Deep Unsupervised Binary Coding Networks for Multivariate Time Series Retrieval

Dixian Zhu[†*], Dongjin Song[‡*], Yuncong Chen[‡], Cristian Lumezanu[‡], Wei Cheng[‡], Bo Zong[‡], Jingchao Ni[‡], Takehiko Mizoguchi[‡], Tianbao Yang[†], Haifeng Chen[‡]

[†]University of Iowa, IA 52242, USA

[‡]NEC Laboratories America, Inc., NJ 08540, USA

[†]{dixian-zhu,tianbao-yang}@uiowa.edu, [‡]{dsong,yuncong,lume,weicheng,bzong,tmizoguchi,jni,,haifeng}@nec-labs.com

Abstract

Multivariate time series data are becoming increasingly ubiquitous in varies real-world applications such as smart city, power plant monitoring, wearable devices, *etc*. Given the current time series segment, how to retrieve similar segments within the historical data in an efficient and effective manner is becoming increasingly important. As it can facilitate underlying applications such as system status identification, anomaly detection, *etc*. Despite the fact that various binary coding techniques can be applied to this task, few of them are specially designed for multivariate time series data in an unsupervised setting. To this end, we present Deep Unsupervised Binary Coding Networks (DUBCNs) to perform multivariate time series retrieval. DUBCNs employ the Long Short-Term Memory (LSTM) encoder-decoder framework to capture the temporal dynamics within the input segment and consist of three key components, *i.e.*, a temporal encoding mechanism to capture the temporal order of different segments within a mini-batch, a clustering loss on the hidden feature space to capture the hidden feature structure, and an adversarial loss based upon Generative Adversarial Networks (GANs) to enhance the generalization capability of the generated binary codes. Thoroughly empirical studies on three public datasets demonstrated that the proposed DUBCNs can outperform state-of-the-art unsupervised binary coding techniques.

Introduction

Nowadays, multivariate time series data are increasingly collected in numerous real-world applications. For instance, in a power plant (Prickett, Davies, and R. 2011), a large number of sensors can be employed to monitor the operation status in real-time. With a fitness tracking device, a temporal sequence of actions (Parkka et al. 2006), *e.g.*, walking for 5 minutes, running for 1 hour, and sitting for 15 minutes, *etc*, can be recorded and detected with related sensors. With the huge amount of historical multivariate time series data, how to interpret the current status becomes an important problem to investigate.

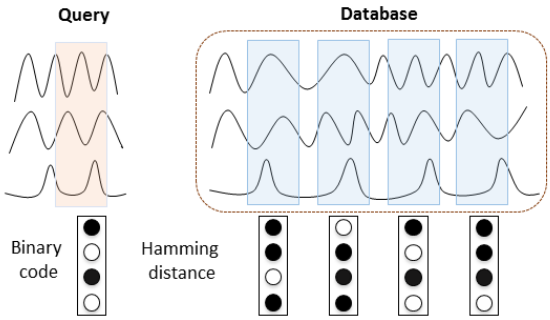


Figure 1: Multivariate time series retrieval task. A binary vector is learned to represent each segment and the similarity is measured with Hamming distance.

Since it is often difficult to obtain the label information of the historical data, we formulate it as an unsupervised multivariate time series retrieval problem. Specifically, given the current multivariate time series segment, *i.e.*, a slice of multivariate time series that lasts for a fixed period of time, we aim to find similar time series segments in the historical data (or database) as shown in Figure 1. An ideal solution to this problem can facilitate underlying applications such as system status identification, anomaly detection, *etc*.

A naive solution for this problem is to measure the pairwise distance in the raw space based upon Euclidean distance (EU) or Dynamic Time Warping (DTW) (Berndt and J. 1994; Rakthanmanon et al. 2012). This is usually computationally infeasible if the number and (or) the length of time series are relatively large. As a result, a surrogate is to obtain a compact representation of the multivariate time series segments, and then use a distance measure (*e.g.*, EU or DTW) to retrieve relevant segments. Over the past few decades, a number of approaches have been developed to denote a time series segment, *e.g.*, Discrete Fourier Transform (DCT) (Faloutsos, Ranganathan, and Manolopoulos 1994), Discrete Wavelet Transform (DWT) (Chan and Fu 1999), Piecewise Aggregate Approximation (PAA) (Keogh et al. 2001), *etc*. Most of these representations, however, are obtained based on human prior knowledge and hence could be suboptimal for multivariate time series retrieval because their objectives are inconsistent. In addition, even though the

[*]These two authors contributed equally to this work. Dongjin Song is the corresponding author.

obtained representations are effective, measuring the pairwise distance in the feature space based upon EU or DTW is still computationally expensive.

More recent advances (Kale et al. 2014; Luo and Shrivastava 2016) suggest that binary coding techniques, *e.g.*, Locality Sensitive Hashing (LSH) (Andoni and Indyk 2008), Sketch, Single, & Hash (SSH) (Luo and Shrivastava 2016), can be employed to further reduce the query complexity of high-dimensional similarity search compared to EU or DTW. In addition, deep unsupervised hashing techniques, *e.g.*, DeepBit (Lin et al. 2016) and HashGAN (Ghasedi Dizaji et al. 2018), have been employed to learn binary representations of images based upon deep neural networks. These approaches, however, cannot explicitly capture temporal dynamics within the input time series segment and the temporal order across different segments. Moreover, they may not capture the nonlinear hidden feature structure of the input segment and their representations may lack generalization capability.

