Deep r-th Root of Rank Supervised Joint Binary Embedding for Multivariate Time Series Retrieval

Dongjin Song[†], Ning Xia[†], Wei Cheng[†], Haifeng Chen[†], Dacheng Tao[‡]

[†]NEC Laboratories America, Inc.
[‡]UBTECH Sydney AI Centre, SIT, FEIT, University of Sydney
{dsong,nxia,weicheng,haifeng}@nec-labs.com,dacheng.tao@sydney.edu.au

ABSTRACT

Multivariate time series data are becoming increasingly common in numerous real world applications, *e.g.*, power plant monitoring, health care, wearable devices, automobile, *etc.* As a result, multivariate time series retrieval, *i.e.*, given the current multivariate time series segment, how to obtain its relevant time series segments in the historical data (or in the database), attracts significant amount of interest in many fields. Building such a system, however, is challenging since it requires a compact representation of the raw time series which can explicitly encode the temporal dynamics as well as the correlations (interactions) between different pairs of time series (sensors). Furthermore, it requires query efficiency and expects a returned ranking list with high precision on the top. Despite the fact that various approaches have been developed, few of them can jointly resolve these two challenges. To cope with this issue, in this paper we propose a Deep r-th root of Rank Supervised Joint Binary Embedding (Deep r-RSJBE) to perform multivariate time series retrieval. Given a raw multivariate time series segment, we employ Long Short-Term Memory (LSTM) units to encode the temporal dynamics and utilize Convolutional Neural Networks (CNNs) to encode the correlations (interactions) between different pairs of time series (sensors). Subsequently, a joint binary embedding is pursued to incorporate both the temporal dynamics and the correlations. Finally, we develop a novel r-th root ranking loss to optimize the precision at the top of a Hamming distance ranking list. Thoroughly empirical studies based upon three publicly available time series datasets demonstrate the effectiveness and the efficiency of Deep r-RSJBE.

KEYWORDS

Deep learning, multivariate time series retrieval, r-th root ranking loss, supervised binary embedding.

ACM Reference Format:
Dongjin Song[†], Ning Xia[†], Wei Cheng[†], Haifeng Chen[†], Dacheng Tao[‡]. 2018. Deep r-th Root of Rank Supervised Joint Binary Embedding for Multivariate Time Series Retrieval. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220108

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220108

1 INTRODUCTION

In recent years, multivariate time series data is increasingly generated in numerous real world applications. For instance, in a power plant [32], a large number of sensors can be deployed to monitor the operation status in real time. In the field of health care [20, 21], electroencephalography (EEG) utilizes multiple electrodes to record and analyze brain activities. For a fitness tracking device, multiple sensors are employed to detect a temporal sequence of actions [31], *e.g.*, walking for 5 minutes, running for 1 hour, and then sitting for 15 minutes. Given the huge amount of historical (multivariate) time series data in a system, how to interpret the current status becomes an important problem to investigate.

For this purpose, we formulate it as a supervised multivariate time series retrieval problem. Specifically, given the current multivariate time series segment, *i.e.*, a slice of multivariate time series which lasts for a short period of time, we aim to find its most similar time series segments in the historical data (or database). Assuming that label information (*e.g.*, walking, running, sitting) is available in the historical data (or database), it will be straightforward to interpret the current system status.

A key challenge to build such a system is to obtain a good representation for multivariate time series segments. Previous studies [17, 19] suggests that besides the temporal dynamics in the raw multivariate time series segments, the correlations (interactions) between different pairs of time series (sensors) are also essential to characterize the system status. Therefore, a good representation refers to a compact abstraction which can explicitly encode the temporal dynamics of the raw time series segment as well as the correlations (interactions) between different pairs of time series (sensors). Over the past few decades, a number of approaches have been developed to denote a time series segment, *e.g.*, Discrete Fourier Transform (DCT) [12, 39], Discrete Wavelet Transform (DWT) [7], Piecewise Aggregate Approximation (PAA) [22], *etc.* Most of these approaches, however, focus on univariate time series representation and ignore the correlations between different pairs. In addition, almost all these representations are obtained based on human prior knowledge and hence could be suboptimal for multivariate time series retrieval giving the fact that their objectives and feature extraction are decoupled.

Another critical challenge is to obtain query results efficiently while maintain optimal precision at the top of the ranking list. Since

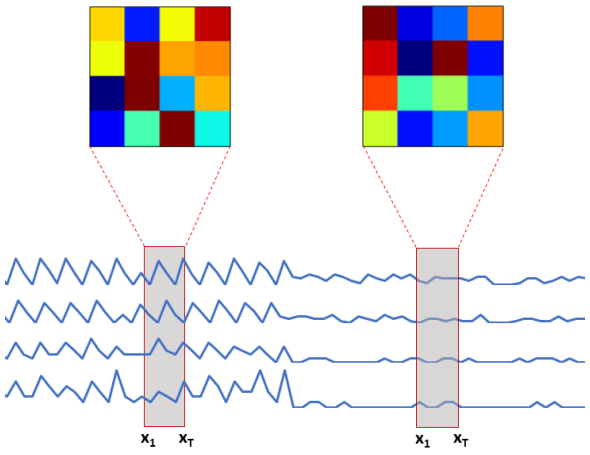


Figure 1: Raw multivariate time series (with 4 sensors) and correlation maps (4×4) of two time series segments. Because the system status is different at two time steps, the correlation maps are also different.

both the query efficiency and the ranking precision are partially determined by the similarity measure, a number of similarity measures have been developed. For instance, Dynamic Time Warping (DTW) [4, 34] and Edit Distance with Real Penalty (ERP) [8] employ dynamic programming to measure the similarity of two sequences. Since dynamic programming requires that each element of one time series to be compared with each element of the other, both DTW and ERP will be slow if the length of query segment is relatively large. To speed up the expensive similarity search, a number of branch and bound based pruning strategies [11, 24] can be utilized to quickly produce a superset of the desired results. These techniques, however, still require a refinement step (*e.g.*, based upon DTW) and may not work well when the length of query segment is relatively large. More recent advances [21, 29] suggest that hashing techniques, *e.g.*, Locality Sensitive Hashing (LSH) [1], Sketch, Single, & Hash (SSH) [29], can be employed to further reduce the query complexity of high-dimensional similarity search. These approaches, however, mainly focus on univariate time series and cannot guarantee satisfied precision at the top of the ranking list when label information is provided.

To resolve these two challenges, in this paper we propose a Deep r-th root of Rank Supervised Joint Binary Embedding (Deep r-RSJBE) to perform multivariate time series retrieval. Given a raw multivariate time series segment, we employ Long Short-Term Memory (LSTM) units [10, 16] to encode the temporal dynamics and utilize Convolutional Neural Networks (CNNs) [26] to encode the correlations (interactions) between different pairs of time series (sensors). In this way, both temporal dynamics and the correlations in the raw time series segment are explicitly represented with two separate feature vectors. Subsequently, a joint binary embedding is pursued to incorporate both the temporal dynamics and the correlations. With this embedding, the similarity between two multivariate time series segments can be measured in Hamming space which could be extremely efficient based upon assembly POPCOUNT instruction. Finally, we develop a novel r-th root ranking loss to train the disciplined embedding functions, by which the mistakes at the top of a Hamming-distance ranking list are penalized more than those at the bottom. This ranking loss can also enforce two misaligned (or different) segments to share similar binary embeddings as long as they belong to the same class. To obtain such embedding functions, we relax the original discrete objective with a continuous surrogate, and derive a stochastic gradient descent to optimize the surrogate objective. To the best of our knowledge, Deep r-RSJBE is the first end-to-end learning based approach for supervised multivariate time series retrieval which explicitly encodes the temporal dynamics in the raw time series as well as the correlations (interaction) between different pairs of time series (sensors). To demonstrate the effectiveness of Deep r-RSJBE, we conduct thorough empirical studies based upon three public available datasets, *i.e.*, EEG Eye State dataset, PAMAP2 dataset, and Sussex-Huawei Locomotion (SHL) dataset. Our experiment results show the effectiveness and the efficiency of the proposed Deep r-RSJBE.

