataset is partitioned into training, validation and test sets. **"Spoken Arabic Digits (SAD)" dataset** from the UCI Machine Learning Repository [60] contains 8,800 vector sequences from ten digit classes with 880 sequences per class. The dataset is partitioned into training and test sets. **"High-quality recordings of Australian Sign Language signs (HAS)" dataset** [60], [61] consists of $2,565$ sequences from 95 classes with 27 sequences per class. Following [62], we split the sequences into five subsets and conduct experiments by five-fold cross-validation. Each time four subsets are used for training and the remaining subset is used for testing. **"NTU RGB+D" dataset** [63] consists of 56,880 Kinect video samples from 60 action classes. In the Cross-Subject (CS) evaluation, the dataset is split into a training set of 40,320 sequences and a test set of 16,560 sequences. In the Cross-View (CV) evaluation, the dataset is split into a training set of 37,920 sequences and a test set of 18,960 sequences. Sequences have different lengths in all datasets, e.g., the length varies from 6 to 100 on the ChaLearn dataset and from 4 to 93 on the SAD dataset.

Sequence representations. For video sequences, we extract a feature vector from each frame, so as to represent each video as a sequence of frame-wide vectors. For the MSR Action3D dataset, we adopt the 192-dimensional relative 3D joint angles based frame-wide vectors as in [58]. For the MSR Activity3D dataset, we employ the 390-dimensional relative 3D joint positions based frame-wide features as in [57]. For the ChaLearn dataset, we adopt the 100-dimensional joint-based frame-wide vectors as in [64]. For the SAD dataset, the sequences have already been represented as a series of 13-dimensional mel-frequency cepstrum coefficients features. For the HAS dataset, the sequences have already been represented as a series of 22-dimensional feature vectors. For the NTU dataset, we concatenate all joint locations of the two subjects to form the 150-dimensional raw skeleton-based frame-wide features.

Classification and evaluation measures. We evaluate the proposed RVSML instantiated by DTW and OPW, respectively. The codes are publicly available[1]. After learning the ground metric, we employ the 1-nearest neighbor (NN) classifier with the DTW distance and the OPW distance to perform sequence classification, respectively. The parameters λ_1, λ_2, and σ of OPW are fixed to 10, 0.1, and 12, respectively, on the Activity3D dataset, 50, 0.1, 12, respectively, on the HAS dataset, and 50, 0.1, and 1, respectively, on other datasets, as suggested in [50]. We report accuracy as the performance measure. Following [3], [50], we also regard each test sequence as a query to retrieval all training sequences and report the mean average precision (MAP).



(a) (b)

Fig. 4. Performances of RVSML as functions of (a) m and (b) $log(\beta)$ on the MSR Action3D dataset.

6.2 Influence of hyper-parameters

The RVSML framework has one hyper-parameter: β. The generation of the TS-based virtual sequences has one hyper-parameter: m, the number of elements in each virtual sequence. We evaluate their influence on RVSML instantiated by OPW on the MSR Action3D dataset. We first evaluate the influence of m by fixing β to 0.01. The performances as functions of m are shown in Fig. 4(a). We observe that a small m within the range of 4 to 8 works well. When $m = 1$, RVSML is equivalent to treating elements in sequences as independent samples and degenerates into RVML. As shown in Tab. 1, RVSML with $m > 1$ outperforms RVML, this indicates that introducing more temporal structures helps to better explore the temporal information. However, since the dimensionality of elements in virtual sequences depends on m, the size of $\boldsymbol{W}$ increases with m. Therefore, the parameters in $\boldsymbol{W}$ may be too many to be adequately trained for large m. We then evaluate β by fixing m to 4. The performances as functions of $log(\beta)$ are illustrated in Fig. 4(b). Generally, as β is a regularization coefficient, it seems that very small β leads to satisfactory results.

6.3 Comparison with metric learning methods

We compare the proposed RVSML with the baseline NN classifier without metric learning (Ori) and several state-of-the-art conventional metric learning methods: ITML [5], LMNN [7], SCML [16], and RVML [9]. These methods are originally developed for vector representations. We apply them to sequences by viewing all elements in the sequence from a class as independent samples of this class. On the ChaLearn dataset, SCML learned 0 LDA base and hence we remove it for comparison. For RVML, we employ the class-based virtual points. On the HAS dataset, the training of LMNN is much slower than other methods, so we fix the metric learned in one validation. In addition to the average performance measures, the standard deviations over different folds are shown in parentheses on this dataset.

We also compare with two metric learning methods for sequence data, including LDMLT [33] and SWMD [23].

[1] https://github.com/BingSu12/RVSML

TABLE 1
Comparison of the proposed RVSML variants instantiated by (left) DTW and (right) OPW with other metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the MSR Action3D dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
Ori [50]	58.95	81.32	58.70	**84.25**
ITML [5]	59.19	80.95	59.48	83.52
LMNN [7]	54.14	80.95	32.73	82.42
SCML [16]	42.79	63.00	39.63	64.10
RVML [9]	57.41	80.95	44.58	73.63
LDMLT [33]	64.29	84.98	53.61	80.59
SWMD [23]	59.65	80.95	43.23	66.67
RVSML	59.30	82.78	47.54	76.56
RVSML-LM	63.10	83.15	**60.77**	**84.25**
RVSML-BC	**65.31**	**85.35**	59.21	78.75

