Generalizing Long Short-Term Memory Network for Deep Learning from Generic Data

HUIMEI HAN, Zhejiang University of Technology and Florida Atlantic University
XINGQUAN ZHU, Florida Atlantic University
YING LI, Xidian University

Long Short-Term Memory (LSTM) network, a popular deep-learning model, is particularly useful for data with temporal correlation, such as texts, sequences, or time series data, thanks to its well-sought after recurrent network structures designed to capture temporal correlation. In this article, we propose to generalize LSTM to generic machine-learning tasks where data used for training do not have explicit temporal or sequential correlation. Our theme is to explore feature correlation in the original data and convert each instance into a synthetic sentence format by using a two-gram probabilistic language model. More specifically, for each instance represented in the original feature space, our conversion first seeks to horizontally align original features into a sequentially correlated feature vector, resembling to the letter coherence within a word. In addition, a vertical alignment is also carried out to create multiple time points and simulate word sequential order in a sentence (*i.e.,* word correlation). The two dimensional horizontal-and-vertical alignments not only ensure feature correlations are maximally utilized, but also preserve the original feature values in the new representation. As a result, LSTM model can be utilized to achieve good classification accuracy, even if the underlying data do not have temporal or sequential dependency. Experiments on 20 generic datasets show that applying LSTM to generic data can improve the classification accuracy, compared to conventional machine-learning methods. This research opens a new opportunity for LSTM deep learning to be broadly applied to generic machine-learning tasks.

CCS Concepts: • **Computing methodologies** → Instance-based learning;

Additional Key Words and Phrases: Deep learning, feature learning, long short-term memory, classification

ACM Reference format:
Huimei Han, Xingquan Zhu, and Ying Li. 2020. Generalizing Long Short-Term Memory Network for Deep Learning from Generic Data. *ACM Trans. Knowl. Discov. Data* 14, 2, Article 13 (February 2020), 28 pages.
https://doi.org/10.1145/3366022

This work was conducted while the first author (Huimei Han) was a visiting scholar at the Florida Atlantic University. This research is supported by the US National Science Foundation (NSF) through Grant Nos. IIS-1763452 and CNS-1828181.
Authors' addresses: H. Han, Zhejiang University of Technology, College of Information Engineering, Hangzhou, Zhejiang, 310032, P.R. China; email: hmhan1215@zjut.edu.cn; X. Zhu, Florida Atlantic University, Department of Computer & Electrical Engineering and Computer Science, Boca Raton, FL, 33431; email: xzhu3@fau.edu; Y. Li, Xidian University, School of Telecommunications Engineering, Xi'an, Shannxi, 710071, P.R. China; email: yli@mail.xidian.edu.cn.

1556-4681/2020/02-ART13 $15.00
https://doi.org/10.1145/3366022

1 INTRODUCTION

Long Short-Term Memory (LSTM) network, a type of deep recurrent neural network (RNN) architecture, is becoming increasingly popular in recent years [20, 39]. Benefiting from its recurrent network design, this kind of deep neural network (DNN) has a powerful ability in preserving temporal information. When unrolling its loop structure along the time axis, all LSTM units (or cells) share the same structure and the temporal behavior of data can be captured and characterized clearly, making LSTM inherently advantageous to temporal or sequential data, as shown in Figure 1. As a result, recent years has witnessed great success of LSTM being applied to a broad range of applications involving temporal or sequential data, such as sequences, texts, time series, audio, video recognition, and the like [22, 24, 49, 50].

While the special recurrent neural structure allows LSTM to exploit temporal or sequential correlations of the data in a deep multi-layered fashion, to extract features preferable for the underlying tasks (*e.g.,* classification), using LSTM to solve generic data classification tasks is infeasible because generic data are typically collected/stored in the form of instance-feature table and do not have explicit temporal or sequential correlation. Conventional machine-learning methods, such as random forest, k-NN, and so on [45], do not take the correlations between feature/data into consideration for generic data classification tasks.