2 RELATED WORK

This work relates to recent advances in time series representation, time series similarity measures, and binary embedding.

Univariate time series representation is a well developed field. Existing techniques can be divided into three main categories: temporal methods, spectral methods, and learning based methods. Temporal representations, *e.g.*, extrema extraction [13], bit-level representation [3], Piecewise Aggregate Approximation (PAA) [23, 50], Adaptive Piecewise Constant Approximation (APCA) [22], *etc.*, aim to encode the temporal structure of raw data. Spectral based methods, *e.g.*, Discrete Fourier Transform (DCT) [12, 39], Discrete Wavelet Transform (DWT) [7], Mel-Frequency Cepstral Coefficients (MFCC) [51], *etc.*, represent the raw data with frequency information. Learning based methods include principle component analysis (PCA), Hidden Markov Models (HMMs) [2], *etc.* Although most of these approaches can be extended to represent multivariate time series, few of them can explicitly encode the correlations between different pairs of time series (sensors). Furthermore, these representations are mainly obtained based upon human prior knowledge and thus could be suboptimal for multivariate time series retrieval since their objectives and feature extraction are decoupled.

Similarity measures between two univariate time series [4, 22, 23, 34, 37, 50] have been studied for decades. The purpose is to seek for a measure which is robust to noise and time shifting (misalignment). For this purpose, one type of methods are developed based upon the ℓ_1 and ℓ_2 norms. Examples include Dynamic Time Warping (DTW) [4, 34] and Edit Distance with Real Penalty (ERP) [8]. Another type of approaches are based upon a matching threshold. Examples of this class are the Longest Common Subsequence (LCSS) [46], and the Edit Distance on Real Sequence (EDR) [9]. These methods, however, are computational expensive due to the dynamic programming and will be slow if query segment is consists of a long sequence. Although a number of branch and bound based pruning strategies [11, 24] are utilized to quickly produce a superset of the desired results, these techniques still require a refinement

step (*e.g.*, based upon DTW) and may not work well when the length of query segment is relatively large. More recent trends [21, 29] suggest that hashing techniques, *e.g.*, Locality Sensitive Hashing (LSH) [1], Sketch, Single, & Hash (SSH) [29], can be employed to further reduce the query complexity of high-dimensional similarity search. These approaches, however, focus on univariate time series and cannot guarantee optimal precision at the top of a ranking list when label information is available. Our proposed Deep r-RSJBE leverages label information to learn binary embeddings so as to represent multivariate time series segments. In this way, even if two time series segments are misaligned, we can still obtain similar binary embeddings as long as they share the same label.

Our work is more closely related to binary embedding methods which include two main categories: data independent binary embedding approaches [1, 5] and data dependent (learning based) binary embedding techniques [15, 28, 30, 36, 41, 47]. In particular, learning based embeddings can be further categorized into unsupervised methods [15, 47] and supervised methods [28, 30, 36, 40, 42]. More recently, deep binary embedding methods [6, 27, 49] are becoming more popular since they achieve state-of-the-art performance for image retrieval task. Compared to these techniques, our proposed Deep r-RSJBE is unique from three perspectives: (1) rather than focusing on image retrieval, we consider multivariate time series retrieval problem in which it is essential to jointly consider the temporal dynamics of raw time series segment and the correlations between different pairs of time series; (2) a joint binary embedding is pursued to encode both the temporal dynamics and the correlations; (3) r-th root ranking loss is developed to learn the binary embedding such that the precision at the top of the Hamming distance ranking list is optimized.

3 DEEP r-TH ROOT OF RANK SUPERVISED JOINT BINARY EMBEDDING

In this section, we present Deep r-th root of Rank Supervised Joint Binary Embedding (Deep r-RSJBE). Specifically, we first state the problem we aim to study. Then we describe how to utilize LSTM units to encode a raw multivariate time series segment and how to employ CNNs to explicitly encode the correlations between different pairs of time series. Subsequently, we pursue a joint binary embedding to incorporate both the temporal dynamics in the raw time series segment and the correlations between different pairs. Finally, we present a r-th root ranking loss and introduce a detailed optimization procedure.

3.1 Problem Statement

We introduce some main notations used in the paper. Given a multivariate time series segment, *i.e.*, n time series with $\mathbf{X} = (\boldsymbol{x}^1, \cdots, \boldsymbol{x}^n)^\top = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T) \in \mathbb{R}^{n \times T}$, where T is the length of window size, we use $\boldsymbol{x}^k = (x_1^k, x_2^k, \cdot, x_T^k)^\top \in \mathbb{R}^T$ to represent a time series of length T and employ $\boldsymbol{x}_t = (x_t^1, x_t^2, \cdots, x_t^n)^\top \in \mathbb{R}^n$ to denote a vector of n input series at time t. In addition, we use $\|\cdot\|_{\mathrm{F}}$ to denote the Frobenius norm of matrices, and $\|\boldsymbol{x}\|_{\mathrm{H}}$ to represent the Hamming norm of a vector $\boldsymbol{x}$, which is defined as the number of nonzero entries in $\boldsymbol{x}$, *i.e.*, ℓ_0 norm. We use $\|\boldsymbol{x}\|_1$ to represent the ℓ_1 norm of vector $\boldsymbol{x}$, which is defined as the sum of absolute values of the entries in $\boldsymbol{x}$.

Given a multivariate time series segment $\mathbf{X}_q \in \mathbb{R}^{n \times T}$, *i.e.*, a slice of n time series which lasts T time steps, we aim to find its most similar time series segments in the historical data (or database) *i.e.*, we expect to obtain:

$$\arg \min_{\boldsymbol{X}_p \in \mathcal{D}} \mathcal{S}(\mathbf{X}_q, \mathbf{X}_p) \tag{1}$$

where $\mathcal{D} = \{\mathbf{X}_p\}$ is a collection of segments, p denotes the index for p-th segment ($\forall 1 \le p \le N$), N denotes the total number of segments in the collection, and $\mathcal{S}(\cdot)$ represents a similarity measure function. As long as the label information is available, it will be straightforward to interpret the current system status in $\mathbf{X}_q$ based upon the statuses of the top ranked candidates (*e.g.*, walking, running, sitting, etc.).

3.2 Raw Multivariate Time Series Segment Representation

To perform multivariate time series retrieval, it is essential to obtain a good representation for the multivariate raw time series segment which can capture the temporal dynamics. Given a multivariate time series segment $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T)$ with $\boldsymbol{x}_t \in \mathbb{R}^n$, where n is the number of time series, we aim to learn a mapping from $\boldsymbol{X}$ to $\boldsymbol{h}$ with

$$\boldsymbol{h} = \mathcal{F}(\mathbf{X}), \tag{2}$$

where $\boldsymbol{h} \in \mathbb{R}^m$ is the feature vector, m is the dimension of $\boldsymbol{h}$, and $\mathcal{F}$ is a non-linear mapping function.

In the past, LSTM units [10, 33, 44] have been widely applied to sequence to sequence learning in natural language processing and machine translation. The key idea of an LSTM unit is that the cell state sums activities over time, which can overcome the problem of vanishing gradients and better capture long-term dependencies of time series. Therefore, we employ LSTM units as $\mathcal{F}$ to capture the temporal dynamics as well as long-term dependencies in $\mathbf{X}$. In particular, each LSTM unit has a memory cell with the state $\boldsymbol{s}_t$ at time t. Access to the memory cell will be controlled by three sigmoid gates: forget gate $\boldsymbol{f}_t$, input gate $\boldsymbol{i}_t$ and output gate $\boldsymbol{o}_t$. The update of an LSTM unit can be summarized as follows:

$$\boldsymbol{f}_t = \sigma(\mathbf{W}_f[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_f) \tag{3}$$

$$\boldsymbol{i}_t = \sigma(\mathbf{W}_i[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{4}$$

$$\boldsymbol{o}_t = \sigma(\mathbf{W}_o[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{5}$$

$$\boldsymbol{s}_t = \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1} + \boldsymbol{i}_t \odot \tanh(\mathbf{W}_s[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_s) \tag{6}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{s}_t) \tag{7}$$

where $[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] \in \mathbb{R}^{m+n}$ is a concatenation of the previous hidden state $\boldsymbol{h}_{t-1}$ and the current input $\boldsymbol{x}_t$. $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_s \in \mathbb{R}^{m \times (m+n)}$, and $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_o$, $\boldsymbol{b}_s \in \mathbb{R}^m$ are parameters to learn. σ and $\odot$ are a logistic sigmoid function and an element-wise multiplication operator (*i.e.*, Hadamard product), respectively.