TABLE 2
Comparison of the proposed RVSML variants instantiated by (left) DTW and (right) OPW with other metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the MSR Activity3D dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
Ori [50]	33.79	58.75	34.62	**58.13**
ITML [5]	33.80	58.75	33.69	**58.13**
LMNN [7]	32.24	55.63	32.06	**58.13**
SCML [16]	29.42	45.62	28.50	45.00
RVML [9]	41.55	60.62	38.73	56.87
LDMLT [33]	36.56	55.00	34.84	54.37
SWMD [23]	37.81	61.25	35.62	55.00
RVSML	**42.18**	**62.50**	36.64	57.50
RVSML-LM	38.98	59.38	36.88	50.62
RVSML-BC	38.25	59.38	**41.43**	54.37

SWMD can not be directly applied to unconstrained sequences because it requires that the elements in sequences are from a finite set and learns the weights for all possible elements in this set. The weights determine the marginal constraints for the transport matrix. We modify SWMD by removing the weight learning procedures and setting the marginal constraints uniformly so that SWMD can be applied to unconstrained sequences. For different metric learning methods, the NN classifiers with DTW and OPW distances are used for classification by taking the learned metrics as ground metrics, respectively. Although conventional metric learning methods produce the same projected sequences, they perform differently by the NN classifier with different distances.

RVSML, RVSML-LM, and RVSML-BC use the TS-based, LM-based, and BC-based virtual sequences, respectively. Each of them is instantiated by the meta-distances used by the corresponding NN classifiers, respectively. For RVSML, we set the hyper-parameters m and β via cross-validation by randomly selecting 30% of the training sequences to form a held-out validation set. We retrain RVSML with the selected hyper-parameters using all training sequences. For RVSML-LM, we fix the only hyper-parameter β to $1e-5$ on all datasets. For RVSML-BC, we fix the length per barycenter and β to 20 and 1e-5 on all datasets, respectively.

The comparisons on five datasets are presented in Tab. 1, Tab. 2, Tab. 3, Tab. 4, and Tab. 5, respectively. For all the methods, the MAPs are much lower than accuracies on the SAD dataset and the ChaLearn dataset. The MAP is computed by using each test sequence as a query to rank all training sequences and then taking mean of the average precisions of all test sequences. Some outlier test sequences with very low APs may affect the final MAP. As can be observed from the results, MAP and accuracy are largely synchronized. When RVSML has higher accuracy than other methods, in most cases, it also has a higher MAP. Therefore, this does not mean overfitting.

On the ChaLearn and SAD datasets, RVSMLs instantiated by both distances generally outperform the corresponding baseline classifiers and other metric learning methods, respectively. RVSML is able to learn a discriminative ground metric that incorporates the holistic temporal dependencies of sequences and enhances different meta-distances consistently. In some cases, several conventional metric learning methods obtain worse results than the baseline classifiers. This may indicate that temporal information is inherent for sequence data and cannot be discarded.

On the Action3D dataset with the DTW distance and the HAS dataset, RVSML performs inferior to LDMLT, but generally outperforms other metric learning methods. LDMLT is based on the dynamic triplet constraints, cannot ensure the convergence, and requires much more time for training. The training times of different metric learning methods for sequences on four datasets are shown in Tab. 6. We can observe that RVSML trains much faster compared with these methods. Specifically, the training time of LDMLT is more than ten times the training time of RVSML instantiated by DTW on most datasets, the training of SWMD is also at least 5 times slower than RVSML.

RVSML-LM and RVSML-BC outperform RVSML on the small-scale Action3D dataset. LM-based and BC-based virtual sequences lie in the same space with the original sequences. On this dataset, applying the NN classifiers to the original sequences obtain relatively high MAPs. This indicates that the within-class distributions of original sequences are relatively concentrated and hence the LM-based and BC-based virtual sequences from different classes are well separated. Pushing sequences towards their associated virtual sequences tunes the distributions of different classes and increases their margins. TS-based virtual sequences locate in a different space whose dimension depends on m. The desired class distributions differ greatly from those in the original space. Consequently, a few training sequences may not be sufficient to learn a reliable mapping to bridge the space gap.

On other datasets, the TS-based approach generally

TABLE 3
Comparison of RVSML instantiated by (left) DTW and (right) OPW with other methods using the NN classifier with the (left) DTW and (right) OPW distance on the ChaLearn dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
Ori [50]	11.75	61.12	12.21	59.38
ITML [5]	13.46	52.17	13.92	64.71
LMNN [7]	11.67	63.78	12.07	62.83
RVML [9]	31.21	83.79	30.19	80.66
LDMLT [33]	21.30	84.37	21.56	82.74
SWMD [23]	14.39	64.45	15.36	60.31
RVSML	**33.83**	**87.38**	**33.07**	**83.82**
RVSML-LM	19.47	71.16	18.34	57.21
RVSML-BC	23.20	64.65	24.91	65.34