To address the aforementioned issues, we present Deep Unsupervised Binary Coding Networks (DUBCNs) to perform multivariate time series retrieval. DUBCNs employ the Long Short-Term Memory (LSTM) encoder-decoder framework to capture the temporal dynamics within the input multivariate time series segment. In this framework, we first introduce a novel temporal encoding mechanism to encode the temporal order of different segments within a mini-batch, with the intuition that relatively close (or consecutive) segments in temporal dimension tend to share similar binary codes. Then, a clustering loss is imposed on the hidden feature space in order to capture the nonlinear hidden feature structure and enhance the discriminative property of generated binary codes. Finally, an adversarial loss based upon Generative Adversarial Networks (GANs) is utilized to improve the generalization capability of the generated binary codes. Thoroughly empirical studies on three public datasets demonstrated that the proposed DUBCNs can outperform state-of-the-art unsupervised binary coding techniques.

Related Works

The proposed DUBCNs are related to recent advances in time series representation as well as unsupervised binary coding techniques.

Existing techniques for time series representation can be divided into three main categories: temporal methods, spectral methods, and learning based methods. Temporal representations, *e.g.*, extrema extraction (Fink, Pratt, and Gandhi 2003), bit-level representation (Bagnall et al. 2006), Piecewise Aggregate Approximation (PAA) (Yi and Faloutsos 2000; Keogh et al. 2000), Adaptive Piecewise Constant Approximation (APCA) (Keogh et al. 2001), temporal logic (Bufo et al. 2014), *etc*., aim to encode the temporal structure of raw data. Spectral based methods, *e.g.*, Discrete Fourier Transform (DCT) (Faloutsos, Ranganathan, and Manolopoulos 1994), Discrete Wavelet Transform (DWT) (Chan and Fu 1999), Mel-Frequency Cepstral Coefficients (MFCC) (Zheng, Zhang, and Song 2001), *etc*., represent the raw data with frequency information. Learning based methods include principle component analysis (PCA), Hidden Markov Models (HMMs) (Azzouzi and Nabney 1998), *etc*. These representations are obtained based on human prior knowledge and thus could be suboptimal for multivariate time series retrieval since their objectives are usually decoupled.

DUBCNs is more closely related to binary coding techniques which include data independent binary coding approaches (Andoni and Indyk 2008; Broder et al. 1998) as well as data dependent (learning based) binary embedding methods (Weiss, Torralba, and Fergus 2008; Gong et al. 2012; Liu et al. 2012; Norouzi, Fleet, and Salakhutdinov 2012; Shen et al. 2015; Song et al. 2015a; 2015b; Song, Liu, and Meyer 2016). In particular, learning based binary coding techniques can be further categorized into unsupervised methods such as spectral hashing (Weiss, Torralba, and Fergus 2008), Iterative Quantization (ITQ) (Gong et al. 2012), and supervised methods such as kernelized supervised hashing (Liu et al. 2012), *etc.*. More recently, unsupervised deep binary coding methods such as DeepBit (Lin et al. 2016) and HashGAN (Ghasedi Dizaji et al. 2018) have shown their effectiveness for unsupervised image retrieval task. Compared to these techniques, DUBCNs are unique since it is specially designed for multivariate time series data and thus can explicitly capture temporal dynamics within the input time series segment and the temporal order across different segments. Moreover, DUBCNs can not only capture the nonlinear hidden feature structure of the input segment and enhance the discriminative property of generated binary codes, but also improve the generalization capability.

Deep Unsupervised Binary Coding Networks

In this section, we present Deep Unsupervised Binary Coding Networks (DUBCNs) to perform multivariate time series retrieval (as shown in Figure 2). Specifically, we first state the problem to study. Then, we introduce DUBCNs which are essentially a novel LSTM Encoder-Decoder framework with three key components, *i.e.*, a temporal encoding mechanism, a clustering loss, and an adversarial loss, to perform binary coding. Finally, we introduce the objective of DUBCNs and the detailed training procedure.

Problem Statement

We first introduce notations used in the paper. Given a multivariate time series segment, *i.e.*, n time series with $\mathbf{X}_{t,w} = (\boldsymbol{x}^1, \cdots, \boldsymbol{x}^n)^\top = (\boldsymbol{x}_{t-w+1}, \cdots, \boldsymbol{x}_t) \in \mathbb{R}^{n\times w}$ where t is the time step index and w is the length of window size, we use $\boldsymbol{x}^k = (x^k_{t-w+1}, x^k_{t-w+2}, \cdots, x^k_t)^\top \in \mathbb{R}^w$ to represent the k-th time series of length w and employ $\boldsymbol{x}_t = (x^1_t, x^2_t, \cdots, x^n_t)^\top \in \mathbb{R}^n$ 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.