Indeed, in generic machine-learning settings, feature correlation is deemed a flaw which is primarily solved by using feature preprocessing to produce independent features for learning. We take the feature extraction, a classical feature preprocessing method, as an example. The feature extraction method, such as principle component analysis or manifold learning [44, 45], aims to produce orthogonal feature space preferable for the classification methods by utilizing arithmetic decomposition of features, implying that the new features do not have temporal or sequential correlation, as shown in Figure 2.

While the traditional two-step paradigm, feature preprocessing then learning, has been commonly used in generic machine-learning tasks, existing methods often rely on given input features and use data preprocessing to create independent features. Under this routine, the succeeding learning algorithms often do not aware, or pay attention to, feature correlations. Recently, deep learning has shifted the traditional two-step learning paradigm into a one-step feature learning framework, where the underlying deep-learning algorithms not only output a classification model but also extract meaningful features to represent the original data for a better classification accuracy [21, 39]. Furthermore, while conventional classification algorithms (along with feature selection/extraction) may achieve good performance for generic data classification tasks. Such methods heavily rely on hand-selected features by experts, requiring strong domain knowledge and data understanding. The implementation of this process is difficult because of the expensive expertise and a large amount of time requirements. Deep-learning algorithms focus on learning high-level features from the original data, which reduces the task of developing new feature extractor for underlying learning problems. For most of existing deep learning frameworks, such as convolutional neural networks (CNN) or LSTM, the feature learning process is to exploit spatial or temporal correlation of the data and utilize such correlations to form new features for learning. In addition, attention mechanism has been utilized as a useful tool for improving the performance of deep-learning model, such as attention-based LSTM [53].

For many real-world applications, such as image/speech recognition, deep learning has shown superb performance outperforming best conventional machine-learning classifiers. Nevertheless, for generic dataset classification tasks, where data are already represented in the form of instance-feature table, deep learning has not yet shown any privilege, compared to conventional machine-learning algorithms, such as decision trees, SVM, and so on. For existing deep-learning models, they are often selectively used depending on different types of domains and inputs. Specifically,

Fig. 1. Typical workflow of utilizing LSTM to process temporal data for extracting features preferable for classification.

Fig. 2. Typical workflow of applying traditional machine learning methods to generic data for classification tasks, where generic data is shown in the form of instance-feature table and instances are assumed identical and independently distributed (i.i.d.).

CNNs are commonly used for applications with image data, whereas LSTM is specially designed for sequential inputs such as speech and language. In our recent works [27, 28], original generic data is converted into a "synthetic image" format by creating "artificial correlation," and thus CNN can be used to learn effective features for classification, yet the research of using LSTM (and its subtypes) for generic machine learning still remain unexplored, at least in the literature.

The above observations motivate interesting questions on (1) whether LSTM is still effective in learning better classification models from generic instance-feature tabular represented machine-learning tasks, and reducing the task of developing new feature extractor for underlying learning problems, and (2) how to make LSTM work for generic data to obtain better performance. More fundamentally, the major challenges of generalizing LSTM for generic data are described as follows, which are also the challenges for other deep learning models:

- **Rationality of LSTM for Feature Learning:** While the recurrent neural structure allows LSTM to exploit data in temporal or spatial order to learn new features, generic instance-feature represented data do not have such temporal or spatial data correlation. We need to artificially create correlation for the rational use of LSTM to generic data for feature learning.
- **Feature Correlation and Ordering Exploration:** For the generic dataset, there are no explicit correlations between features. Since it is the correlation between features that the LSTM utilizes to extract meaningful features for classification, a critical issue of enabling LSTM for generic dataset is to explore correlations between features.
- **Instance Representation for LSTM:** Given properly ordered features, we need to construct LSTM compatible instances, which retain all the information of original dataset and ensure the temporal/sequential correlations at the same time. Then, the new instances can be utilized by the LSTM to extract meaningful features.
- **Construction Modeling:** We need to find a theoretical model to construct the new representation, ensuring that the new instance representation has sequential correlation with sound theoretical basis.