TABLE 5
Comparison of the proposed RVSML instantiated by (left) DTW and (right) OPW with other metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the HAS dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
Ori[b] [50]	48.87 (1.09)	86.95 (2.89)	49.59 (1.10)	86.65 (3.20)
ITML [5]	14.50 (1.58)	48.90 (3.69)	62.01 (1.05)	92.20 (2.02)
LMNN [7]	60.94 (1.08)	92.34 (1.88)	17.72 (2.72)	48.40 (6.46)
SCML [16]	45.85 (10.62)	80.82 (10.93)	48.34 (2.27)	82.31 (3.54)
RVML [9]	74.21 (1.45)	94.82 (2.07)	70.24 (1.39)	93.77 (3.21)
LDMLT [33]	**82.80** (1.28)	**96.60** (0.82)	**79.92** (0.99)	**95.73** (1.11)
SWMD [23]	47.16 (3.74)	85.05 (4.68)	41.99 (2.38)	79.22 (2.81)
RVSML	74.64 (1.47)	95.65 (2.01)	71.95 (1.17)	94.11 (2.46)
RVSML-LM	60.96 (1.41)	89.66 (2.14)	61.74 (1.37)	88.78 (2.90)
RVSML-BC	62.59 (1.30)	90.55 (1.01)	65.18 (0.82)	90.27 (1.97)

[b] In the supplementary file of [10], σ is set to 1. In this paper, we set σ to 12 following [50], this leads to improved performances.

TABLE 6
Comparison of the training times.

Dataset	Action3D	SAD	ChaLearn	HAS
LDMLT	1905.24	67329.29	213921.7	10863.32 (247.0112)
SWMD	970.70	7756.72	11489.10	1497.786 (65.3938)
RVSML(DTW)	115.72	662.41	2477.76	212.3670 (32.0198)
RVSML(OPW)	124.48	208.67	836.67	150.2897 (8.7143)

TABLE 4
Comparison of RVSML instantiated by (left) DTW and (right) OPW with other metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the SAD dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
Ori [50]	56.58	96.36	59.77	96.36
ITML [5]	51.13	95.55	54.51	96.36
LMNN [7]	56.25	96.00	59.33	96.27
SCML [16]	47.98	93.27	50.08	94.50
RVML [9]	57.94	**96.59**	60.71	95.77
LDMLT [33]	59.54	96.50	61.07	96.73
SWMD [23]	52.44	93.95	58.00	95.41
RVSML	**60.24**	96.23	**65.63**	**97.09**
RVSML-LM	56.06	95.95	58.43	94.95
RVSML-BC	57.78	95.41	55.22	92.86

outperforms the other two approaches. Among them, on the SAD dataset, original sequences obtain relatively high MAPs and the performances of RVSML-LM and RVSML-BC are comparable with those of RVSML. On the ChaLearn dataset, MAPs of original sequences are very low, thus the LM-based and BC-based virtual sequences for different classes may be close and unevenly distributed in the original space, resulting in poor performances of RVSML-LM and RVSML-BC. In contrast, with sufficient training sequences, RVSML can map sequences into a different space in which well separated virtual sequences induce good separability among different classes.

RVSML-BC generally achieves higher MAP than RVSML-LM, while RVSML-LM often obtains higher top-1 accuracy. For the LM-based approach, each class may have multiple virtual sequences. Sequences from the same classes are drawn towards their nearest virtual sequences, respectively. For the BC-based approach, all sequences of the same class are pushed towards the barycenter. Therefore, the within-class variances are reduced, which is conducive to improving MAP.

6.4 Comparison with deep metric learning methods

We compare the proposed Deep-RVSML with seven deep metric learning methods using different losses, including NCA loss (NCA) [37], contrastive loss (Contrastive) [38], binomial deviance loss (Binomial) [34], lifted structured loss (Lifted) [35], triplet loss with hard-mining (Hard-Mining) [65], triplet loss with semi-hard mining (Semi-Hard) [42], multi-similarity loss(MS) [43]. These methods are developed for independent static data. To apply these methods to sequence samples, we take all vectors in sequences from a class as independent vector samples of this class and employ these losses with a deep encoder network which shares the same architecture with the encoder of Deep-RVSML. Batch normalization is performed before each layer and L_2 normalization is applied to the output embedding. The number of neurons in all the three hidden layers is set to 1024 and the embedding dimension is set to be the same as the original dimension of vectors in sequences. We adapt the code in [43] to implement these deep metric learning methods by replacing the convolutional neural network with the encoder.

For Deep-RVSML, we set the length m of TS-based virtual sequences to 4 on all datasets. On the MSR Action3D, MSR Activity3D, and SAD datasets, since the frame-wide features are non-normalized, we also apply batch normalization and L_2 normalization as done in competitive deep metric learning models. Deep-RVSML-LM does not introduce hyper-parameters. For Deep-RVSML-BC, we set the length per barycenter to 20 on all datasets. For all the methods, the NN classifiers with DTW and OPW distances are used to classify the encoded sequences, respectively. Other experimental settings remain the same as in Sec. 6.3.

The comparisons on five datasets are presented in Tab. 7, Tab. 8, Tab. 9, Tab. 10, and Tab. 11, respectively. On the MSR Activity3D dataset, Deep-RVSML performs inferior to other deep metric learning methods. This dataset has fewer training sequences, which may not be sufficient for Deep-RVSML to capture the temporal structures since Deep-

TABLE 7
Comparison of the proposed Deep-RVSML variants instantiated by (left) DTW and (right) OPW with other deep metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the MSR Action3D dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
NCA [37]	37.84	67.03	38.53	65.20
Contrastive [38]	45.91	60.44	47.46	62.64
Binomial [34]	48.12	60.81	49.99	62.64
Lifted [35]	43.96	67.40	50.42	65.20
HardMining [65]	43.19	57.51	43.82	55.31
SemiHard [42]	46.64	64.47	47.98	62.64
MS [43]	34.40	45.42	34.19	40.66
Deep-RVSML	61.73	79.49	69.14	74.36
Deep-RVSML-LM	65.76	**85.35**	62.16	84.25
Deep-RVSML-BC	**78.99**	**85.35**	**72.90**	**86.08**

