DEEP SYMBOLIC REPRESENTATION LEARNING FOR HETEROGENEOUS TIME-SERIES CLASSIFICATION

Shengdong Zhang[1,2], **Soheil Bahrampour**[1], **Naveen Ramakrishnan**[1], **Mohak Shah**[1,3] *
[1]Bosch Research and Technology Center, Palo Alto, CA
[2]Simon Fraser University, Burnaby, BC
[3]University of Illinois at Chicago, Chicago, IL
sza75@sfu.ca, Soheil.Bahrampour@us.bosch.com,
Naveen.Ramakrishnan@us.bosch.com, Mohak.Shah@us.bosch.com

ABSTRACT

In this paper, we consider the problem of event classification with multi-variate time series data consisting of heterogeneous (continuous and categorical) variables. The complex temporal dependencies between the variables combined with sparsity of the data makes the event classification problem particularly challenging. Most state-of-art approaches address this either by designing hand-engineered features or breaking up the problem over homogeneous variates. In this work, we propose and compare three representation learning algorithms over symbolized sequences which enables classification of heterogeneous time-series data using a deep architecture. The proposed representations are trained jointly along with the rest of the network architecture in an end-to-end fashion that makes the learned features discriminative for the given task. Experiments on three real-world datasets demonstrate the effectiveness of the proposed approaches.

1 INTRODUCTION

Rapid increase in connectivity of physical sensors and systems to the Internet is enabling large scale collection of time series data and system logs. Such temporal datasets enable applications like predictive maintenance, service optimizations and efficiency improvements for physical assets. At the same time, these datasets also pose interesting research challenges such as complex dependencies and heterogeneous nature of variables, non-uniform sampling of variables, sparsity, etc which further complicates the process of feature extraction for data mining tasks. Moreover, the high dependence of the feature extraction process on domain expertise makes the development of new data mining applications cumbersome. This paper proposes a novel approach for feature discovery specifically for temporal event classification problems such as failure prediction for heating systems.

Feature extraction from time-series data for classification has been long studied (Mierswa & Morik, 2005). For example, well-known Crest factor (Jayant & Noll, 1984) and Kurtosis method (Altmann, 2004) extract statistical measures of the amplitude of time-series sensory data. Other popular algorithms include feature extraction using frequency domain methods, such as power spectral density (Li et al., 2002), or time-frequency domain such as wavelet coefficients (Lu et al., 2014). More recent methods include wavelet synchrony (Mirowski et al., 2009), symbolic dynamic filtering (Gupta & Ray, 2007; Bahrampour et al., 2013) and sparse coding (Huang & Aviyente, 2006; Bahrampour et al., 2013). On the other hand, summary statistics such as count, occurrence rate, and duration have been used as features for event data (Murray et al., 2005).

These feature extraction algorithms are usually performed as a pre-processing step before training a classifier on the extracted features and thus are not guaranteed to be optimally discriminative for a given learning task. Several recent works have shown that better performance can be achieved when a feature extraction algorithm is jointly trained along with a classifier in an end-to-end fashion. For example, in Mairal et al. (2012); Bahrampour et al. (2016), dictionaries are trained jointly with classifiers to extract discriminative sparse codes as feature. Recent successes of deep learning

*All authors were with the Bosch Research and Technology Center at the time this work is done.

methods (Goodfellow et al., 2016) on extracting discriminative features from raw data and achieving state-of-the-art performance have boosted the effort for automatic feature discovery in several domains including speech (Krizhevsky et al., 2012b), image (Krizhevsky et al., 2012a), and text (Sutskever et al., 2014) data. In particular, it has been shown that recurrent neural networks (Elman, 1990) and its variants such as LSTMs (Hochreiter & Schmidhuber, 1997; Graves et al., 2013) are capable of capturing long-term time-dependency between input features and thus are well suited for feature discovery from time-series data.