As shown in Figure 2, the last hidden state of LSTM units, *i.e.*, $\boldsymbol{h}_T$, is employed as the representation for a raw multivariate time series segment $\mathbf{X}$ since it encodes temporal dynamic information in the entire segment.

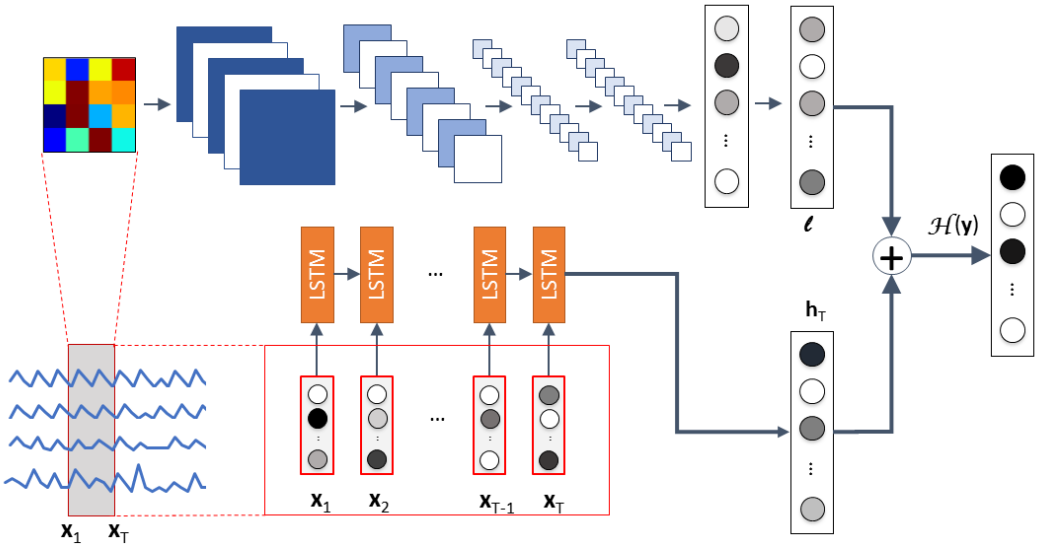


Figure 2: The network architecture for Deep r-RSJBE. At the bottom, a raw time series segment is encoded with LSTM units ($\boldsymbol{h}_T$). At the top, the correlation map of the raw time series segment is calculated and encoded via CNNs ($\boldsymbol{l}$). After two representations are obtained from the raw time series and the correlation map, a joint binary embedding ($\mathcal{H}(\boldsymbol{y})$) is learned under the supervision of the r-th root ranking loss.

3.3 Correlation Map Representation

Many previous studies [17, 19] suggests that besides the temporal dynamics in the raw multivariate time series segments, the correlations (interactions) between different pairs of time series (sensors) are also critical to characterize the system status. To represent the correlations between different pairs of time series in a multivariate time series segment, we construct a $n \times n$ correlation map based upon Pearson's correlation coefficient. Given two time series $\boldsymbol{x}^i = (x_1^i, x_2^i, \cdot, x_T^i)^\top \in \mathbb{R}^T$ and $\boldsymbol{x}^j = (x_1^j, x_2^j, \cdot, x_T^j)^\top \in \mathbb{R}^T$, their Pearson's correlation coefficient can be calculated with:

$$c_i^j = \frac{\sum_{k=1}^{T}(x_k^i - \bar{x}^i)(x_k^j - \bar{x}^j)}{\sqrt{\sum_{k=1}^{T}(x_k^i - \bar{x}^i)^2 \sum_{k=1}^{T}(x_k^j - \bar{x}^j)^2}} \tag{8}$$

where $\bar{x}^i$ and $\bar{x}^j$ denotes sample means of the two time series.

In order to get a compact representation of the correlation map $\boldsymbol{C} \in \mathbb{R}^{n \times n}$, we adopt Convolutional Neural Networks (CNNs) which have similar architecture as AlexNet [26]. As shown in Figure 2, the CNNs in Deep r-RSJBE contain 4 convolutional layers (conv1-conv4 with $3 \times 3 \times 16$, $3 \times 3 \times 32$, $3 \times 3 \times 64$, and $3 \times 3 \times 64$ filters; as well as 1×1, 2×2, 2×2, and 1×1 strikes, respectively) and 2 fully connected layers (fc5-fc6). Each convolutional layer is a three-dimensional array of size $h \times w \times d$ followed by batch normalization and rectifier linear units (ReLU), *i.e.*, $\max(0, x)$, where h and w are spatial dimensions, and d is the feature or channel dimension. Fully connected layers (fc5-fc6) transforms convolutional feature maps to a vector and project it to a fixed dimension m. Note that more sophisticated networks, *e.g.*, VGGNet [38] or ResNet [18], could be used when dealing with more complex tasks.

As shown in Figure 2, the output of fc6, *i.e.*, $\boldsymbol{l} \in \mathbb{R}^m$, is used to encode the correlations between different pairs of time series in a multivariate time series segment.

3.4 Jointly Binary Embedding

Given the representation for a raw multivariate time series segment ($\boldsymbol{h}_T$) as well as the representation for the correlations between different pairs of time series in the same segment ($\boldsymbol{l}$), we concatenate them together as $\boldsymbol{y} = [\boldsymbol{h}_T; \boldsymbol{l}] \in \mathbb{R}^{2m}$ and aim to learn a joint binary embedding which comprises a group of mapping functions $\{\mathcal{H}_c(\boldsymbol{y})\}_{c=1}^{v}$ such that a $2m$-dimensional floating-point input $\boldsymbol{y} \in \mathbb{R}^{2m}$ is compressed into an v-bit binary code $[\mathcal{H}_1(\boldsymbol{y}), \cdots, \mathcal{H}_v(\boldsymbol{y})]^\top \in \mathbb{H}^v \equiv \{1, -1\}^v$. This mapping, known as binary embedding or hash function in the literature, is formulated by

$$\mathcal{H}_c(\boldsymbol{y}) = \mathrm{sgn}(\mathcal{F}_c(\boldsymbol{y})), \quad c = 1, \cdots, v, \tag{9}$$

where $\mathrm{sgn}(\cdot)$ is the sign function that returns 1 if input variable is greater than 0 and -1 otherwise, and $\mathcal{F}_c : \mathbb{R}^{2m} \mapsto \mathbb{R}$ is a proper prediction function. A variety of mathematical forms for $\mathcal{F}_c$ (*e.g.*, linear or nonlinear) can be utilized to serve to specific data domains and practical applications. In this work, we focus on using a linear prediction function, that is, $\mathcal{F}_c(\boldsymbol{y}) = \boldsymbol{w}_c^\top \boldsymbol{y} + b_c$ (where $\boldsymbol{w}_c \in \mathbb{R}^{2m}$ and $b_c \in \mathbb{R}$) for simplicity. Following the previous work [15, 28], we set the bias term $b_c = -\boldsymbol{w}_c^\top \boldsymbol{u}$ by using the mean vector $\boldsymbol{u} = \sum_{i=1}^{n} \boldsymbol{y}_i / n$, which will make each generated binary bit $\{\mathcal{H}_c(\boldsymbol{y}_i)\}_{c=1}^{v}$ for $c \in [1 : v]$ be nearly balanced and thus exhibit maximum entropy. For brevity, we further define the whole binary embedding function $\mathcal{H} : \mathbb{R}^{2m} \mapsto \mathbb{H}^v$ to comprise the functionality of v individual binary embedding functions $\{\mathcal{H}_c\}_{c=1}^{v}$, that is,

$$\mathcal{H}(\boldsymbol{y}, \mathbf{W}) = \mathrm{sgn}(\mathbf{W}^\top (\boldsymbol{y} - \boldsymbol{u})), \tag{10}$$

which is parameterized by a matrix $\mathbf{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_v] \in \mathbb{R}^{2m \times v}$. Note that Eq. (10) applies the sign function $\mathrm{sgn}(\cdot)$ in the element-wise way. For simplicity, we will abbreviate $\mathbf{W}$ and use $\mathcal{H}(\boldsymbol{y}) = \mathcal{H}(\boldsymbol{y}, \mathbf{W})$ in the following description.