Motivated by the above challenges, our research proposes to generalize LSTM to generic machine-learning tasks (GeLSTM), where the **goal** is to explore feature correlation in the original data and convert each instance into a "synthetic sentence" format by using a two-gram probabilistic

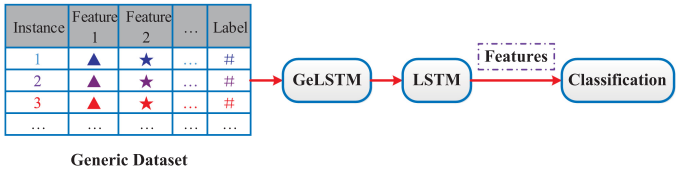

Fig. 3. Typical workflow of utilizing the proposed GeLSTM framework to make LSTM available for generic data classification tasks.

language model, such that LSTM and its subtypes can be utilized to the generic machine-learning tasks to obtain accurate classification models. Note that, since LSTM and its subtypes (including attention-based LSTM) belong to the same type of deep-learning model, we primarily focus on how to design "synthetic sentence" for such deep learning model to improve the classification performance, instead of considering which specific deep-learning model should be used for the "synthetic sentence." For simplify, we utilize LSTM as the deep-learning model for validation.

It is also worth noting that our research is fundamentally different from existing work on feature extraction/construction/selection, and so on [42, 55], mainly because that our goal is not to extract/learn/select good features from the given data, but to find an effective way to allow LSTM to be applied to the given data for feature learning. We believe that by enabling LSTM for generic learning tasks, it will eventually open a new paradigm to allow a rich set of deep-learning algorithms, including LSTM, to handle generic dataset classification tasks.

Figure 3 shows the typical workflow of utilizing the proposed GeLSTM for generic data classification tasks. In summary, we advance the LSTM based deep learning to generic machine learning tasks, by proposing solutions to address three challenges as follows.

- —To tackle the challenge of feature correlation and ordering exploration (*i.e., the second challenge*), we propose to derive the feature–feature correlation matrix and feature-label correlation vector, by utilizing Pearson correlation and chi-square accordingly. The correlation matrix and vector are further utilized to reorder features for data correlation construction.
- —To resolve the challenge of instance representation for LSTM (*i.e., the third challenge*), we convert each original instance into a synthetic sentence, which not only ensures the correlation within a signal word but also the correlation between words. For the synthetic sentence, each word includes all of the original features, but the order of features in each word is different. By doing so, the synthetic sentence can be fed to the LSTM for learning.
- —For the challenge of the construction modeling (*i.e., the last challenge*), we utilize two-gram model, which is a generally used model in the natural language modeling, to construct synthetic sentences. This model ensures that the synthetic sentence has the sequential correlation, and thus makes the synthetic sentence resemble to a genuine sentence.

The rest of this article is organized as follows. In Section 2, we introduce work related to feature representation learning. In Section 3, we present the problem definition and system overview. Section 4 describes the proposed GeLSTM algorithm. Experiments and conclusion are reported in Section 5 and Section 6, respectively.

2 RELATED WORK

Feature learning is to find informative and discriminating features to represent underlying objects. This is a crucial step for classification tasks, because machine-learning algorithms often require input data to be computationally convenient for computing. A critical step of all machine-learning tasks is to obtain meaningful features from the original data to represent the data. In other words, how to represent the original data is the key for learning. In general, there are two kinds of data

representation methods. One is the feature selection and extraction based data representation, and the other is deep learning based feature representation learning.

2.1 Feature Selection and Extraction Based Data Representation

Feature selection and extraction based data representation methods can be divided into feature selection based methods and feature extraction based data representation methods. They are commonly used technologies in traditional machine-learning models, where the original feature space is transformed into a low dimensional subspace to represent the data. A main characteristic of these methods is that they often require domain information to find meaningful features.

2.1.1 Feature Selection Based Data Representation Method. Feature selection based data representation methods aim to choose a subset from the original features by removing redundant or irrelevant features to represent the data without losing much information. This process does not introduce new features [56]. This kind of methods are used for better human interpretations, training time decreasing, dimensionality reduction, and generalization enhancement [8, 32]. Three types of feature selection techniques (*i.e.,* filter methods, wrapper methods, and embedded methods) are commonly used, and the differences between these selection algorithms can be found in the literature [37, 51]. In summary, filter methods choose highly ranked original features to represent the data, and wrapper methods explore features with the highest accuracy via warping predictors to a search algorithm [35]. Embedded methods decrease the computation time by performing the feature selection in the training process instead of before the training process, compared to the wrapper methods [9, 25, 38].