While neural networks have also been used for event classification, these efforts have been mostly focused on either univariate signal (Hannun et al., 2014) or uniformly sampled multi-variate time-series data (Mirowski et al., 2009). In this paper, we focus on event classification task (and event prediction task that can be reformulated as event classification), where the application data consists of multi-variate, heterogeneous (categorical and continuous) and non-uniformly sampled time-series data. This includes a wide variety of application domains such as sensory data for internet of things, health care, system logs from data center, etc. Following are the main contributions of the paper:

- We propose three representation learning algorithms for time-series classification. The proposed algorithms are formulated as embedding layers, which receive symbolized sequences as their input. The embedding layer is then trained jointly with a deep learning architecture (such as convolutional or recurrent network) to automatically extract discriminating representations for the given classification task. The proposed algorithms differ in the way they embed the symbolized data and are named as Word Embedding, Shared Character-wise Embedding, and Independent Character-wise Embedding.
- The deep learning architectures combined with the proposed algorithms provide a unified framework to handle heterogeneous time-series data which regularly occur in most sensor data mining applications. They uniformly map data of any type into a continuous space, which enables representation learning within the space. We will provide detailed discussions on the suitability of the proposed representations and their respective strengths and limitations.
- We show that the proposed algorithms achieve state-of-the-art performance compared to both a standard deep architecture without symbolization and also compared to other classification approaches with hand-engineered features from domain experts. This is shown with experimental results on three real-world applications including hard disk failure prediction, seizure prediction, and heating system fault prediction.

2 Symbolization

Symbolization has been widely used as first step for feature extraction on time-series data, providing a more compact representation and acting as a filter to remove noise. Moreover, symbolization can be used to deal with heterogeneity of the data where multi-variate time-series contain both categorical, ordinal, and continuous variables. For example, symbolized sequences are used in Bahrampour et al. (2013) to construct a probabilistic finite state automata and a measure on the corresponding state transition matrix is then used as final feature which is fed into a classifier. However, this kind of features, which are extracted without explicitly optimizing for the given discriminative task, are typically suboptimal and are not guaranteed to be discriminative. Moreover, incorporating symbol-based representations and jointly training a deep network is non-trivial. In this work, we propose a unified architecture to embed the symbolized sequence as an input representation and jointly train a deep network to learn discriminative features. Symbolization for a discrete variable is trivial as the number of symbols is equal to the number of available categories. For continuous variables, this requires partitioning (also known as quantization) of data given an alphabet size (or the symbol set). The signal space for each continuous variable, approximated by training set, can be partitioned into a finite number of cells that are labeled as symbols using a clustering algorithm such as uniform partitioning, maximum entropy partitioning (Rajagopalan & Ray, 2006), or Jenks natural breaks algorithm (Jenks, 1967). The alphabet size for continuous variable is a hyper-parameter which can be tuned by observing empirical marginal distributions.

Figures 1 and 2 illustrate the symbolization procedure in a simple example converting a synthetic 2 dimensional time series $\{Z_1, Z_2\}$ into a sequence of representations. The histogram of continuous variable Z_1 contains two Gaussian-like distributions and thus is partitioned into 2 splits, i.e. for

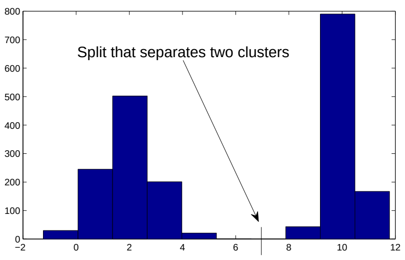


Figure 1: Partitiong of continous variable Z_1 based on its histogram.