3.5 r-th Root Ranking Loss

To pursue such a binary embedding function $\mathcal{H}(\cdot)$, rather than considering pairwise similarities (*i.e.*, pair-level labels) as in [27, 28], we leverage relative segment similarities in the form of triplets $\mathcal{D}_{\text{Triplet}} = \{(\mathbf{X}_q, \mathbf{X}_i, \mathbf{X}_j)\}$, in which the segment pair $(\mathbf{X}_q, \mathbf{X}_i)$ is more similar than the segment pair $(\mathbf{X}_q, \mathbf{X}_j)$ (*e.g.*, $\mathbf{X}_q$ and $\mathbf{X}_i$ belong to the same class while $\mathbf{X}_q$ and $\mathbf{X}_j$ belong to different classes). Intuitively, we would expect that these relative similar relationships revealed by $\mathcal{D}_{\text{Triplet}}$ can be preserved within the Hamming space by the virtue of a good binary embedding function $\mathcal{H}(\cdot)$, which makes the Hamming distance between the embeddings $\mathcal{H}(\boldsymbol{y}_q)$ and $\mathcal{H}(\boldsymbol{y}_i)$ smaller than that between the embeddings $\mathcal{H}(\boldsymbol{y}_q)$ and $\mathcal{H}(\boldsymbol{y}_j)$. Suppose that $\boldsymbol{y}_q$ denotes a query segment, $\boldsymbol{y}_i$ denotes its similar segment, and $\boldsymbol{y}_j$ represents its dissimilar segment. Then the "rank" of $\boldsymbol{y}_i$ with respect to the query $\boldsymbol{y}_q$ can be defined as the number of dissimilar segments $\boldsymbol{y}_j$ (when j varies) which are more closer to the query $\boldsymbol{y}_q$ than $\boldsymbol{y}_i$ within the projected Hamming space. Therefore, the "rank" can be defined as:

$$\mathrm{R}(\boldsymbol{y}_q, \boldsymbol{y}_i) = \sum_j \mathcal{I}\Big(\big\|\mathcal{H}(\boldsymbol{y}_q) - \mathcal{H}(\boldsymbol{y}_j)\big\|_{\mathrm{H}} \leq \big\|\mathcal{H}(\boldsymbol{y}_q) - \mathcal{H}(\boldsymbol{y}_i)\big\|_{\mathrm{H}}\Big), \tag{11}$$

where $\mathcal{I}(\cdot)$ is an indicator function which returns 1 if the condition in the parenthesis is satisfied and returns 0 otherwise. Intuitively, the function $\mathrm{R}(\boldsymbol{y}_q, \boldsymbol{y}_i)$ explicitly measures the number of the incorrectly ranked dissimilar segments $\boldsymbol{y}_j$'s which are closer to the query $\boldsymbol{y}_q$ than the similar segment $\boldsymbol{y}_i$ in terms of Hamming distance and therefore indicates the position of $\boldsymbol{y}_i$ in a Hamming distance ranking list with respect to the query $\boldsymbol{y}_q$. In order to explicitly optimize the search precision at top positions of a ranking list, we introduce the r-th root ranking loss as:

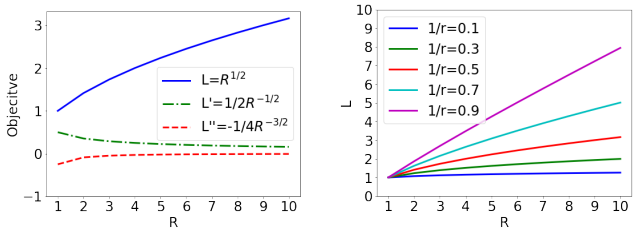
$$\mathcal{L}\big(\mathrm{R}(\boldsymbol{y}_q, \boldsymbol{y}_i)\big) = \sqrt[r]{\mathrm{R}(\boldsymbol{y}_q, \boldsymbol{y}_i)} = \mathrm{R}^{\frac{1}{r}}(\boldsymbol{y}_q, \boldsymbol{y}_i), \tag{12}$$

where $r > 1$. This loss penalizes the segments (*i.e.*, $\boldsymbol{q}_j$'s) that are incorrectly ranked at the top of a Hamming-distance ranking list more than those at the bottom. This is because the increment of $\mathcal{L}(\mathrm{R})$ gradually decays as R increases linearly. The detailed properties of the ranking loss are shown in Figure 3. As we can notice in Figure 3(a), $\mathcal{L}(\mathrm{R})$ is a one to one monotonic increasing function with first order derivative $\mathcal{L}'(\mathrm{R})$ large than zero and second order derivative $\mathcal{L}''(\mathrm{R})$ smaller than zero. Since $\mathcal{L}(\mathrm{R})$ can be seen as an integral of its gradient, intuitively, $\mathcal{L}'(\mathrm{R}) > 0$ preserves the rank by penalizing the "rank" (*i.e.*, R) we defined more severe at the top (*i.e.*, when R is small) than at the bottom (*i.e.*, when R is large).

Our Deep r-th RSJBE makes use of the above ranking loss and the learning objective is formulated as follows:

$$O(\mathcal{D}_{\text{Triplet}}, \mathbf{W}) = \sum_q \sum_i \mathrm{R}^{\frac{1}{r}}(\boldsymbol{y}_q, \boldsymbol{y}_i) + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{13}$$

where the first term is the proposed ranking loss, the second term enforces regularization, and $\lambda > 0$ is a positive parameter controlling the trade-off between the ranking loss and the regularization term. The parameter $r > 1$ determines to what degree the penalization should be put on top of the ranking list (when r becomes larger, the penalization becomes more severe). By optimizing this objective with respect to $\mathbf{W}$, we expect to optimize precision at top positions of a Hamming distance ranking list.



(a) (b)

Figure 3: Properties of the r-th root ranking loss. (a) Loss function $\mathcal{L}(R)$, its gradient $\mathcal{L}'(R)$, and $\mathcal{L}''(R)$ when $r = 2$. (b) $\mathcal{L}(R)$ vs r when $\frac{1}{r} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

We are aware that similar idea of our ranking loss in Eq. (12) was previously used for information retrieval [41, 45], link prediction [43] and image annotation [48]. Our work differs from them because (1) the relative similarity of the triplets are measured with Hamming distance which is discrete and discontinuous; (2) r-th ranking loss is continuous differentiable with respect to the "rank" while most of the previous ones are discrete and difficult to optimize; (3) r is a hyper-parameter which controls the degree of penalization over the top of the ranking list and makes the loss function more flexible compared to existing works.

3.6 Optimization

Although the ranking loss in Eq. (12) is continuous and differentiable with respect to R, our objective in Eq. (13) is still difficult to optimize. This is because: (a) the binary embedding function is a discrete mapping; and (b) the Hamming norm lies in a discrete space. Therefore, the proposed Deep r-RSJBE objective is discrete in nature and the associated optimization problem is combinatorially difficult.