TABLE 8
Comparison of the proposed Deep-RVSML variants instantiated by (left) DTW and (right) OPW with other deep metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the MSR Activity3D dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
NCA [37]	34.24	64.38	37.72	**66.87**
Contrastive [38]	**59.19**	65.00	62.73	66.25
Binomial [34]	56.76	62.50	61.56	60.62
Lifted [35]	48.39	62.50	59.57	65.62
HardMining [65]	58.85	**65.62**	61.33	63.12
SemiHard [42]	58.14	**65.62**	61.84	63.12
MS [43]	47.91	51.25	47.97	50.00
Deep-RVSML	53.07	61.88	**68.86**	65.62
Deep-RVSML-LM	46.95	64.38	48.85	60.00
Deep-RVSML-BC	53.85	64.38	59.16	52.50

RVSML views each sequence as a single sample. The within-class variances can not be fully reflected so that most classes may be distinguished only by the differences between their frames. In contrary, other methods have relatively more training data because all vectors of each sequence are used as independent training samples.

On all other datasets, the proposed Deep-RVSML outperforms all these deep metric learning methods significantly by using both DTW and OPW as the meta-distance measure. In many cases, some deep metric learning methods perform even worse than conventional metric learning methods evaluated in Sec. 6.3. Since elements in sequences violate the i.i.d. assumption, the stronger the fitting ability of the model, the more the loss of temporal information, the more serious the overfitting, and the worse the performance.

TABLE 9
Comparison of the proposed Deep-RVSML variants instantiated by (left) DTW and (right) OPW with other deep metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the ChaLearn dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
NCA [37]	9.07	62.77	9.25	63.38
Contrastive [38]	17.01	68.18	18.85	68.24
Binomial [34]	18.19	69.72	19.92	69.17
Lifted [35]	12.98	66.85	14.56	67.66
HardMining [65]	23.72	67.28	25.59	68.41
SemiHard [42]	15.81	67.37	17.45	66.62
MS [43]	18.14	65.95	19.67	63.67
Deep-RVSML	**43.96**	**81.15**	**46.28**	**79.91**
Deep-RVSML-LM	18.31	58.86	20.49	56.37
Deep-RVSML-BC	25.13	54.14	28.75	57.96

TABLE 10
Comparison of the proposed Deep-RVSML variants instantiated by (left) DTW and (right) OPW with other deep metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the SAD dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
NCA [37]	14.47	46.73	15.46	50.27
Contrastive [38]	59.69	92.09	67.40	95.45
Binomial [34]	48.20	91.73	60.22	95.77
Lifted [35]	46.41	89.64	54.80	94.86
HardMining [65]	65.56	92.68	73.80	95.95
SemiHard [42]	40.66	84.82	49.08	93.73
MS [43]	58.00	85.64	63.11	92.27
Deep-RVSML	**78.24**	97.32	**83.08**	**98.91**
Deep-RVSML-LM	59.49	97.73	68.36	98.09
Deep-RVSML-BC	77.04	**98.09**	80.80	98.05

In comparison with the results of RVSML in Sec. 6.3, we observe that Deep-RVSML achieves much better MAPs and comparable accuracies. Since the objective is to minimize the average meta-distance among all training-virtual sequence pairs, this only requires that sequences from the same class are more gathered around their virtual sequence. For any particular sequence, sequences from the same class are closer on the whole, but the nearest sequence is not necessarily in the same class. Due to the better fitting capacity, compared with RVSML, the meta-distance learned by Deep-RVSML better optimizes the objective. Therefore, by using a test sequence as a probe to retrieval all the gallery sequences with the learned meta-distance, as a whole, sequences from the same class as the probe get better rankings, resulting in higher MAP, but the top-1 accuracy may not be improved.

For different virtual sequence generation approaches, similar observations can be concluded as in the linear case in Sec. 6.3. By improving the structure of the encoder and employing other nonlinear activations, the performances of Deep-RVSML may be further improved.

6.5 Combination with state-of-the-art methods

The proposed RVSML learns a transformation that projects the sequences into another space. In the resulting space, we can use other advanced classification methods instead of the NN classifier. That is, we first apply the proposed RVSML to the original sequences and then employ state-of-the-art classification methods by taking the transformed sequences as input. In this way, the proposed RVSML can be combined with these methods.

We combine RVSML with kernelized-COV [66], which extracts the kernelized covariance representation from each sequence and applies SVM for classification. We instantiate RVSML with OPW, because OPW generates soft alignment, which preserves more local variances between element pairs so that covariance-based representation can capture more discriminative information. In [66], the 120-dimensional velocity and acceleration of the raw joint positions based frame-wide features [67] were employed. On the MSR Activity3D dataset, the pre-computed features are provided and hence we directly apply RVSML to them. On the MSR Action3D dataset, we compute the features following [67],

TABLE 11
Comparison of the proposed Deep-RVSML instantiated by (left) DTW and (right) OPW with other deep metric learning methods using the NN classifier with the (left) DTW and (right) OPW distance on the HAS dataset.