$$\begin{array}{lcccc} \text{time} & 0 & 1 & 2 & \cdots \\ Z_1 & 2.1 & 11 & 5 & \cdots \\ Z_2 & C_5 & C_2 & C_1 & \cdots \end{array} \Rightarrow \begin{bmatrix} a_{Z_1} & b_{Z_1} & a_{Z_1} & \cdots \\ e_{Z_2} & b_{Z_2} & a_{Z_2} & \cdots \end{bmatrix} \begin{array}{c} \nearrow \\ \rightarrow \\ \searrow \end{array} \begin{array}{ll} \text{WdE} & \begin{bmatrix} \boldsymbol{v}_{ae} & \boldsymbol{v}_{bb} & \boldsymbol{v}_{aa} & \cdots \end{bmatrix} \\ \text{SCE} & \begin{bmatrix} \boldsymbol{v}_a^1 + \boldsymbol{v}_e^2 & \boldsymbol{v}_b^1 + \boldsymbol{v}_b^2 & \boldsymbol{v}_a^1 + \boldsymbol{v}_a^2 & \cdots \end{bmatrix} \\ \text{ICE} & \begin{bmatrix} (v_a^1\ v_e^2)^T & (v_b^1\ v_b^2)^T & (v_a^1\ v_a^2)^T & \cdots \end{bmatrix} \end{array}$$

Figure 2: Heterogeneous time-series symbolization along with word embedding (WdE), shared character-wise embedding (SCE), and independent character-wise embedding (ICE).

any realization of this variable in the time series, the value is replaced with symbol a_{Z_1} if it is less than 7, and with b_{Z_1} otherwise. For discrete variable Z_2, assuming it has 5 categories $Z_2 \in \{C_1, C_2, C_3, C_4, C_5\}$, we assign symbol a_{Z_2} to C_1, symbol b_{Z_2} to C_2, and so on.

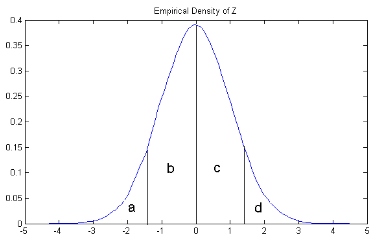
3 REPRESENTATION LEARNING

In this section, we propose three methods to learn representation from symbolized data.

3.1 WORD EMBEDDING (WDE)

Symbolized sequences at each time-step can be used to form a word by orderly collecting the symbol realizations of the variables. Thus, each time-series is represented by a sequence of words where each word represents the state of the multi-variate input at a given time. In Figures 2, word embedding vector (WdE) for word w is shown as $\boldsymbol{v}_w$. Each word of the symbolized sequence is considered as a descriptor of a "pattern" at a given time step. Even though the process of generating words ignores dependency among variables, it is reasonable to hypothesize that as long as a dataset is large enough and representative patterns occur frequently, an embedding layer along with a deep architecture should be able to capture the dependencies among the "patterns". The set of words on training data construct a vocabulary. Rare words are excluded from the vocabulary and are all represented using a out-of-vocabulary (OOV) word. OOV word is also used to represent words in test set which are not present in training data.

One natural choice for learning representation of the symbolized sequence is to learn embeddings of the words within the vocabulary. This is done by learning an embedding matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times v}$ where d is the embedding size and v is the vocabulary size (including OOV word), similar to learning word embedding in a natural language processing task (Dai & Le, 2015). One difference is that all words here have same length as the number of input variables. Each multi-variate sample is thus represented using a d-dimensional vector. It should be noted that the embedding matrix is learned jointly along with the rest of the network to learn discriminative representations. It should also be noted that although the problem of having rare words in training data is somewhat addressed by using OOV embedding vectors, this can limit the representation power if symbolization results in too many low-frequency words. Therefore, the quality of learning with word embeddings highly depends on the cardinality of the symbol set and the splits used for symbolization.