To tackle this issue, we need relax the original discrete objective to a continuous and differentiable surrogate.

Specifically, the target binary embedding function $\mathcal{H}(\boldsymbol{y}) = \mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{y} - \boldsymbol{u})\big)$ can be relaxed as:

$$\overline{\mathcal{H}}(\boldsymbol{y}) = \tanh\big(\mathbf{W}^\top(\boldsymbol{y} - \boldsymbol{u})\big), \tag{14}$$

which is continuous and differentiable. $\tanh(\cdot)$ is a good approximation for $\mathrm{sgn}(\cdot)$ function because it transforms the value in the parenthesis to be in between of -1 and $+1$. Next, the Hamming norm in Eq. (11) is relaxed to ℓ_1 norm which is convex. Finally, we relax the indicator function in Eq. (11) with sigmoid function. Accordingly, Eq. (11) can be can be approximated with:

$$\begin{aligned} &\mathcal{I}\Big(\big\|\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_j)\big\|_1 \leq \big\|\overline{\mathcal{H}}(\boldsymbol{x}_q) - \overline{\mathcal{H}}(\boldsymbol{x}_i)\big\|_1\Big) \\ \approx &\sigma\Big(\big\|\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_i)\big\|_1 - \big\|\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_j)\big\|_1\Big) \end{aligned} \tag{15}$$

where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the sigmoid function.

Based upon these relaxations, the objective in Eq. (13) can be approximated with:

$$\overline{O(\mathcal{D}_{\text{Triplet}}, \mathbf{W})} = \sum_q \sum_i \overline{\mathrm{R}}^{\frac{1}{r}}(\boldsymbol{y}_q, \boldsymbol{y}_i) + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 \tag{16}$$

where $\overline{\mathrm{R}}(\boldsymbol{y}_q, \boldsymbol{y}_i)$ is a soft-approximated rank of $\boldsymbol{y}_i$ with respect to the query $\boldsymbol{y}_q$ which can be given by:

$$\overline{\mathrm{R}}(\boldsymbol{y}_q, \boldsymbol{y}_i) = \sum_j \sigma\Big(\mathrm{V}_{qi} - \mathrm{V}_{qj}\Big), \tag{17}$$

where V_{qi} is written as

$$\mathrm{V}_{qi} = \big\|\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_i)\big\|_1, \tag{18}$$

and V_{qj} is denoted as

$$\mathrm{V}_{qj} = \big\|\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_j)\big\|_1. \tag{19}$$

Although sub-gradient descent approach can be derived to optimize Eq. (16), it may converge slowly or even will be infeasible because of the expensive computation for the full gradient at each iteration. Therefore, a stochastic gradient descent method is derived to resolve this issue.

To optimize r-th root ranking loss with stochastic gradient descent algorithm, given a collection of triplets $\mathcal{D}_{\text{Triplet}}$, we first randomly select a query $\mathbf{X}_q$ and its similar segment $\mathbf{X}_i$. Then we fix $\mathbf{X}_q$ and $\mathbf{X}_i$, and randomly draw s ($s \leq M$) different $\mathbf{X}_j$'s when j varies to form a set of triplets $\{\mathbf{X}_q, \mathbf{X}_i, \mathbf{X}_j\}_{j=1}^s$. Note that M is the total number of possible choices of j. Assuming that the violated examples are uniformly distributed, then $\overline{\mathrm{R}}(\boldsymbol{y}_q, \boldsymbol{y}_i)$ can be approximated with $\lfloor \frac{M}{s} \rfloor \cdot \sum_{j=1}^s \sigma\Big(\mathrm{V}_{qi} - \mathrm{V}_{qj}\Big)$ where $\lfloor \cdot \rfloor$ is the floor function.

In this way, the objective of in Eq. (16) can be further approximated with:

$$\begin{aligned}&\overline{O(\mathcal{D}_{\text{Triplet}}, \mathbf{W})}\\ =&\Big(\Big\lfloor \frac{M}{s} \Big\rfloor \cdot \sum_{j=1}^{s} \sigma\Big(\mathrm{V}_{qi} - \mathrm{V}_{qj}\Big)\Big)^{\frac{1}{r}} + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2,\end{aligned} \tag{20}$$

and its associated gradient is given by:

$$\begin{aligned}&\frac{\partial \overline{O(\mathcal{D}_{\text{Triplet}}, \mathbf{W})}}{\partial \mathbf{W}} = \lambda \mathbf{W} +\\ &\frac{1}{r}\Big\lfloor \frac{M}{s} \Big\rfloor \Big[\sum_{j=1}^{s} \sigma\Big(\mathrm{V}_{qi} - \mathrm{V}_{qj}\Big)\Big]^{\frac{1}{r}-1} \sum_{j=1}^{s} \sigma\Big(\mathrm{V}_{qi} - \mathrm{V}_{qj}\Big)\sigma\Big(-\mathrm{V}_{qi} + \mathrm{V}_{qj}\Big)\cdot\\ &\Big\{(\boldsymbol{y}_q - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_i)\Big) \odot \Big(1 - \overline{\mathcal{H}}^2(\boldsymbol{y}_q)\Big)\Big]^\top\\ &-(\boldsymbol{y}_i - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_i)\Big) \odot \Big(1 - \overline{\mathcal{H}}^2(\boldsymbol{y}_i)\Big)\Big]^\top\\ &-(\boldsymbol{y}_q - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_j)\Big) \odot \Big(1 - \overline{\mathcal{H}}^2(\boldsymbol{y}_q)\Big)\Big]^\top\\ &+(\boldsymbol{y}_j - \boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\mathcal{H}}(\boldsymbol{y}_q) - \overline{\mathcal{H}}(\boldsymbol{y}_j)\Big) \odot \Big(1 - \overline{\mathcal{H}}^2(\boldsymbol{y}_j)\Big)\Big]^\top\Big\},\end{aligned} \tag{21}$$

where the $\odot$ denotes Hadamard product (*i.e.,* elementwise product).

Based upon this gradient, we can easily perform backpropagation over the entire network based upon minibatch stochastic gradient descent together with Adam optimizer [25] to optimize the network parameters of Deep r-RSJBE.

Algorithm 1: Optimization for Deep r-RSJBE

Input: $\mathcal{D}_{\text{Triplet}} = \big\{(\mathbf{X}_q, \mathbf{X}_i, \mathbf{X}_j)\big\}$, λ, m, r, v, max_iter
Output: Network parameters
for *iteration=1→max_iter* **do:**
  Randomly pick up a query $\mathbf{X}_q$;
  Randomly select a similar sample $\mathbf{X}_i$;
  Fixing $\mathbf{X}_q$ and $\mathbf{X}_i$, randomly select $s \leq M$ dissimilar samples $\mathbf{X}_j$ to form a batch of triplets $\{\mathbf{X}_q, \mathbf{X}_i, \mathbf{X}_j\}_{j=1}^s$;
  Calculate the gradient in Eq. (21) and perform back-propagation on the entire network;
end

Table 1: The statistics of three multivariate time series datasets.

Dataset	# of time series	Length	# of classes
EEG Eye State	14	14,980	2
PAMAP2	52	376,416	13
SHL	22	1,048,575	6

Although the ideal case is that within these s sampled triplets $\{\mathbf{X}_q, \mathbf{X}_i, \mathbf{X}_j\}_{j=1}^s$ at least one violation (*i.e.*, the ℓ_1 distance between $\mathcal{H}(\boldsymbol{y}_q)$ and $\mathcal{H}(\boldsymbol{y}_j)$ is smaller than that between $\mathcal{H}(\boldsymbol{y}_q)$ and $\mathcal{H}(\boldsymbol{y}_i)$) exists, in practical applications we find that stochastic gradient descent works well even when none violation exists in these s triplets. This is because $\overline{\mathrm{R}}(\boldsymbol{y}_q, \boldsymbol{y}_i)$ is a soft-approximated rank of $\boldsymbol{y}_i$ with respect to the query $\boldsymbol{y}_q$ and making a gradient update can still increase the margin in $\overline{\mathrm{R}}(\cdot)$. Therefore, it is flexible to set the batch size in Deep r-RSJBE. The detailed optimization procedure for Deep r-RSJBE is shown in Algorithm 1.