Method	DTW		OPW	
	MAP	Accuracy	MAP	Accuracy
NCA [37]	19.04 (5.39)	60.50 (8.15)	16.41 (4.72)	55.10 (8.76)
Contrastive [38]	24.77 (2.57)	59.64 (5.65)	24.40 (2.62)	59.28 (7.28)
Binomial [34]	22.02 (1.34)	57.52 (5.84)	21.57 (1.44)	57.11 (5.51)
Lifted [35]	32.38 (2.10)	80.10 (4.18)	29.22 (1.57)	75.31 (3.35)
HardMining [65]	15.73 (2.13)	37.92 (4.78)	15.89 (2.33)	39.36 (4.99)
SemiHard [42]	32.77 (2.34)	71.36 (3.41)	31.73 (2.26)	69.56 (4.03)
MS [43]	29.76 (4.16)	74.46 (10.53)	26.29 (3.50)	67.75 (10.15)
Deep-RVSML	**96.15** (0.73)	**98.81** (0.69)	**91.78** (1.52)	**98.22** (0.93)
Deep-RVSML-LM	75.06 (2.49)	94.48 (1.05)	56.84 (1.81)	83.70 (3.56)
Deep-RVSML-BC	83.54 (0.78)	95.93 (1.01)	69.44 (0.69)	91.61 (2.38)

TABLE 12
Comparison with state-of-the-art methods on the MSR Activity3D dataset.

Method	Accuracy
Actionlet Ensemble [57]	85.8%
Moving Pose [67]	73.8%
COV-$J_{\mathcal{H}}$-SVM [68]	75.5%
Ker-RP-POL [69]	96.9%
Ker-RP-RBF [69]	96.3%
Kernelized-COV [66]	96.3%
Luo et al. [70]	86.9%
Ji et al. [71]	81.3%
DSSCA SSLM [72]	97.5%
RVSML-DTW+Kernelized-COV	96.9%
RVSML-OPW+Kernelized-COV	97.5%
RVSML-OPW-Mar+Kernelized-COV	97.5%
RVSML-OPW-Bar+Kernelized-COV	97.5%
DeepRVSML-DTW+NN	**98.1%**
DeepRVSML-OPW+NN	97.5%

TABLE 13
Comparison with state-of-the-art methods on the MSR Action3D dataset.

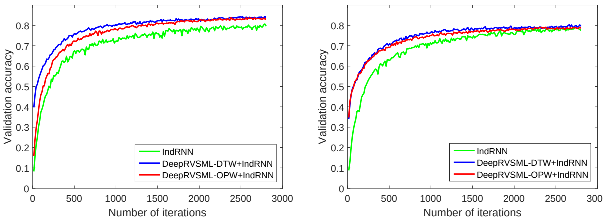
Method	Accuracy
Actionlet Ensemble [57]	88.2%
Moving Pose [67]	91.7%
COV-$J_{\mathcal{H}}$-SVM [68]	80.4%
Ker-RP-POL [69]	96.2%
Ker-RP-RBF [69]	**96.9%**
Kernelized-COV [66]	96.2%
SCK+DCK [74]	91.45%
TS-LSTM-GM [73]	91.21%
FTP-SVM [75]	90.01%
Bi-LSTM [75]	86.18%
RVSML-OPW+Kernelized-COV	**96.34%**
RVSML-OPW-Mar+Kernelized-COV	93.40%
RVSML-OPW-Bar+Kernelized-COV	88.64%
RVSML-DTW+TS-LSTM-GM	93.04%
RVSML-OPW+TS-LSTM-GM	90.48%

where the velocity and acceleration features are augmented by the raw joint positions. We perform Kernelized-COV to the transformed sequences. Tab. 12 and Tab. 13 show the results in comparison with the state-of-the-art methods on the two datasets, respectively. The combinations of RVSML with different virtual sequences and Kernelized-COV achieve comparable results with other competitors.

On the MSR Activity3D dataset, we also apply Deep-RVSML to the same features used by Kernelized-COV [66]. As shown in Tab. 12, a simple NN classifier in the non-linear metric spaces learned by DeepRVSML achieves better results than Kernelized-COV.

On the MSR Action3D dataset, we combine RVSML with the generalized temporal sliding LSTM (TS-LSTM) Network with the geometric mean [73] denoted by TS-LSTM-GM. We apply RVSML to the 60-dimensional motion features used in [73], perform L_2 normalization to the transformed features, and input the resulting sequences to TS-LSTM-GM. The results are shown in Tab. 13. The proposed RVSML instantiated by DTW improves the accuracy of TS-LSTM-GM by 1.8%. RVSML instantiated by different meta-distances fits for different classification methods.

On the NTU dataset, due to the large number of training sequences, linear RVSML and the NN classifier are too computationally intensive because they need to calculate the meta-distances of all training sequences to the corresponding virtual sequences or the test sequence. We use the batch-based DeepRVSML to learn the ground metric space and employ deep Independent Recurrent Neural Network (IndRNN) [76], [77] for classification. The hyper-parameters and settings of IndRNN remain the same as in [77]. The results in comparison with IndRNN and other RNN-based methods without data augmentation for both CS and CV settings are shown in Tab. 14. In [77], the results of IndRNN are obtained by preprocessing the skeleton data, but neither the preprocessing algorithm nor the preprocessed data are provided. "IndRNN*" indicates our reproduced results using the raw skeleton features. We observe that DeepRVSML improves the performances of IndRNN. Moreover, as shown in Fig. 5, IndRNN converges much faster in the non-linear ground metric space learned by DeepRVSML.