Initial Embedding

$$\begin{array}{cc} a & -0.12 \\ b & -0.01 \\ c & 0.02 \\ d & 0.1 \end{array} \xrightarrow[Update]{Gradient} \begin{array}{cc} a & -0.18 \\ b & 0.02 \\ c & -0.01 \\ d & 0.12 \end{array} \xrightarrow{Sort} \begin{array}{cc} a & -0.18 \\ b & -0.01 \\ c & 0.02 \\ d & 0.12 \end{array}$$

Figure 3: Empirical Probability Density of an counitous variable Z is shown along with the corresponding independent character-wise embeddings. The vairable has 4 symbols which are initiazed to maintain the ordered information. During training, and after each gradient update, the representations are sorted to enforce the ordered constraint.

3.2 Shared Character-wise Embedding (SCE)

The proposed word-embedding representation can capture the relation among multiple input variables given sufficient amount of training samples. However, as discussed in previous section, the proposed word-embedding representation learning needs careful selection of the alphabet size to avoid having too many low-frequency words and thus is inherently implausible to use in applications where the number of input time-series are too large. In this section, we propose an alternative character-level representation, which we call Shared Character-wise Embedding (SCE), to address this limitation while still being able to capture the dependencies among the inputs.

Instead of forming words at each time step, we use character embedding to represent each symbol and each observation at a time step is represented by the sum of all embedding vectors at a given time step. To formulate this, consider an m-dimensional time-series data where the symbol size for the i-th input is s^i. Let $\boldsymbol{e}_l^i \in \mathbb{R}^{s^i}$ be the one-hot representation for symbol l of the i-th input and $\boldsymbol{v}_l^i$ be the corresponding embedding vector. Also Let $\Phi = [\boldsymbol{V}^1 \dots \boldsymbol{V}^m] \in \mathbb{R}^{d \times \sum_i s^i}$ be the embedding matrix where $\boldsymbol{V}^i \in \mathbb{R}^{d \times s^i}$ is the collection of the embedding vectors for i-th input. Then, a given input sample $x^1, x^2, ..., x^m$ is represented as $\sum_i \boldsymbol{V}^i \boldsymbol{e}^i_{x^i} \in \mathbb{R}^d$, where x^i is the symbol realization of the i-th input. See Figure 2 for an example of the embedding (SCE) generated using this proposed representation. Since the representation of each word is constructed by summing the embeddings of individual characters, this method does not suffer from the unseen words issues.

3.3 Independent Character-wise Embedding (ICE)

Although SCE does not suffer from the low-frequency issue of WdE, both of these representations do not capture the ordinal information in the input time series. In this section, we propose an Independent Character-wise Embedding (ICE) representation that maintains ordinal information for symbolized continuous variables and categorical variables that have ordered information. To enforce the order constraint, we embed each symbol with a scalar value. Each input $i \in \{1, \dots, m\}$ is embedded independently and the resulting representation for a given sample $x^1, \dots, x^m$ is $[\boldsymbol{V}^1 e^1_{x^1} \dots \boldsymbol{V}^m e^m_{x^m}]^T \in \mathbb{R}^m$ where x^i is the symbol realization of the ith input and $\boldsymbol{V}^i$ is a row vector consisting of embedding scalars. The possible correlation among inputs are left to be captured using following layers after the embedding layer in the network. See Figure 2 for an example of generating embedding vector (ICE) using the proposed algorithm.

The embedding scalars for each symbol is initialized to satisfy the ordered information and during training we make sure that the learned representations satisfy the corresponding ordinal information, i.e. the embedding scalars of an ordinal variable are sorted after each gradient update. Figure 3 illustrates this process.

It should be noted that the embedding layer here has $\sum_i s^i$ parameters to learn and thus is slimmer compared to $d \times \sum_i s^i$ parameters of the shared character-wise embedding proposed in previous section. Both of the proposed character-wise representations have more compact embedding layer than the word-embedding representation that has $d \times v$ parameters as vocabulary size v is usually large.