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

$$\arg\max_{\boldsymbol{X}_{p,w}\in\mathcal{D}} \mathcal{S}(\mathbf{X}_{q,w}, \mathbf{X}_{p,w}) \tag{1}$$

where $\mathcal{D} = \{\mathbf{X}_{p,w}\}$ is a collection of segments, p denotes the time index for p-th segment ($\forall w - 1 \leq p \leq T$), T

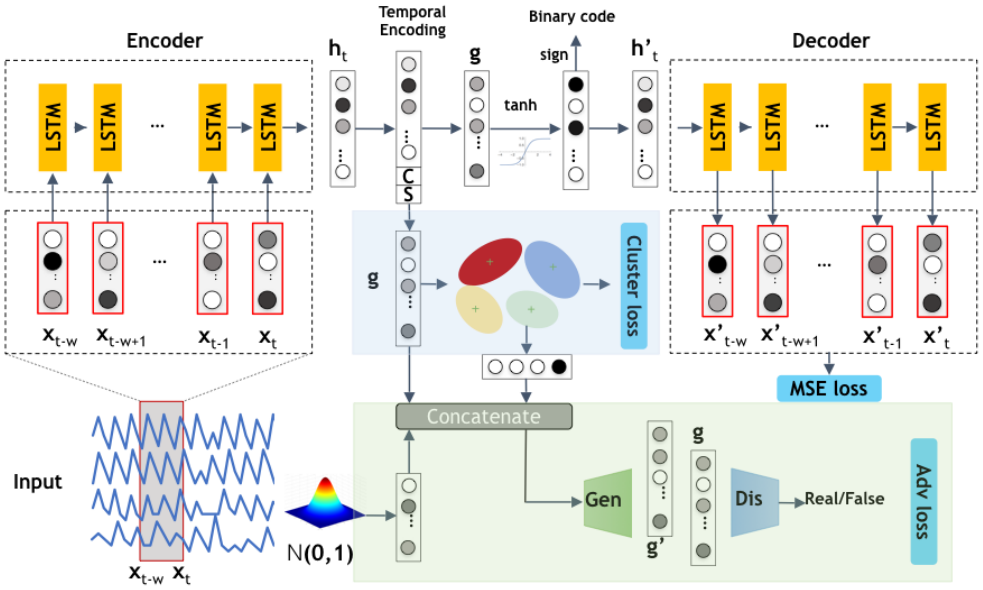


Figure 2: The architecture of DUBCNs. The left part is a LSTM encoder which is used to represent the input segment. The right part is a LSTM decoder which is employed to reconstruct the input segment. In the middle, the temporal encoding mechanism, the clustering loss, and the adversarial loss are shown from top to down.

denotes the total length of time series, and $\mathcal{S}(\cdot)$ represents a similarity measure function. Note that in practise, we can also use distance measure to retrieve similar segments with small distance.

LSTM Encoder

To encode the temporal information within a multivariate time series segment, similar to Seq2Seq model used in machine translation and video representation (Cho et al. 2014; Sutskever, Vinyals, and Le 2014; Srivastava, Mansimov, and Salakhudinov 2015), we adopt a LSTM encoder to represent the input time series segment. Specifically, given the input sequence $\boldsymbol{x}^k = (x^k_{t-w+1}, x^k_{t-w+2}, \cdots, x^k_t)^\top \in \mathbb{R}^w$ with $\boldsymbol{x}_t \in \mathbb{R}^n$, where n is the number of input time series, the encoder can be applied to learn a mapping from $\boldsymbol{x}_{t-1}$ to $\boldsymbol{h}_t$ (at time step t) with

$$\boldsymbol{h}_t = \text{LSTM}_{\text{enc}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t), \tag{2}$$

where $\boldsymbol{h}_t \in \mathbb{R}^m$ is the hidden state of the encoder at time t, m is the size of the hidden state, and LSTM_{enc} is a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) unit. 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:

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

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

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

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

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

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, respectively. The key reason for using 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. Several previous works (Qin et al. 2017; Song et al. 2018; Zhang et al. 2019) have shown its effectiveness on modeling time series data.

Temporal Encoding

Although LSTM encoder can capture the temporal information within each segment, the temporal order across different segments is not modeled explicitly. Based upon the intuition that two consecutive (or very close) segments are more likely to have similar binary codes, we develop a temporal encoding mechanism to explicitly encode temporal order of different segments. Specifically, for a given batch of $2N$ segments, we randomly sample half of them and sequentially sample the other half. Randomly sampled segments are employed to avoid unstable gradient and improve generalization capability. For these segments, a 2-dimensional vector of zero entries is concatenated with the original hidden feature vector $\boldsymbol{h}_t$. For sequentially sampled segments, a temporal encoding (TE) vector $(\sin(\frac{\pi i}{N}),\cos(\frac{\pi i}{N}))$ is employed to encode the relative temporal position of different segments with $N \geq i \geq 0$. Therefore, for each batch of segments, the temporal encoding vector (C,S) can be denoted as:

$$(C, S) = \begin{cases} (0, 0), & \text{if the segment is randomly sampled} \\ (\sin(\frac{\pi i}{N}), \cos(\frac{\pi i}{N})), & \text{sequential segments} \end{cases} \tag{8}$$

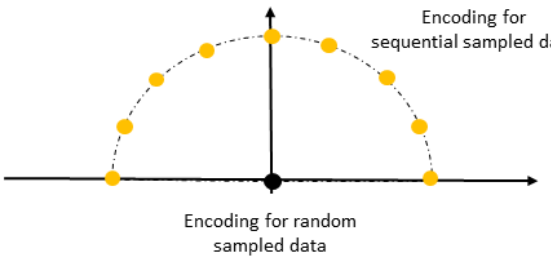


Figure 3: Temporal within-batch encoding for (C,S). The yellow points denote the temporal encoding vectors (C,S) for the sequentially sampled half batch and the black points represent the temporal encoding vectors (0,0) for the randomly sampled half batch.

where i is the index of the segment within the sequentially sampled half batch. The temporal encoding mechanism is shown in Figure 3.