(a) (b)

Fig. 5. The frame-level validation accuracy of IndRNN as a function of the number of training iterations using the original frame-wide features and the transformed features by DeepRVSML-DTW and DeepRVSML-OPW on the NTU dataset for the (a) CS and (b) CV setting.

TABLE 14
Comparison with state-of-the-art methods on the NTU RGB+D dataset.

Method	CS	CV
PLSTM [63]	62.93%	70.27%
Clips+CNN+MTLN [78]	79.57%	84.83%
STA-LSTM [79]	73.40%	81.20%
ST-LSTM [80]	69.2%	77.7%
HCN [81]	86.5%	91.1%
TCN+TTN [29]	77.55%	84.25%
EleAtt-GRU [82]	79.8%	87.1%
TS-SAN [83]	**87.2%**	**92.7%**
ARRN-LSTM [84]	80.7%	88.8%
IndRNN [77]	84.88%	90.43%
IndRNN*	80.79%	87.14%
DeepRVSML-DTW + IndRNN	79.72%	86.68%
DeepRVSML-OPW + IndRNN	**83.20%**	**87.51%**

6.6 Evaluation on zero-shot classification

We evaluate DeepRVSML-ZS in the zero-shot sequence classification task on the NTU dataset. We follow the nearest split (NS) and furthest split (FS) settings in [85], where the top 5 classes with least and highest distances from other classes based on the normalized language embeddings are selected as unseen classes for testing, respectively, and the remaining 55 classes are used for training, respectively. For DeepRVSML-ZS, we perform the same preprocessing as in [85], [86] to the skeleton data. We concatenate all preprocessed joint locations of two subjects per frame to form 150-dimensional frame-wide features. Because it is difficult to establish a single frame with the semantic of the action class, we use a sliding window of 8 frames with a moving step of 4 frames to convert each action sequence into a sequence of 150×8 segments. For each segment, we apply a 1-D convolution layer with three 1×8 kernels and flatten their ReLu activations into a 150×3-dimensional vector. Finally, we use the encoder network with the same architecture as in DeepRVSML to transform the resulting vectors into the semantic space and perform L_2 normalization to the output embeddings. Since semantic words and visual frames may do not correspond in order, we use the Sinkhorn distance to instantiate DeepRVSML-ZS.

We also employ the 700-dimensional sentence embedding vectors of class descriptions used in [85] as virtual sequences. In this case, the length per virtual sequence is one and all encoded elements of a sequence are aligned to the corresponding embedding. This is equivalent to viewing these elements as independent vector samples of the same class. We denote this method by DeepRVSML-Vec. Tab. 15 shows the results. DeepRVSML-ZS-Sinkhorn outperforms DeepRVSML-Vec because it distinguishes different localities and establishes the correspondences among local visual segments and semantic compositions. In [85], spatio-temporal graph convolutional network is used to extract visual features from skeleton sequences, DeViSE [87] and RelationNet [88] are used to learn a projection or metric, and description embeddings of both seen and unseen classes are utilized for training. DeepRVSML-ZS only applies simple 1-D convolution and fully connected layers to the skeleton sequences. It does not require any information about unseen classes during training, while obtains comparable results with DeViSE in the FS setting.

TABLE 15
Evaluation of DeepRVSML-ZS for zero-shot sequence classification on the NTU RGB+D dataset.

Method	NS	FS
DeViSE [85]	75.16%	42.06%
RelationNet [85]	74.50%	50.06%
DeepRVSML-Vec	51.78%	40.33%
DeepRVSML-ZS-Sinkhorn	67.33%	42.62%

7 Conclusion

We present a metric learning framework for sequence data, which learns the meta-distance for sequences via learning the ground metric. The objective is to minimize the meta-distances between training sequences and their associated a prior defined virtual sequences. Constructing the meta-distance needs to infer the temporal alignments, but the inference also depends on the ground metric. We propose an efficient iterative solution to learn the ground metric and the latent alignments jointly. We unify a family of meta-distance measures for sequences into a common formulation and show that any meta-distance with such form can be employed to instantiate our framework. Additionally, we propose several approaches to generate virtual sequences. We empirically show that our method is able to enhance different types of meta-distances and state-of-the-art sequence classification methods.

Acknowledgments

The authors would like to thank the associate editor and anonymous reviewers for their valuable comments. This work was supported in part by the National Natural Science Foundation of China No. 61976206, No. 61832017, No. 61603373, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, Youth Innovation Promotion Association CAS No. 2019110, and National Science Foundation grant IIS-1619078, IIS-1815561.

References

[1] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv preprint arXiv:1306.6709*, 2013.

[2] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[3] B. Su and G. Hua, "Order-preserving wasserstein distance for sequence matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1049–1057.

[4] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 209–216.

[6] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *Advances in neural information processing systems*, 2004, pp. 41–48.

[7] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[8] P. C. Mahalanobis, "On the generalized distance in statistics." National Institute of Science of India, 1936.

[9] M. Perrot and A. Habrard, "Regressive virtual metric learning," in *Advances in Neural Information Processing Systems*, 2015, pp. 1810–1818.

[10] B. Su and Y. Wu, "Learning distance for sequences by learning a ground metric," in *International Conference on Machine Learning*, 2019, pp. 6015–6025.

[11] R. Jin, S. Wang, and Y. Zhou, "Regularized distance metric learning: Theory and algorithm," in *Advances in neural information processing systems*, 2009, pp. 862–870.