4 EXPERIMENTS

In this section, we first describe the three datasets we used for empirical studies. Then, we introduce the evaluation metrics and the parameter settings of Deep r-RSJBE. Finally, we compare the proposed Deep r-RSJBE against Euclidean distance and six different baseline methods, study its efficiency and parameter sensitivity.

4.1 Dataset

We consider three real world multivariate time series datasets as shown in Table 1, *i.e.*, EEG Eye State dataset, PAMAP2 dataset, and Sussex-Huawei Locomotion (SHL) dataset. In these three datasets, a discrete label is provided for each time step.

EEG Eye State dataset [1] is collected with the Emotiv EEG Neuroheadset. The eye state is detected via a camera during the EEG measurement. '1' indicates the eye-closed and '0' denotes the eye-open state. In our experiment, we uniformly sample 7,490 segments of length 5 and overlap 2. Among them, we randomly select 6,490 segments as the database for training, 500 segments as the validation set, and 500 segments as the test set.

[1] https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State

Table 2: Multivariate time series retrieval performance (MAP) on EEG Eye State, PAMAP2, and SHL when v = 32, 64, and 128. The best MAP is displayed in bold-face type.

Algorithms	EEG Eye State			PAMAP2			SHL		
♯ Bits	v =32	v =64	v =128	v = 32	v = 64	v = 128	v = 32	v = 64	v = 128
LSH [1]	0.5326	0.5335	0.5355	0.3354	0.3684	0.4030	0.4109	0.4319	0.4423
ITQ [15]	0.5435	0.5425	0.5439	0.3870	0.3849	0.4087	0.4200	0.4249	0.4316
HDML [30]	0.5646	0.5716	0.5714	0.5552	0.5919	0.6191	0.4619	0.4562	0.4578
TopRSBC [41]	0.5803	0.5832	0.5876	0.5758	0.6036	0.6149	0.4716	0.4724	0.4709
CNNs+Pairwise loss [27]	0.9567	0.9551	0.9569	0.6416	0.6327	0.6261	0.9008	0.9004	0.8928
LSTM+Triplet loss [10]	0.9570	0.9464	0.9433	0.6409	0.6662	0.6280	0.8892	0.8895	0.8451
LSTM+r-th root ranking loss	0.9805	0.9849	0.9663	0.6588	0.6779	0.6506	0.8986	0.8957	0.8652
Deep r-RSJBE	**0.9869**	**0.9855**	**0.9796**	**0.8381**	**0.8102**	**0.7948**	**0.9169**	**0.9369**	**0.9027**

The PAMAP2 is a physical activity monitoring dataset [2] [35] which contains data of various different physical activities (such as walking, cycling, playing soccer, *etc.*), performed by 9 subjects wearing 3 inertial measurement units (IMU) and a heart rate monitor. The sampling frequency for IMUs is 100Hz. In our study, we employ the raw data from Subject101 which contains 52 time series of 376,416 time steps. For this subject, the types of physical activities include: 'lying', 'sitting', 'standing', 'walking', 'running', 'cycling', 'Nordic walking', 'ascending stairs', 'descending stairs', 'vacuum cleaning', 'ironing', 'rope jumping', and 'others'. We uniformly select 75,283 segments with length 10 and overlap 5. Then we randomly pick up 71,528 segments as the database for training, 2,000 segments for validation, and 2,000 segments for test.

The Sussex-Huawei Locomotion (SHL) dataset [3] [14] is a versatile annotated dataset of modes of locomotion and transportation of mobile users. The dataset contains multi-modal data from a body-worn camera and from 4 smart phones, carried simultaneously at typical body locations. In our experiment, we select one subject's motion data (sampled at 100 Hz) that contains 1,048,575 time steps and 22 sensors for recording acceleration, gyroscope, magnetometer, orientation, *etc.* The types of motion include: 'null', 'still', 'walking', 'bike', 'train', and 'subway'. To perform multivariate time series retrieval, we uniformly sample 209,715 segments of length 10 and overlap 5. Among them, we randomly select 205,715 segments as the database for training, 2,000 segments for validation, and 2,000 segments for test.

The detailed information about the partitions of these three datasets are provided online [4].

4.2 Evaluation Metrics and Parameter Settings

To measure the effectiveness of various binary embedding techniques for multivariate time series retrieval, we consider three evaluation metrics, *i.e.*, Mean Average Precision (MAP), precision at top-k positions (Precision@k), and recall at top-k positions (Recall@k).

Deep r-RSJBE has 5 hyper-parameters. For simplicity, we fix the size of fc5 and fc6 in CNNs and set them as 256 on three datasets. Meanwhile, we set the hidden size of LSTM units as 64, 256, and 256 on EEG Eye State, PAMAP2, and SHL, respectively. For two hyper-parameters in r-th root ranking loss, λ and r, they are optimized based upon grid search over $\lambda = \{0.0001, 0.001, 0.01, 0.1, 1\}$ and $\frac{1}{r} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. To determine the optimal parameters for test, we conduct 5 trials on each parameter combination and the combination which achieves best average MAP on the validation set is utilized for test. In all our experiments, we fixed the batch size as 512 for simplicity. Deep r-RSJBE is implemented with TensorFlow and trained on a server with Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz and 4 NVIDIA GTX 1080 Ti graphics cards.

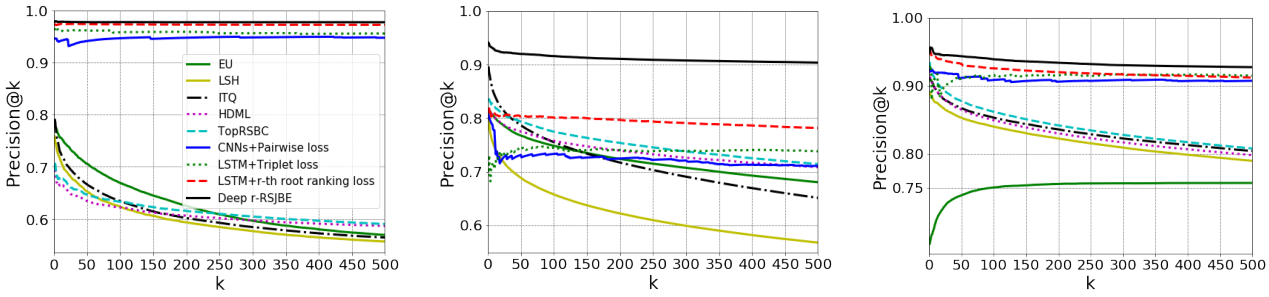
4.3 Multivariate Time Series Retrieval

In the experiments, we evaluate the effectiveness of the proposed Deep r-RSJBE for multivariate time series retrieval based upon three multivariate time series datasets.

For this purpose, we compare Deep r-RSJBE against Euclidean distance (EU) as well as six representative binary embedding and hashing algorithms. Among them, two are unsupervised methods, including one randomized method Locality-Sensitive Hashing (LSH) [1] and one linear projection method Iterative Quantization (ITQ) [15]. The other four are supervised algorithms, including two linear algorithms, *i.e.*, Hamming Distance Metric Learning (HDML) [30] and Top Rank Supervised Binary Coding (TopRSBC) [41], and two deep learning based algorithms, *i.e.*, CNNs+Pairwise loss [27] and LSTMs+Triplet loss [10]. For a fair comparison, the network architecture for CNNs in CNNs+Pairwise loss and that for LSTM in LSTM+Triplet loss are set to be exact the same as in Deep r-RSJBE. In addition, we also report the performance of LSTM+r-th root ranking loss to demonstrate the effectiveness of the r-th root ranking loss. By comparing it with Deep r-RSJBE, we will know that it is necessary to leverage the correlations between different pairs of time series. All experiments are repeated 5 times and the average performance (MAP, Precision@k, Recall@k) is reported for comparison. Note that EU, LSH, ITQ, HDML, and TopRSBC employ the vectorized raw time series segment as the input while deep learning based approaches directly utilize the raw multivariate time series segment $\mathbf{X}$ as the input.