After temporal encoding, a fully connected layer is employed to obtain a feature vector $\boldsymbol{g} \in \mathbb{R}^m$. Then, the hyperbolic tangent function is used to generate the approximated binary code $\tanh(\boldsymbol{g})$. Finally, another fully connected layer is used to get the feature vector $\boldsymbol{h}'_t \in \mathbb{R}^m$ which serves as the input of LSTM decoder. The detailed procedure is shown in Figure 2.

Note that although the idea of TE is similar to Positional Encoding (PE) in Transformer (Vaswani et al. 2017), they are essentially different. This is because, similar to LSTMs, PE encodes the temporal information within the segment while TE focuses on capturing the temporal order of different segments.

Clustering Loss

With the intuition that input multivariate time segments may exhibit different properties (such as uptrend, downtrend, *etc*.), it is rational to explore the nonlinear hidden feature structure of the input time series segments and encourage those segments falling into the same cluster to have more similar features than those segments falling into different clusters. In this way, the generated binary code can also preserve the discriminative information among clusters. For this purpose, assuming the initial cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^k$ are available in the hidden space, we can compute a soft assignment between the hidden feature points $\boldsymbol{g}_i$ and the cluster centroids, *i.e.*,

$$q_{ij} = \frac{(1 + ||\boldsymbol{g}_i - \boldsymbol{\mu}_j||/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j=1}^{k}(1 + ||\boldsymbol{g}_i - \boldsymbol{\mu}_j||/\alpha)^{-\frac{\alpha+1}{2}}}, \tag{9}$$

where $\boldsymbol{g}_i \in \mathbb{R}^m$ is the hidden feature obtained after a fully connected layer based upon TE. α are the degrees of freedom for the Student's t-distribution (Maaten and Hinton 2008). q_{ij} represents the probability of assigning segment i to cluster j. For simplicity, we set $\alpha = 1$ for all experiments. In practical applications, the initial cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^k$ are obtained based upon centroids in the raw space with k-means algorithm.

Inspired by DEC (Xie, Girshick, and Farhadi 2016), we adopt a clustering objective based upon KL divergence loss between the soft assignments q_i and the auxiliary target distribution p_i:

$$\mathcal{L}_{\text{cluster}} = \sum_{i=1}^{N}\sum_{j=1}^{k} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{10}$$

Since we expect the target distribution to improve cluster purity, put more emphasis on segments assigned with high confidence, and prevent large clusters from distorting the hidden feature space, p_{ij} can be calculated with

$$p_{ij} = \frac{q_{ij}^2/z_j}{\sum_{j'} q_{ij'}^2/z'_j}, \tag{11}$$

where $z_j = \sum_i q_{ij}$ denotes soft cluster counts.

Adversarial Loss

When exploring clustering in the hidden feature space of DUBCNs, one of the potential issues is overfitting. This is because the training is conducted over the batch level and the sampled segments in each batch could be biased. To overcome this issue, we employ an adversarial loss to enhance the generalization capability of DUBCNs. Specifically, an adversarial loss (Goodfellow et al. 2014; Mirza and Osindero 2014) is employed

$$\begin{aligned}\mathcal{L}_{adv} =& \mathbb{E}_{\boldsymbol{g}\sim p_{data}(\boldsymbol{g})}[\log D(\boldsymbol{g})] + \\ & \mathbb{E}_{\boldsymbol{z}\sim p_{data}(\boldsymbol{z})}[\log(1 - D(G([\boldsymbol{g} + \boldsymbol{z}; \boldsymbol{c}])))],\end{aligned} \tag{12}$$

where $G(\cdot)$ denotes a generator which tries to generate a feature vector that looks similar to feature vectors from the raw input segments, while $D(\cdot)$ aims to distinguish between the generated samples $G(\cdot)$ and real feature vector $\boldsymbol{g} \in \mathbb{R}^m$. $\boldsymbol{z}$ is a random noise vector of dimension m draw from a normal distribution. Here, instead of using a generator purely based upon $\boldsymbol{z}$, we use $\boldsymbol{g} + \boldsymbol{z}$ and concatenate the clustering membership $\mathbf{c} \in \mathbb{R}^k$ since it helps to generalize the hidden features within a specific cluster.

Specifically, $G(\cdot)$ consists of two fully connected layers with an output dimension of m. $D(\cdot)$ also consists of two fully connected layers. The output dimension of these two layers is m and 1, respectively.

LSTM Decoder

The feature vector $\boldsymbol{h}'_t \in \mathbb{R}^m$ serves as the context feature vector for LSTM decoder at time 0, *i.e.*, $\boldsymbol{b}'_0 = \boldsymbol{h}'_t$. Specifically, LSTM decoder is defined as

$$\boldsymbol{d}'_t = \text{LSTM}_{\text{dec}}(\boldsymbol{d}'_{t-1}, \boldsymbol{x}'_{t-1}), \tag{13}$$

where $\boldsymbol{d}_t$ is updated as:

$$\boldsymbol{f}'_t = \sigma(\mathbf{W}'_f[\boldsymbol{d}'_{t-1}; \boldsymbol{x}'_{t-1}] + \boldsymbol{b}'_f) \tag{14}$$