4 Prediction architecture

4.1 Formulation of Prediction Problem

In this section, we formulate the event prediction problem as a classification problem. Let $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_T\}$ be a time-ordered collections of observation sequences (or clips) collected over T time steps, where $\boldsymbol{X}_t = \{\boldsymbol{x}_{t1}, \ldots, \boldsymbol{x}_{tN_t}\}$ represents tth sequence consisting of N_t consecutive measurements. As notation indicates, it is *not* assumed that the number of observations within each time step is constant. Let $\{l_1, l_2, \ldots, l_T\}$ be the corresponding sequence labels for $\boldsymbol{X}$, where $l_t \in \{0, 1\}$ encodes presence of an event within the tth time step, i.e. $l_t = 1$ indicates that a fault event is observed within the period of time input sequence $\boldsymbol{X}_t$ is collected. We define target labels $y = \{y_1, y_2, \ldots, y_T\}$ where $y_t = 1$ if an event is observed in the next K time-steps, i.e. $\sum_{j=t+1}^{t+K} l_j > 0$, and $y_t = 0$ otherwise. In this formulation, K indicates the prediction horizon and $y_t = 0$ indicates that no event is observed in the next K time-steps, refered to as monitor window in this paper. The prediction task is then defined as predicting y_t given input $\boldsymbol{X}_t$ and its corresponding past measurements $\{\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-2}, \ldots, \boldsymbol{X}_1\}$. Using the prediction labels y, the event prediction problem on time series data is converted into a classic binary classification problem. Note that although the proposed formulation can in theory utilize *all* the past measurements for classification, we usually fix a window size of M past measurements to limit computational complexity. For instance, suppose that X_t's are sensory data measurements of a physical system collected at the tth day of its operation and let $K = 7$ and $M = 3$. Then the classification problem for X_t is to predict y_t, i.e., whether an event is going to be observed in the next coming week of the physical system operation, given current and past three days of measurements.

4.2 Temporal Weighting Function

In rare event prediction tasks, the number of positive data samples, i.e. data corresponding to occurrences of a target event, is much fewer than the one of negatives. If not taken care of, this class imbalance problem causes that the decision boundary of a classifier to be dragged toward the data space where negative samples are distributed, artificially increasing the overall accuracy while resulting in low detection rate. This is a classic problem in binary classification and it is a common practice that larger misclassification cost are associated to positive samples to address this issue (Bishop, 2001). However, simply assigning identical larger weights to positive samples for our prediction formulation cannot emphasize the importance of temporal data close to a target event occurrence. We hypothesize that the data collected closer to an event occurrence should be more indicative of the upcoming error than data collected much earlier. Therefore, we design the following weighting function to deal with the temporal importance:

$$w_t = \begin{cases} \sum_{j=1}^{K} (K - j + 1) l_{t+j} & \text{if } y_t = 1 \\ 1 & \text{if } y_t = 0 \end{cases} \tag{1}$$

This weighting function gives relatively smaller weights to data far from event occurrences compared to those which are closer. In addition to temporal importance emphasis, it also deals with overlapping events. For example, suppose that two errors are observed at time samples $t + 1$ and $t + 3$ and prediction horizon K is set to 5. Then input sample X_t is within the monitor windows of both events and thus its weight is set to higher value of $w_t = (5 - 1 + 1) + (5 - 3 + 1) = 8$ as misclassification in this day may result in missing to predict two events. By weighting data samples in this way, a classifier is trained to adjust its decision boundary based on the importance information.

The above weight definition deals with temporal importance information for event prediction. We also need to re-adjust weights to address the discussed class imbalance issue. After determining the weight using Eq. 1 for each training sample, we re-normalize all weights such that the total sum of weights of positive samples becomes equal to the total sum of weights of negative samples.