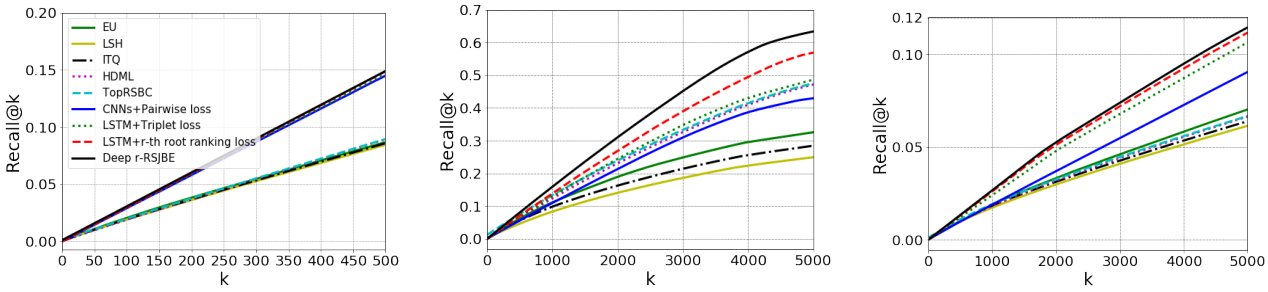
For EU, it achieves 0.5376, 0.4479, and 0.4406 MAP on EEG Eye State, PAMAP2, and SHL, respectively. For all the compared binary embedding algorithms, when the number of binary bits v varies from 32, 64, to 128, their results are shown in Table 2. Among

[2] http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring
[3] http://www.shl-dataset.org
[4] https://songdj.github.io



(a) Precision@k on EEG Eye State (b) Precision@k on PAMAP2 (c) Precision@k on UHL

Figure 4: Precision@k with 32 binary bits on EEG Eye State, PAMAP2, UHL.



(a) Recall@k on EEG Eye State (b) Recall@k on PAMAP2 (c) Recall@k on UHL

Figure 5: Recall@k with 32 binary bits on EEG Eye State, PAMAP2, UHL.

those unsupervised algorithms, we observe that ITQ in general outperforms LSH. This implies that exploring and exploiting underlying data structures, distributions, or topological information can yield more effective binary embeddings for multivariate time series retrieval task. Among the compared algorithms, we notice that supervised linear algorithms (*e.g.*, HDML and TopRSBC) generally outperform unsupervised algorithms since the former leverage label information to learn discriminative binary embedding functions. The two deep learning algorithms, *i.e.*, CNNs+Pairwise loss and LSTM+Triplet loss outperform supervised linear methods and unsupervised methods. This suggests that both CNNs and LSTM can learn good representations for the raw multivariate time series segment. We also notice that LSTM+r-th root ranking loss consistently outperforms six baseline approaches on EEG Eye State as well as PAMAP2 and achieves comparable performance to the best baseline methods on SHL. This indicates that r-th root ranking loss is superior to triplet loss since it optimizes the precision at the top of a ranking list. Finally, we observe that Deep r-RSJBE consistently achieves best MAP on these three datasets. This is because Deep r-RSJBE not only considers the temporal dynamics in the raw time series and the correlations between different pairs but also employs r-th root ranking loss to optimize the precision at the top positions of a Hamming distance ranking list.

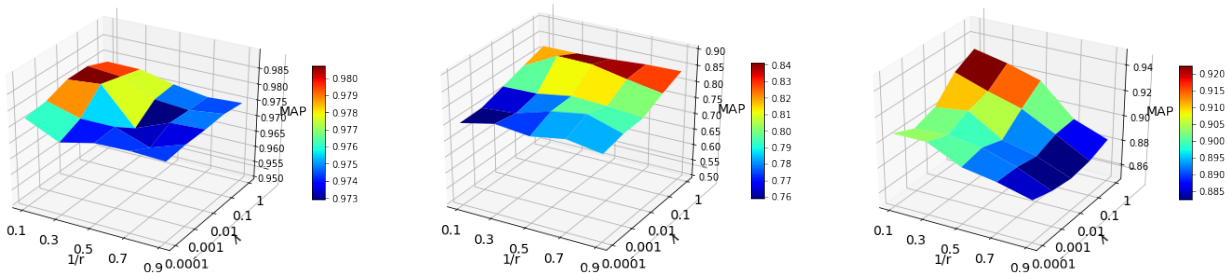
We further investigate the effectiveness of the proposed Deep r-RSJBE by comparing its Precision@k and Recall@k (when k varies) to those of the competing algorithms in Figures 4 and Figure 5, respectively. When k varies, we find that Deep r-RSJBE consistently outperforms the other algorithms over these three datasets when the number of bits is fixed as $v = 32$. This suggests that our learned joint binary embedding and r-th root ranking loss can maintain high precision at the top of a Hamming distance ranking list.

Table 3: The training time (seconds), embedding time (seconds), and query time (seconds) for Deep r-RSJBE on three multivariate time series datasets (v=32 bits).

Dataset	training time	binary embedding	query time
EEG Eye State	238.41	1.72×10^{-5}	5.05×10^{-4}
PAMAP2	4842.54	1.46×10^{-5}	2.60×10^{-3}
SHL	4814.40	1.61×10^{-5}	7.92×10^{-3}

4.4 Efficiency

We examine the training time, embedding time, and query time of Deep r-RSJBE in Table 3 when $v = 32$ bits. We can observe that the training time for Deep r-RSJBE is generally less than 1.5 hours for different datasets (on single GPU). Given a query segment, the average time to obtain an binary embedding (on single GPU) is around 1.46×10^{-5} to 1.72×10^{-5} seconds on these three datasets. Meanwhile, with a query segment, the average query time for Deep r-RSJBE to obtain the top 500, 5000, and 5000 relevant examples (on CPU) is 5.05×10^{-4}, 2.60×10^{-3}, and 7.92×10^{-3} seconds on EEG Eye State, PAMAP2, and SHL, respectively. For EU, the average query time is 1.22×10^{-3}, 7.05×10^{-3}, and 2.21×10^{-2} seconds on



(a) MAP for EEG Eye State. (b) MAP for PAMAP2. (c) MAP for SHL.

Figure 6: The parameter sensitivity of Deep r-RSJBE with respect to $\lambda = \{0.0001, 0.001, 0.01, 0.1, 1\}$ and $\frac{1}{r} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ when $v = 32$ bits.

these three datasets and it will linearly increase as the length of query segment increases. While for Deep r-RSJBE, the query time will not change as long as the size of binary embedding is fixed.

4.5 Parameter Sensitivity

We study the sensitivity of Deep r-RSJBE with respect to the parameters $\lambda = \{0.0001, 0.001, 0.01, 0.1, 1\}$ and $\frac{1}{r} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for three datasets when $v = 32$ bits. When we vary the value of λ or $\frac{1}{r}$, we keep the other parameters fixed. We plot the MAP with respect to λ and $\frac{1}{r}$ in Figure 6. We observe that the performance of Deep r-RSJBE is relatively stable on three datasets when λ and r varies. Furthermore, we notice that Deep r-RSJBE can achieve relative better performance when $\frac{1}{r}$ is relatively small. This is because more penalization will be put on top of the Hamming distance ranking list when r becomes larger.