$$\boldsymbol{i}'_t = \sigma(\mathbf{W}'_i[\boldsymbol{d}'_{t-1}; \boldsymbol{x}'_{t-1}] + \boldsymbol{b}'_i) \tag{15}$$

$$\boldsymbol{o}'_t = \sigma(\mathbf{W}'_o[\boldsymbol{d}'_{t-1}; \boldsymbol{x}'_{t-1}] + \boldsymbol{b}'_o) \tag{16}$$

$$\boldsymbol{s}'_t = \boldsymbol{f}'_t \odot \boldsymbol{s}'_{t-1} + \boldsymbol{i}'_t \odot \tanh(\mathbf{W}'_s[\boldsymbol{d}'_{t-1}; \boldsymbol{x}'_{t-1}] + \boldsymbol{b}'_s) \tag{17}$$

$$\boldsymbol{d}_t = \boldsymbol{o}'_t \odot \tanh(\mathbf{s}'_t), \tag{18}$$

where $[\boldsymbol{d}'_{t-1}; \boldsymbol{x}'_{t-1}] \in \mathbb{R}^{m+n}$ is a concatenation of the previous hidden state $\boldsymbol{d}'_{t-1}$ and the decoder input $\boldsymbol{x}'_{t-1}$. $\mathbf{W}'_f$, $\mathbf{W}'_i$, $\mathbf{W}'_o$, $\mathbf{W}'_s \in \mathbb{R}^{m\times(m+n)}$, 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, respectively.

The reconstructed input at each time step is produced by

$$\hat{\boldsymbol{x}}_t = \boldsymbol{d}'_t \mathbf{W}_{\text{out}} + \boldsymbol{b}_{\text{out}}, \tag{19}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b}_{\text{out}} \in \mathbb{R}^n$. Finally, the Mean Square Error (MSE) is used as the objective for LSTM Encoder-Decoder, *i.e.*,

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{X}}^i_{t,w} - \mathbf{X}^i_{t,w}\|^2_{\text{F}}, \tag{20}$$

where i is the index for a segment and N is the number of segments in a batch.

Objective and Training Procedure

The full objective of DUBCNs is given by

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \lambda_1 \mathcal{L}_{\text{cluster}} + \lambda_2 \mathcal{L}_{\text{adv}}, \tag{21}$$

where $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are hyper-parameters to control the importance of clustering loss as well as adversarial loss. To optimize this objective, we need to solve a two-player Minimax game

$$G^*, D^* = \arg\min_G \max_D \mathcal{L}(G, D). \tag{22}$$

Therefore, we optimize the generator $G(\cdot)$ and discriminator $D(\cdot)$ iteratively. Specifically, when optimizing $D(\cdot)$, we only focus on the two fully connected layers of $D(\cdot)$, while optimizing $G(\cdot)$, we also update the network parameters via MSE loss $\mathcal{L}_{\text{MSE}}$ as well as clustering loss $\mathcal{L}_{\text{adv}}$.

We use Adam optimizer (Kingma and Ba 2014) to train the model. The size of the mini-batch is 128.

Experiments

In this section, we first introduce three datasets used in our evaluation. Next, we present parameter settings and evaluation metrics. Finally, we compare the proposed DUBCNs with 5 different baselines, study its efficiency, parameter sensitivity and model complexity.

Datasets

We consider three real-world multivariate time series datasets, *i.e.*, EEG Eye State, Internet of Things (IoT), and Twitter. The detailed statistics of these datasets are shown in Table 1.

- 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 sequentially sample 7,488 segments with the size of window w=5 and interval 2.

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

Table 1: The statistics of three multivariate time series datasets. N.A. denotes not available.

Dataset	# of time series	Length	# of classes
EEG Eye State	14	14,980	2
IoT	4	20,000	4
Twitter	77	583,250	N.A.

- IoT dataset is collected to monitor the GPU fan status (normal, eccentric, break, and stop). In our experiments, 9998 segments are sequentially sampled with the size of window w=5 and interval 2.
- Twitter dataset [2] does not contain class information and is originally used for Buzz prediction. Here, we use it for multivariate time series retrieval and sequentially generate 58,323 segments with window size w=20 and interval 10.

For those segments in each dataset, the first half is used for training, the next 10% are used for validation, and the last 40% are used for the test in our empirical studies.

Parameter Settings and Evaluation Metrics

DUBCNs contain 6 hyper-parameters. For simplicity, we fix mini-batch size as 128 in all experiments. The learning rate is selected from $\{10^{-6}$,10^{-5},10^{-4},$10^{-3}\}$. In addition, we set the hidden feature dimension of LSTM encoder/decoder as $m = 64, 128, 256$ to obtain different lengths of binary codes. For those two hyper-parameters λ_1 and λ_2 in the objective Eq 21, they are optimized based upon grid search over $\lambda_1 = \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\lambda_2 = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ when the number of cluster varies $k = \{2, 4, 8\}$. To determine the optimal network 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. DUBCNs 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.

To measure the effectiveness of unsupervised multivariate time series retrieval, given a query segment, we first calculate its K Nearest Neighbors (KNN) based upon Euclidean distance in the raw space (by comparing to historical data) and use KNN as the ground truth. Then, after getting the binary codes, we retrieve similar segments (in the historical data) based upon the Hamming distance and employ three evaluation metrics, *i.e.*, Mean Average Precision (MAP), precision at top-K positions (Precision@K), and recall at top-K positions (Recall@K), to measure the effectiveness.