[12] G.-J. Qi, J. Tang, Z.-J. Zha, T.-S. Chua, and H.-J. Zhang, "An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 841–848.

[13] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, 2005, pp. 513–520.

[14] A. Globerson and S. T. Roweis, "Metric learning by collapsing classes," in *Advances in neural information processing systems*, 2006, pp. 451–458.

[15] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, 2006, pp. 1473–1480.

[16] Y. Shi, A. Bellet, and F. Sha, "Sparse compositional metric learning." in *AAAI*, 2014, pp. 2078–2084.

[17] A. Bellet, A. Habrard, and M. Sebban, "Learning good edit similarities with generalization guarantees," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 188–203.

[18] ——, "Good edit similarity learning by loss minimization," *Machine Learning*, vol. 89, no. 1-2, pp. 5–35, 2012.

[19] B. Paaßen, C. Gallicchio, A. Micheli, and B. Hammer, "Tree edit distance learning via adaptive symbol embeddings," in *International Conference on Machine Learning*, 2018.

[20] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning sequence kernels," in *2008 IEEE Workshop on Machine Learning for Signal Processing*. IEEE, 2008, pp. 2–8.

[21] C. Cortes, P. Haffner, and M. Mohri, "Rational kernels: Theory and algorithms," *Journal of Machine Learning Research*, vol. 5, no. Aug, pp. 1035–1062, 2004.

[22] M. Cuturi and D. Avis, "Ground metric learning," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 533–564, 2014.

[23] G. Huang, C. Guo, M. J. Kusner, Y. Sun, F. Sha, and K. Q. Weinberger, "Supervised word mover's distance," in *Advances in Neural Information Processing Systems*, 2016, pp. 4862–4870.

[24] F. Zhou and F. Torre, "Canonical time warping for alignment of human behavior," in *Proc. Advances in Neural Information Processing Systems*, 2009, pp. 2286–2294.

[25] F. Zhou and F. De la Torre, "Generalized time warping for multimodal alignment of human motion," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012, pp. 1282–1289.

[26] F. Zhou and F. De la Torre, "Generalized canonical time warping," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 279–294, 2016.

[27] G. Trigeorgis, M. A. Nicolaou, S. Zafeiriou, and B. W. Schuller, "Deep canonical time warping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5110–5118.

[28] G. Trigeorgis, M. A. Nicolaou, B. W. Schuller, and S. Zafeiriou, "Deep canonical time warping for simultaneous alignment and representation learning of sequences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1128–1138, 2017.

[29] S. Lohit, Q. Wang, and P. Turaga, "Temporal transformer networks: Joint learning of invariant and discriminative time warping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 426–12 435.

[30] D. Garreau, R. Lajugie, S. Arlot, and F. Bach, "Metric learning for temporal sequence alignment," in *Advances in neural information processing systems*, 2014, pp. 1817–1825.

[31] J. Zhao, Z. Xi, and L. Itti, "metricdtw: local distance metric learning in dynamic time warping," *arXiv preprint arXiv:1606.03628*, 2016.

[32] J. Mei, M. Liu, Y.-F. Wang, and H. Gao, "Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification," *IEEE transactions on Cybernetics*, vol. 46, no. 6, pp. 1363–1374, 2016.

[33] J. Mei, M. Liu, H. R. Karimi, and H. Gao, "Logdet divergence-based metric learning with triplet constraints and its applications," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4920–4931, 2014.

[34] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *International Conference on Pattern Recognition*, 2014, pp. 34–39.

[35] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4004–4012.

[36] Z. Che, X. He, K. Xu, and Y. Liu, "Decade: A deep metric learning model for multivariate time series," in *3rd SIGKDD Workshop on mining and learning from time series*, 2017.

[37] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Artificial Intelligence and Statistics*, 2007, pp. 412–419.

[38] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[39] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[40] W. Ge, "Deep metric learning with hierarchical triplet loss," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 269–285.

[41] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems*, 2016, pp. 1857–1865.

[42] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[43] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5022–5030.

[44] J. Bayer, C. Osendorfer, and P. V. D. Smagt, "Learning sequence neighbourhood metrics," in *Proceedings of the 22Nd International Conference on Artificial Neural Networks and Machine Learning*, 2012, pp. 2638–2644.

[45] B. Mokbela, B. Paassen, F. M. Schleif, and B. Hammer, "Metric learning for sequences in relational lvq," *Neurocomputing*, vol. 169, pp. 306–322, 2015.

[46] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *Proc. SIAM Int. Conf. Data Mining*. Lake Buena Vista, Florida, 2004, pp. 11–22.

[47] B. Su, J. Zhou, X. Ding, and Y. Wu, "Unsupervised hierarchical dynamic parsing and encoding for action recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5784–5799, 2017.

[48] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[49] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in neural information processing systems*, 2013, pp. 2292–2300.

[50] B. Su and G. Hua, "Order-preserving optimal transport for distances between sequences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 12, pp. 2961–2974, 2019.

[51] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[52] B. Su and Y. Wu, "Learning low-dimensional temporal representations," in *International Conference on Machine Learning*, 2018, pp. 4768–4777.

[53] ——, "Learning low-dimensional temporal representations with latent alignments," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[54] B. Su, J. Zhou, and Y. Wu, "Order-preserving wasserstein discriminant analysis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9885–9894.

[55] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[56] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *IEEE Int'l Workshop on CVPR for Human Communicative Behavior Analysis*, 2010.