5 CONCLUSION

We developed a Deep r-th root of Rank Supervised Joint Binary Embedding (Deep r-RSJBE) to perform multivariate time series retrieval. Given a raw multivariate time series segment, Deep r-RSJBE employed LSTM units to encode the temporal dynamics and utilized CNNs to explicitly encode the correlations (interactions) between different pairs of time series (sensors). Subsequently, a joint binary embedding was learned to incorporate both the temporal dynamics and correlations. Finally, a novel r-th root ranking loss was employed to optimize the precision at the top of a Hamming distance ranking list. Our empirical studies on EEG Eye State dataset, PAMAP2 dataset, and UHL dataset, demonstrated the effectiveness and the efficiency of the proposed Deep r-RSJBE.

In future, we will be interested to investigate how to perform deep joint binary embedding when label information is not available for multivariate time series retrieval task. In addition, we are also interested to apply Deep r-RSJBE to other applications, *e.g.*, anomaly detection.

ACKNOWLEDGEMENT

Dacheng Tao is supported by Australian Research Council Projects FL-170100117, DP-180103424, DP-140102164, LP-150100671.

REFERENCES

[1] A. Andoni and P. Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, 1 (2008), 117–122.
[2] M. Azzouzi and I.T. Nabney. 1998. Analysing time series structure with Hidden Markov Models. In *Proc. of the IEEE Conference on Neural Networks and Signal Processing*. 402–408.
[3] A. Bagnall, C. Ratanamahatana, , E. Keogh, S. Lonardi, and J. Gareth. 2006. "A bit level representation for time series data mining with shape based similarity". *Data Mining and Knowledge Discovery* 13, 1 (2006), 11–40.
[4] D. Berndt and Clifford J. 1994. Using dynamic time warping to find patterns in time series. In *AAAI-94 Workshop on Knowledge Discovery in Databases*. 359–370.
[5] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. 1998. Min-wise independent permutations. In *Proc. of ACM Symposium on Theory of Computing*.
[6] Y. Cao, M. Long, J. Wang, Q. Yang, and P. S. Yu. 2016. Deep visual-semantic hashing for cross-modal retrieval. In *KDD*. 1445–1454.
[7] K. Chan and A. W. Fu. 1999. Efficient time series matching by wavelets. In *ICDE*. 126–133.
[8] L. Chen and Ng R. 2004. On the marriage of Lp-norms and edit distance. In *VLDB*. 792–803.
[9] L. Chen, M. T. ÄŰzsu, and V. Oria. 2005. Robust and fast similarity search for moving object trajectories. In *SIGMOD*. 491–502.
[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078* (2014).
[11] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. In *VLDB*. 1542–1552.
[12] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. 1994. Fast subsequence matching in time-Series databases. In *SIGMOD*. 419–429.
[13] E. Fink, K.B. Pratt, and H.S. Gandhi. 2003. Indexing of time series by major minima and maxima. In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*. 2332–2335.
[14] H. Gjoreski, M. Ciliberto, F. J. OrdoÃśez Morales, D. Roggen, S. Mekki, and S. Valentin. 2017. A versatile annotated dataset for multimodal locomotion analytics with mobile devices. In *Proc. ACM Conference on Embedded Networked Sensor Systems*.
[15] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. 2012. Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2012).
[16] Sepp H. and Jürgen S. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[17] D. Hallac, S. Vare, S. Boyd, and J. Leskovec. 2017. Toeplitz inverse covariance-based clustering of multivariate time series data. In *KDD*. 215–223.
[18] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *CVPR*.
[19] G. Jiang, H. Chen, and Yoshihira K. 2007. Efficient and scalable algorithms for inferring invariants in distributed system. *IEEE Transactions on Knowledge and Data Engineering* 19, 11 (2007), 1508–1523.
[20] S. S. Jones, R. S. Evans, T. L. Allen, A. Thomas, P. J. Haug, S. J. Welch, and G. L. Snow. 2009. A multivariate time series approach to modeling and forecasting demand in the emergency department. *Journal of Biomedical Informatics* 42, 1 (2009), 123–139.
[21] D. C. Kale, D. Gong, Z. Che, G. Medioni, R. Wetzel, P. Ross, and Y. Liu. 2014. An examination of multivariate time Series hashing with applications to health care. In *ICDM*.

[22] E. Keogh, Chakrabarti, M. K., Pazzani, and S. Mehrotra. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*. 151–162.

[23] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. 2000. "Dimensionality reduction for fast similarity search in large time series Databases". *KAIS* 3, 3 (2000), 263–286.

[24] S.-W. Kim, S. Park, and W. W. Chu. 2001. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*. 607–614.

[25] D. Kingma and J. Ba. 2014. Adam: a method for stochastic optimization. *arXiv:1412.6980* (2014).

[26] A. Krizhevsky. 2009. Learning multiple layers of features from tiny images. In *Technical report*.

[27] W.-J. Li, S. Wang, and W.-C. Kang. 2016. Feature learning based deep supervised hashing with pairwise Labels. In *IJCAI*.

[28] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. 2012. Supervised hashing with kernels. In *CVPR*.

[29] C. Luo and A. Shrivastava. 2016. SSH (Sketch, shingle, & hash) for indexing massive-scale time series. In *NIPS 2016 Time Series Workshop*. 38–58.

[30] M. Norouzi, D. J. Fleet, and R. Salakhutdinov. 2012. Hamming distance metric learning. In *NIPS*.

[31] J. Parkka, M. Ermes, Korpipaa P., Mantyjarvi J., and Peltola J. 2006. Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine* 6883 (2006), 119–128.

[32] P. Prickett, G. Davies, and Grosvenor R. 2011. A SCADA based power plant monitoring and management system. *Knowledge-Based and Intelligent Information and Engineering Systems* 6883 (2011), 433–442.

[33] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*. 2627–2633.

[34] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*. 262–270.

[35] A. Reiss and D. Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *Proc. of the 16th IEEE International Symposium on Wearable Computers (ISWC)*.

[36] F. Shen, C. Shen, W. Liu, and H. T. Shen. 2015. Supervised discrete hashing. In *CVPR*. 37–45.

[37] J. Shieh and E. Keogh. 2008. iSAX: indexing and mining terabyte sized time series. In *KDD*. 623–631.

[38] K Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

[39] S. W. Smith. 1999. The discrete fourier transform. In *The Scientist and Engineer's Guide to Digital Signal Processing (Second ed.)*. California Technical Publishing, California, Chapter 8.

[40] D. Song, W. Liu, and D. A. Meyer. 2016. Fast structural binary coding. In *IJCAI*. 2018–2024.

[41] D. Song, W. Liu, D. A. Meyer, R. Ji, and J. R. Smith. 2015. Top rank supervised binary coding for fast image search. In *ICCV*. 1922–1930.

[42] D. Song, W. Liu, D. A. Meyer, D. Tao, and R. Ji. 2015. Rank preserving hashing for rapid image search. In *DCC*. 353–362.

[43] D. Song, D. A. Meyer, and D. Tao. 2015. Top-k link recommendation in social networks. In *ICDM*. 389–398.

[44] I. Sutskever, O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. 3104–3112.

[45] N. Usunier, D. Buffoni, and P. Gallinari. 2009. Ranking with ordered weighted pairwise classification. In *ICML*. 1057–1064.

[46] M. Vlachos, G. Kollios, and G. Gunopulos. 2002. Discovering similar multidimensional trajectories. In *ICDE*. 673–684.

[47] Y. Weiss, A. Torralba, and R. Fergus. 2008. Spectral Hashing. In *NIPS*.

[48] J. Weston, S. Bengio, and N. Usunier. 2012. Large scale image annotation: learning to rank with joint world-image embeddings. *Machine Learning* 81, 1 (2012), 21–35.

[49] R. Xia, Y. Pan, Lai H., C. Liu, and S. Yan. 2014. Supervised hashing via image representation learning. In *AAAI*.

[50] B.-K. Yi and C. Faloutsos. 2000. Fast time sequence indexing for arbitrary Lp norms. In *VLDB*. 385–394.

[51] F. Zheng, G. Zhang, and Z. Song. 2001. "Comparison of different implementations of MFCC". *Journal of Computer Science & Technology* 16, 6 (2001), 582–589.