Results

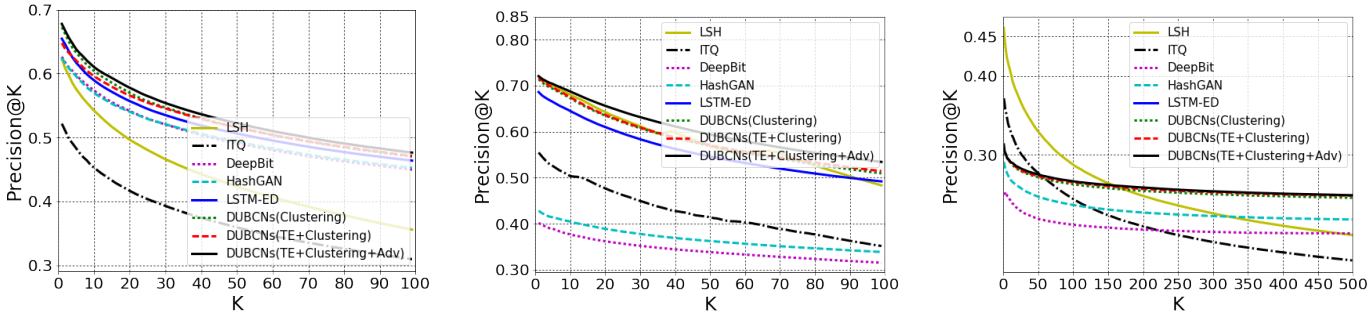
We evaluate the effectiveness of DUBCNs for unsupervised multivariate time series retrieval tasks based upon three different datasets.

Baselines. We compare DUBCNs with 5 different baseline methods. Among these baseline methods, two are unsupervised shallow methods, including one randomized

[2] https://archive.ics.uci.edu/ml/datasets/Buzz+in+social+media+#

Table 2: Unsupervised multivariate time series retrieval performance (**MAP**) on EEG Eye State (KNN=100), IoT (KNN=100), and Twitter (KNN=500) when $m = 64$, 128, and 256. The best MAP is displayed in bold-face type.

Algorithms	**EEG Eye State**			**IoT**			**Twitter**		
♯ **Bits**	64	128	256	64	128	256	64	128	256
LSH	0.1999	0.2561	0.3063	0.3839	0.4833	0.5346	0.0775	0.1159	0.1599
ITQ	0.1558	0.1425	0.1448	0.2199	0.2173	0.2173	0.0575	0.0904	0.1115
DeepBit	0.2725	0.3342	0.3816	0.1789	0.2038	0.2150	0.0711	0.0896	0.1149
HashGAN	0.2732	0.3352	0.3706	0.2034	0.2280	0.2452	0.0808	0.1010	0.1139
LSTM-ED	0.2832	0.3585	0.3870	0.3844	0.4972	0.5211	0.1015	0.1331	0.1562
DUBCNs(Clustering)	0.2915	0.3648	0.3979	0.4091	0.4950	0.5416	0.1057	0.1349	0.1601
DUBCNs(TE+Clustering)	0.2926	0.3640	0.4103	0.4117	0.4954	0.5446	0.1077	0.1337	0.1607
DUBCNs(TE+Clustering+Adv)	**0.2972**	**0.3669**	**0.4279**	**0.4406**	**0.5210**	**0.5646**	**0.1080**	**0.1371**	**0.1615**



(a) Precision@K on EEG Eye State (b) Precision@K on IoT (c) Precision@K on Twitter

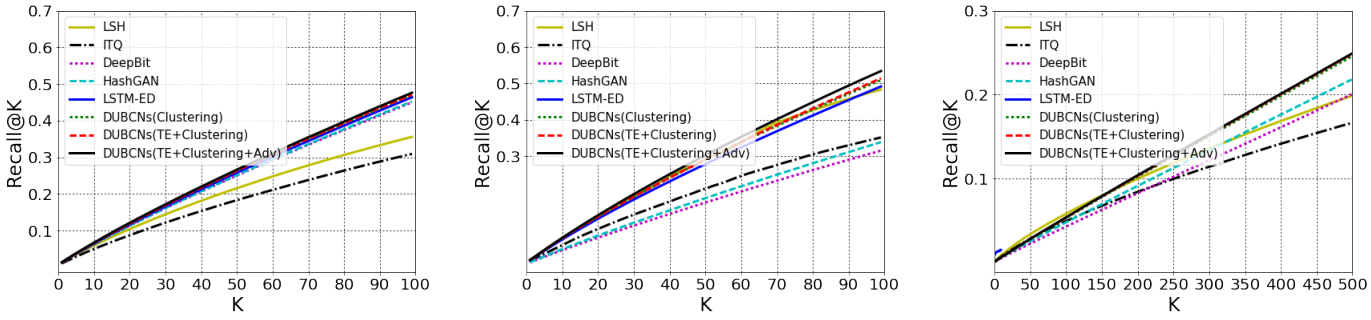
Figure 4: Precision@K with 64 binary bits on EEG Eye State, IoT, and Twitter.

method Locality-Sensitive Hashing (LSH) (Andoni and Indyk 2008) and one linear projection method Iterative Quantization (ITQ) (Gong et al. 2012). The other three are unsupervised deep binary coding approaches, *i.e.*, DeepBit (Lin et al. 2016), HashGAN (Ghasedi Dizaji et al. 2018), and LSTM Encoder-Decoder (LSTM-ED) (Srivastava, Mansimov, and Salakhudinov 2015). All experiments are repeated 5 times and the average performance (MAP, Precision@K, Recall@K) is reported for comparison. Note that LSH and ITQ employ the vectorized raw time series segment as the input while unsupervised deep binary coding approaches DeepBit, HashGAN, and LSTM-ED directly utilize the raw multivariate time series segment $\mathbf{X}_{t,w}$ as the input.