The weighted cross entropy loss function is used as the optimization criterion to find parameters for our model. For the given input $\boldsymbol{X}_t$ with weight w_t, target label y_t, and the predicted label $\hat{y}_t$, the loss function is defined as :

$$l(y_t, \hat{y}_t) = w_t(y_t log \hat{y}_t + (1 - y_t) log(1 - \hat{y}_t)). \tag{2}$$

4.3 Network Architecture

Each of the proposed embedding layers can be used as the first layer of a deep architecture. The embedding layer along with the rest of the architecture are learned jointly to optimize a discriminative task, similar to natural language processing tasks (Gal, 2015). Thus, the embedding layer is trained to generate discriminative representations. The specific architectures are further discussed in the results section for each experiment.

5 Results

5.1 Hard-disk Failure Prediction

Backblaze data center has released its hard drive datasets containing daily snapshot S.M.A.R.T (Self-Monitoring, Analysis and Reporting Technology) statistics for each operational hard drive from 2013 to June 2016. The data of each hard drive are recorded until it fails. In this paper, the 2015 subset of the data on drive model "ST3000DM001" are used. As far as we know, no other prediction algorithm has been published on the data set of this model and thus we have generated our own training and test split. The data consists of several models of hard drives. There are $59,340$ hard drives out of which 586 of them (less than 1%) had failed. The data of the following 7 columns of S.M.A.R.T raw statistics are used: $5, 183, 184, 187, 188, 193, 197$. These columns corresponds to accumulative count values of different monitored features. We also added absolute difference between count values of consecutive days for each input column resulting in overall 14 columns. The data has missing values which are imputed using linear interpolation. The task is formulated to predict whether there is a failure in the next $K = 3$ days given current and past 4 days data.

The dataset is randomly split into a training set (containing 486 positives) and a test set (containing 100 positives) using hard disk serial number and without loosing the temporal information. Thus training and test set do not share any hard disk. For the experiment using word embedding (WdE), the data are symbolized with splits determined by observation of empirical histogram of every variable. The vocabulary is constructed using all words that have frequency of more than one. The remaining rare words are all mapped to the OOV word resulting into a vocabulary size of 509. For the experiments using shared and independent character-wise embedding, dubbed as SCE and ICE respectively, partitioning is done using maximum entropy partitioning (Bahrampour et al., 2013) with the alphabet size of 4, i.e. the first split is at the first 25-th percentile, the second split is at the 50-th percentile, and so on. The size of the embedding for WdE and SCE are selected as 16 and 2, respectively, using cross validation. Each of the proposed embedding layers is then followed by an LSTM (Hochreiter & Schmidhuber, 1997) layer with 8 hidden units and a fully connected layer for binary classification. Temporal weighting is not used here as it was not seem to be effective on this dataset, but cost-sensitive formulation is used to deal with this imbalanced dataset. As baseline methods, we also provided the results using logistic regression classification (LR), random forest (RF), and LSTM trained on normalized raw data (without symbolization). For LR and RF, the five days input data are concatenated to form the feature vector. The RF algorithm consists of 1000 decision trees. The LSTM networks are trained using ADAM algorithm (Kingma & Ba, 2014) with default learning rate of 0.001. Tabel 1 summarizes the performance on test data set. We reported the balanced accuracy, arithmetic mean of the true positive and true negative rates, the area under curve (AUC) of ROC as performance metrics. The balanced accuracy numbers are generated by picking a threshold on ROC that maximizes true prediction while maintaining a false positive rate of maximum 0.05. As it is seen, the proposed character-level embedding algorithms result in best performances. It should be noted that the input data is summary statistics, and not raw time-series data, and thus as seen the LR and LSTM algorithms, without symbolization, perform reasonably well.

5.2 Seizure prediction

We have also compared the performance of the proposed algorithms for seizure prediction. The data set is from the Kaggle American Epilepsy Society Seizure Prediction Challenge 2014 and consists of intracranial EEG (iEEG) clips from 16 channels collected from dogs and human. We used the data collected from dogs in our experiments, not including data from "Dog_5" as the data from one channel is missing. We generated the test sets from training data by randomly selecting 20% of one-hour clips. The length of each clip is $240,000$. We have down-sampled them to $1,200$ for efficient processing using recurrent networks. Five-fold cross validation is used to report the results.