[57] J. Wang, Z. Liu, and Y. Wu, "Mining actionlet ensemble for action recognition with depth cameras," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012.

[58] J. Wang and Y. Wu, "Learning maximum margin temporal warping for action recognition," in *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[59] S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopes, I. Guyon, V. Athitsos, and H. Escalante, "Multi-modal gesture recognition challenge 2013: Dataset and results," in *Proceedings of the 15th ACM on International conference on multimodal interaction*. ACM, 2013, pp. 445–452.

[60] K. Bache and M. Lichman, *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

[61] M. W. Kadous, *Temporal classification: extending the classification paradigm to multivariate time series*. PhD Thesis (draft), School of Computer Science and Engineering, University of New South Wales, 2002.

[62] B. Su, X. Ding, H. Wang, and Y. Wu, "Discriminative dimensionality reduction for multi-dimensional sequences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 77–91, 2018.

[63] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1010–1019.

[64] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2015.

[65] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[66] J. Cavazza, A. Zunino, M. San Biagio, and V. Murino, "Kernelized covariance for action recognition," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 408–413.

[67] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2752–2759.

[68] M. Harandi, M. Salzmann, and F. Porikli, "Bregman divergences for infinite dimensional covariance matrices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1003–1010.

[69] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li, "Beyond covariance: Feature representation with nonlinear kernel matrices," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4570–4578.

[70] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2203–2212.

[71] X. Ji, J. Cheng, W. Feng, and D. Tao, "Skeleton embedded motion body partition for human action recognition using depth sequences," *Signal Processing*, vol. 143, pp. 56–68, 2018.

[72] A. Shahroudy, T.-T. Ng, Y. Gong, and G. Wang, "Deep multimodal feature analysis for action recognition in rgb+ d videos," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1045–1058, 2018.

[73] I. Lee, D. Kim, S. Kang, and S. Lee, "Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 1012–1020.

[74] P. Koniusz, A. Cherian, and F. Porikli, "Tensor representations via kernel linearization for action recognition from 3d skeletons," in *European Conference on Computer Vision*. Springer, 2016, pp. 37–53.

[75] A. Ben Tanfous, H. Drira, and B. Ben Amor, "Coding kendall's shape trajectories for 3d action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2840–2849.

[76] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5457–5466.

[77] S. Li, W. Li, C. Cook, Y. Gao, and C. Zhu, "Deep independently recurrent neural network (indrnn)," *arXiv preprint arXiv:1910.06251*, 2019.

[78] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3d action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3288–3297.

[79] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[80] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, "Skeleton-based action recognition using spatio-temporal lstm network with trust gates," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 3007–3021, 2017.

[81] C. Li, Q. Zhong, D. Xie, and S. Pu, "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation," *arXiv preprint arXiv:1804.06055*, 2018.

[82] P. Zhang, J. Xue, C. Lan, W. Zeng, Z. Gao, and N. Zheng, "Eleatt-rnn: Adding attentiveness to neurons in recurrent neural networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 1061–1073, 2019.

[83] S. Cho, M. H. Maqbool, F. Liu, and H. Foroosh, "Self-attention network for skeleton-based human action recognition," *arXiv preprint arXiv:1912.08435*, 2019.

[84] W. Zheng, L. Li, Z. Zhang, Y. Huang, and L. Wang, "Relational network for skeleton-based action recognition," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 826–831.

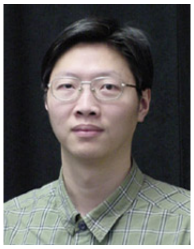
[85] B. Jasani and A. Mazagonwalla, "Skeleton based zero shot action recognition in joint pose-language semantic space," *arXiv preprint arXiv:1911.11344*, 2019.

[86] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 026–12 035.

[87] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *Advances in neural information processing systems*, 2013, pp. 2121–2129.

[88] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208.

Bing Su received the B.S. degree in information engineering from Beijing Institute of Technology, Beijing, in 2010, and the Ph.D. degree in Electronic Engineering from Tsinghua University, Beijing, in 2016. From 2016 to 2020, he worked at Institute of Software, Chinese Academy of Sciences, Beijing. Currently, he is an associate professor with the Gaoling School of Artificial Intelligence, Renmin University of China. His research interests include pattern recognition, computer vision, and machine learning.



Ying Wu received the B.S. from Huazhong University of Science and Technology, Wuhan, China, in 1994, the M.S. from Tsinghua University, Beijing, China, in 1997, and the Ph.D. in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC), Urbana, Illinois, in 2001.

From 1997 to 2001, he was a research assistant at the Beckman Institute for Advanced Science and Technology at UIUC. During summer 1999 and 2000, he was a research intern with Microsoft Research, Redmond, Washington. In 2001, he joined the Department of Electrical and Computer Engineering at Northwestern University, Evanston, Illinois, as an assistant professor. He was promoted to associate professor in 2007 and full professor in 2012. He is currently a full professor of Electrical and Computer Engineering at Northwestern University. His current research interests include computer vision, robotics, image and video analysis, pattern recognition, machine learning, multimedia data mining, and human-computer interaction.

He serves as associate editors for IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Image Processing, IEEE Transactions on Circuits and Systems for Video Technology, SPIE Journal of Electronic Imaging, and IAPR Journal of Machine Vision and Applications. He received the Robert T. Chien Award at UIUC in 2001, and the NSF CAREER award in 2003. He is a Fellow of the IEEE.