Effectiveness. The MAP of DUBCNs as well as 5 baseline approaches for unsupervised multivariate time series retrieval are shown in Table 2. We notice that unsupervised deep binary coding approaches, *i.e.*, DeepBit and HashGAN, outperform shallow unsupervised approaches, *i.e.*, LSH and ITQ, on EEG Eye State and Twitter. This is because DeepBit and HashGAN can obtain good binary representations via deep neural networks. Meanwhile, we observe that LSTM-ED consistently outperforms DeepBit and HashGAN over three different datasets. This may be due to that DeepBit and HashGAN are specially designed for images and cannot explicitly capture the temporal dynamics in the input segment which could be essential for binary coding. Finally, we find that the proposed DUBCNs (TE+Clustering+Adv) consistently outperform all baseline approaches on three datasets. This justifies the effectiveness of the proposed technique.

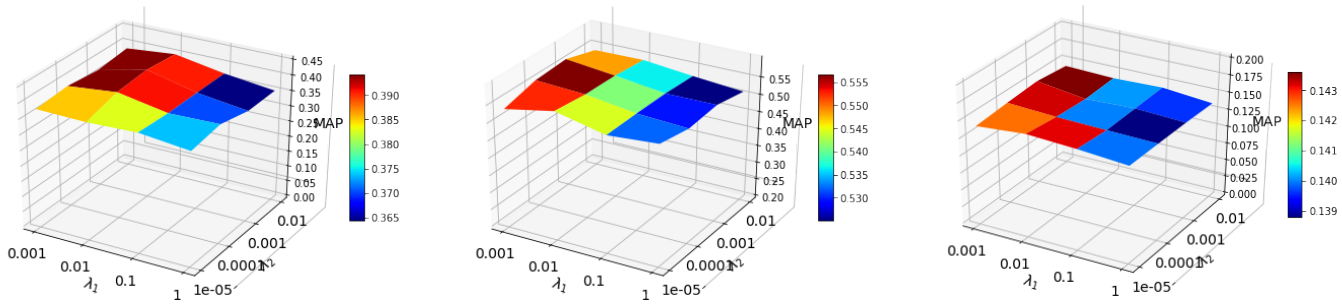
We also compare DUBCNs (m=64 bits) with 5 baseline methods based upon Precision@K and Recall@K in Figure 4 and Figure 5, respectively. We observe DUBCNs (TE+Clustering+Adv) generally outperform baseline approaches. This is because DUBCNs not only encode the temporal order of different segments but also capture the hidden feature structure and can generalize well.

Ablation study. To further investigate how each component of DUBCNs help to improve the unsupervised retrieval task, we also compare the full DUBCNs (TE+Clustering+Adv), with its variants, *i.e.*, DUBCNs (Clustering) in which both clustering loss and MSE loss is considered, and DUBCNs (TE+Clustering) in which Temporal Encoding (TE) mechanism, clustering loss, and MSE loss are all applied. In Table 2, we notice that DUBCNs (Clustering) generally outperforms LSTM-ED. This suggests that modeling the hidden feature structure appropriately is helpful for retrieval tasks. We also observe that DUBCNs (TE+Clustering) consistently outperform DUBCNs (Clustering). It indicates the necessity of the Temporal Encoding (TE) mechanism. Finally, DUBCNs (TE+Clustering+Adv) outperform



(a) Recall@K on EEG Eye State (b) Recall@K on IoT (c) Recall@K on Twitter

Figure 5: Recall@K with 64 binary bits on EEG Eye State, IoT, and Twitter.



(a) EEG Eye State (b) IoT (c) Twitter

Figure 6: Parameter sensitivity study on 256 binary bits on EEG Eye State, IoT, and Twitter.

DUBCNs (TE+Clustering) in general. This suggests that adversarial loss can help enhance the generalization of DUBCNs.

Efficiency. We examine query time for DUBCNs when $m = 64$ bits on EEG Eye State dataset. Given a query segment, the average binary code generation time for an input segment is 1.65×10^{-5} seconds and the average query time to obtain the top 100 relevant examples (on CPU) is 5.05×10^{-4} seconds. For the EU, the average query time is 1.22×10^{-3} seconds and it will linearly increase as the length of the query segment increases. While for DUBCNs, the query time will not change as long as the length of the binary code is fixed. For IoT and Twitter datasets, the results are similar.

Parameter Sensitivity and Model Complexity

We study the sensitivity of DUBCNs with respect to the parameters $\lambda_1 = \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\lambda_2 = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ for three datasets when $m = 256$ bits and k=8. The other parameters are fixed. We plot the MAP with respect to λ_1 and λ_2 in Figure 6 and can observe that the performance of DUBCNs is relatively stable on three datasets when λ_1 and λ_2 varies.

We also study the effect of k (the number of clusters) in Table 3 when $m = 256$ bits, and λ_1 and λ_2 are fixed. We notice that the performance of DUBCNs is gradually increasing when k varies from 2, 4, to 8 on three different datasets.