Table 1: Performance comparison of fault prediction methods on Backblaze Reliability dataset. Base-line methods include random forest (RF), logistic regression classifier (LR), and LSTM, trained without symbolization. The three proposed embedding representations on symbolized data are WdE-LSTM, SCE-LSTM, and ICE-LSTM.

Models	Balanced Accuracy	AUC of ROC
RF	0.803	0.804
LR	0.846	0.851
LSTM	0.832	0.865
WdE-LSTM	0.834	0.812
SCE-LSTM	**0.855**	0.841
ICE-LSTM	0.835	**0.893**

Table 2: Performance comparison of the iRNN on raw data as well as iRNN using the proposed character-level embedding methods on symbolized data for seizure prediction.

Models	Balance Accuracy	AUC of ROC
iRNN	0.669	0.69
SCE-iRNN	0.761	0.811
ICE-iRNN	**0.77**	**0.818**

Table 3: Performance comparison of LR and LSTM on hand-designed features as well as the results generated using LSTM using the proposed embedding methods on symbolized data for fault prediction on Thermo-technology dataset.

Models	Balance Accuracy	AUC of ROC
LR	0.685	0.716
LSTM	**0.735**	0.766
WdE-LSTM	0.729	0.759
SCE-LSTM	0.7	0.697
ICE-LSTM	0.733	**0.769**

For the WdE algorithm, we observed that 90% of total words generated, using alphabet size of 4, have frequency of one which are all mapped to OOV and has resulted in poor performance. We also observed that using smaller alphabet size for WdE is not helpful resulting in too much loss of information and thus the performance of WdE algorithm is not reported here. For character-level embedding algorithms, maximum entropy partitioning is used with alphabet size of 50. The network used here consists of a one-dimensional convolutional layer with 16 filters, filter length of 3 and stride of 1, followed by a pooling layer of size 2 and stride 2 and an iRNN layer (Le et al., 2015) with 32 hidden units. We observed better results using iRNN than LSTM and thus we have reported the performance using iRNN. We have also reported the performance using same network on raw EEG data. The performance of these methods are summarized in Tabel 2. As it is seen, the proposed ICE embedding resulted in the best performance.

5.3 Heating System Failure Prediction

We also applied our method on an internal large dataset containing sensor information from thermo-technology heating systems. This dataset contains $132,755$ time-series of 20 variables where each time-series is data collected within one day. Nine of the variables are continuous and the remaining 11 variables are categorical. The task is to predict whether a heating system will have a failure in coming week. The dataset is highly imbalanced, where more than 99% of the data have no fault in the next seven days. After symbolizing the training data, for the experiment of using word embedding, the words that have relative frequency of less than 1% are considered as OOV words. The averaged length of each sequence is $6,000$. The embedding dimension for WdE and SCE algorithm are both chosen as 20 using a validation set. The network architecture includes an LSTM layer with 15 hidden units and a fully connected layer of dimension 50 which is followed by the final fully connected layer for binary classification. The same model architecture was used for the experiments of shared and independent character-wise embedding.

A simple trick is used to increase the use of GPU parallel computing power during the training phase, due to the large size of the training samples. For a given training time-series with T words, instead of sequentially feeding entire samples to the network, time-series is first divided into M sub-sequences of maximum length $\frac{T}{M}$ where each of these sub-sequences are processed independently and in-parallel. A max-pooling layer is then used on these feature vectors to get the final feature vector which represents the entire time-series. The generated feature is fed into a feed-forward neural network with sigmoid activation function to generate the predictions for the binary classification task. We call the sequence division technique as sequence chopping. Even though training an LSTM with this technique sacrifices temporal dependency longer than $\frac{T}{M}$ time steps, we have observed that by selecting a suitable chopping size, we can achieve competitive results and at the same time significant speed-up of the training procedure. The performance of the model having a similar LSTM architecture which is trained on 119 hand-engineered features is reported in Table 3. It should be noted that the hand engineered features have evolved over years of domain expertise and asset monitoring in the field. The results indicates that our methods results in competitive performance without the need for the costly process of hand-designing features.

6 CONCLUSIONS

We proposed three embedding algorithms on symbolized input sequence, namely WdE, SCE, and ICE, for event classification on heterogeneous time-series data. The proposed methods enable feeding symbolized time-series directly into a deep network and learn a discriminative representation in an end-to-end fashion which is optimized for the given task. The experimental results on three real-world datasets demonstrate the effectiveness of the proposed algorithms, removing the need to perform costly and sub-optimal process of hand-engineering features for time-series classification.

REFERENCES

Jürgen Altmann. Acoustic and seismic signals of heavy military vehicles for co-operative verification. *Journal of Sound and Vibration*, 273(4):713–740, 2004.

Soheil Bahrampour, Asok Ray, Soumalya Sarkar, Thyagaraju Damarla, and Nasser M. Nasrabadi. Performance comparison of feature extraction algorithms for target detection and classification. *Pattern Recognition Letters*, 34(16):2126 – 2134, 2013.

Soheil Bahrampour, Nasser M Nasrabadi, Asok Ray, and William Kenneth Jenkins. Multimodal task-driven dictionary learning for image classification. *IEEE Trans. on Image Processing*, 25(1): 24–38, 2016.

CM Bishop. Bishop pattern recognition and machine learning, 2001.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Y. Gal. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *ArXiv*, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://www.deeplearningbook.org`.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

Shalabh Gupta and Asok Ray. Symbolic dynamic filtering for data-driven pattern recognition. *Pattern recognition: theory and application*, pp. 17–71, 2007.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Ke Huang and Selin Aviyente. Sparse representation for signal classification. In *Advances in neural information processing systems*, pp. 609–616, 2006.

N. S. Jayant and Peter Noll. *Digital Coding of Waveforms, Principles and Applications to Speech and Video*, pp. 688. Prentice-Hall, 1984.

George F. Jenks. The data model concept in statistical mapping. *International yearbook of cartography*, 7(1), 1967.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012a.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012b.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Dan Li, Kerry D Wong, Yu Hen Hu, and Akbar M Sayeed. Detection, classification, and tracking of targets. *IEEE signal processing magazine*, 19(2):17–29, 2002.

Zhiyuan Lu, Xiang Chen, Qiang Li, Xu Zhang, and Ping Zhou. A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices. *IEEE Transactions on Human-Machine Systems*, 44(2):293–299, 2014.

Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012.

Ingo Mierswa and Katharina Morik. Automatic feature extraction for classifying audio data. *Machine learning*, 58(2-3):127–149, 2005.

Piotr Mirowski, Deepak Madhavan, Yann LeCun, and Ruben Kuzniecky. Classification of patterns of eeg synchronization for seizure prediction. *Clinical neurophysiology*, 120(11):1927–1940, 2009.

Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, 6(May):783–816, 2005.

Venkatesh Rajagopalan and Asok Ray. Symbolic time series analysis via wavelet-based partitioning. *Signal Processing*, 86(11):3309 – 3320, 2006.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

7 Appendix

Table 4: Number of trainable parameters for the three proposed embedding layers as well as total parameters of the network used in the three studied application. For the WdE on heating system data, an approximate number is provided as the vocabulary size is dependent on the training set used among the three cross-validations splits.

	Hard-Disk data		Seizure data		Heating System data	
Method	#. of embed. parameters	#. of total parameters	#. of embed. parameters	#. of total parameters	#. of embed. parameters	#. of total parameters
WdE	8144	8953	N/A	N/A	$\sim$713000	$\sim$715000
SCE	112	501	2400	3569	1335	4046
ICE	56	815	80	2465	69	3100