When $m = 64$, the number of model parameters for LSTM-ED, DUBCNs (Clustering), DUBCNs (TE+Clustering), and DUBCNs (TE+Clustering+Adv) are 164800, 164800, 164928, and 173825 for EEG; 153280, 153280, 153408, and 162305 for IoT; 237376, 237376, 237504, and 246401 for Twitter. Compared to LSTM-ED, DUBCNs (TE+Clustering+Adv) only increases the number of parameters around 5%.

Dataset	$k = 2$	$k = 4$	$k = 8$
EEG Eye State	0.3913	0.4028	0.4279
IoT	0.5178	0.5252	0.5646
Twitter	0.1502	0.1510	0.1615

Table 3: The parameter sensitivity of k. MAP of k vs dataset

Conclusion

In this paper, we introduced Deep Unsupervised Binary Coding Networks (DUBCNs) to perform multivariate time series retrieval. DUBCNs are essentially a Long Short-Term Memory (LSTM) encoder-decoder framework consisting of three key components, a temporal encoding mechanism to encode the temporal order of different segments within a mini-batch, a clustering loss on the hidden feature space to capture the nonlinear hidden feature structure and enhance the discriminative property of generated binary codes, and an adversarial loss to improve the generalization capability of the generated binary codes. Our experiment results on three public datasets demonstrated that the proposed DUBCNs can outperform state-of-the-art binary coding techniques.

References

Andoni, A., and Indyk, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM* 51(1):117–122.

Azzouzi, M., and Nabney, I. 1998. Analysing time series structure with hidden markov models. In *Proc. of the IEEE Conference on Neural Networks and Signal Processing*, 402–408.

Bagnall, A.; Ratanamahatana, C.; ; Keogh, E.; Lonardi, S.; and Gareth, J. 2006. A bit level representation for time series data mining with shape based similarity. *DMKD* 13(1):11–40.

Berndt, D., and J., C. 1994. Using dynamic time warping to find patterns in time series. In *AAAI-94 Workshop on Knowledge Discovery in Databases*, 359–370.

Broder, A. Z.; Charikar, M.; Frieze, A. M.; and Mitzenmacher, M. 1998. Min-wise independent permutations. In *Proc. of ACM Symposium on Theory of Computing*.

Bufo, S.; Bartocci, E.; Sanguinetti, G.; Borelli, M.; Lucangelo, U.; and Bortolussi, L. 2014. Temporal logic based monitoring of assisted ventilation in intensive care patients. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, 391–403.

Chan, K., and Fu, A. W. 1999. Efficient time series matching by wavelets. In *ICDE*, 126–133.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Faloutsos, C.; Ranganathan, M.; and Manolopoulos, Y. 1994. Fast subsequence matching in time-series databases. In *SIGMOD*, 419–429.

Fink, E.; Pratt, K.; and Gandhi, H. 2003. Indexing of time series by major minima and maxima. In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, 2,332–2,335.

Ghasedi Dizaji, K.; Zheng, F.; Sadoughi, N.; Yang, Y.; Deng, C.; and Huang, H. 2018. Unsupervised deep generative adversarial hashing network. In *CVPR*, 3,664–3,673.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2012. Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2,672–2,680.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1,735–1,780.

Kale, D. C.; Gong, D.; Che, Z.; Medioni, G.; Wetzel, R.; Ross, P.; and Liu, Y. 2014. An examination of multivariate time series hashing with applications to health care. In *ICDM*.

Keogh, E.; Chakrabarti, K.; Pazzani, M.; and Mehrotra, S. 2000. Dimensionality reduction for fast similarity search in large time series databases. *KAIS* 3(3):263–286.

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

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lin, K.; Lu, J.; Chen, C.-S.; and Zhou, J. 2016. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, 1,183–1,192.

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

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

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-SNE. *JMLR* 9(11):2,579–2,605.

Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

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

Parkka, J.; Ermes, M.; P., K.; J., M.; and J., P. 2006. Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine* 6883:119–128.

Prickett, P.; Davies, G.; and R., G. 2011. A SCADA based power plant monitoring and management system. *Knowledge-Based and Intelligent Information and Engineering Systems* 6883:433–442.

Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; and Cottrell, G. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*, 2,627–2,633.

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

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

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

Song, D.; Liu, W.; Ji, R.; Meyer, D. A.; and Smith, J. 2015b. Top rank supervised binary coding for visual search. In *ICCV*, 1,922–1,930.

Song, D.; Xia, N.; Cheng, W.; Chen, H.; and Tao, D. 2018. Deep r-th root of rank supervised joint binary embedding for multivariate time series retrieval. In *KDD*, 2,229–2,238.

Song, D.; Liu, W.; and Meyer, D. 2016. Fast structural binary coding. In *IJCAI*, 2,018–2,024.

Srivastava, N.; Mansimov, E.; and Salakhudinov, R. 2015. Unsupervised learning of video representations using lstms. In *ICML*, 843–852.

Sutskever, I.; Vinyals, O.; and Le, Q. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3,104–3,112.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5,998–6,008.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *NIPS*.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*, 478–487.

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

Zhang, C.; Song, D.; Chen, Y.; Feng, X.; Lumezanu, C.; Cheng, W.; Ni, J.; Zong, B.; Chen, H.; and Chawla, N. V. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *AAAI*, 1,409–1,416.

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