2.1.2 Feature Extraction Based Data Representation Method. As we described above, feature selection based data representation methods do not generate new features. However, feature extraction is commonly used to create informative and non-redundant features to represent the raw data. By doing so, the created new features facilitate the conventional machine-learning methods and enhances the human interpretations. Like feature selection methods, feature extraction is also a dimensionality reduction process, with information describing the original data being preserved. We can also use feature extraction method to connect data from different domains, such as cross-domain learning [59] and domain-adversarial learning [54].

According to whether the label information is available, feature extraction based data representation methods can be divided into supervised methods, unsupervised methods, and semi-supervised methods. More specifically, supervised feature extraction methods, such as Fisher discriminant analysis, utilize label information to extract informative features for better classification [51]. Unsupervised feature extraction methods, such as principal components analysis (PCA) [48] and independent component analysis (ICA), extract meaningful features based on the statistic characteristics of data without label information. PCA employs the covariance structure of data, and ICA focuses on the statistical dependence. The semi-supervised methods, such as semi-supervised universum, employ both supervised and unsupervised information to learn meaningful features to improve model performance [11, 26, 57].

2.2 Deep Learning Based Feature Representation Learning

Deep learning is a kind of hierarchical learning method, which can be categorized into supervised, semi-supervised, or unsupervised methods [5]. The main feature of deep-learning methods, such as CNN and LSTM deep-learning model, is utilizing the spatial/temporal correlation to obtain meaningful features without domain expertise. Deep learning has made revolutionary progress in fields such as computer vision, speech recognition, audio recognition, and so on, where the performance is comparable to or even superior to human beings in some scenarios [15, 36]. Compared to

traditional machine-learning methods, the major advantage of deep learning is the power of feature learning. More specifically, deep-learning method can learn meaningful feature from the unstructured data without human effort in feature design, whereas the traditional machine-learning methods require hand-engineered features involving significant labor and financial costs.

In deep learning, there are two types of commonly used deep-learning networks, CNN and RNNs. CNN is a popular deep feed-forward artificial neural networks model, which is specially designed for the data represented in array (or in a high order tensor) form and have been popularly used in the domain of image/video/audio/speech [41]. RNNs focus on data represented in the sequential format (*e.g., audio and speech*) to learn effective features to represent data by learning long-term dependencies. To address the long-term storage problem of RNN, LSTM utilizes memory cells to improve the performance [7, 23, 30]. In addition to videos, images, and texts, research has also advanced the deep learning to networked data, such as social networks, to learn feature representation for networks [58].

2.3 Differentiation from the Proposed Work

In conclusion, feature learning is to find informative and discriminating features to represent the initial data. Feature learning methods extract/learn/select good/new, informative and discriminating, features for better classification. Specifically, feature selection and extraction based data representation aims to extract/select good features for better classification, where the original feature space are transformed into a subspace with low dimension to represent the data. Deep-learning architectures utilizes spatial/temporal correlation of the data to obtain new features, which do not exist in the original feature space.

The two data representation methods described in Sections 2.1 and 2.2 are designed for different types of input data. Specifically, feature selection and extraction based methods typically take generic data with weak or no correlation as input, whereas deep learning based methods are specially designed for data with temporal and/or spatial correlation.

In general, deep learning based methods are more popular and effective than feature selection and extraction based methods. On one hand, research shows that deep learning based methods outperform feature selection and extraction based methods in terms of accuracy metric [6]. On the other hand, feature selection and extraction based methods heavily reply on hand-selected features by experts, requiring strong domain knowledge and data understanding, whereas deep learning based methods focus on learning high-level features from the original data, which reduce the task of developing new feature extractor for learning problems. Unfortunately, we can not directly apply deep learning based methods to generic data, because features of generic data have weak or no correlation.

To handle the problem described above, our research proposes a solution to represent generic data, and makes LSTM available to the generic dataset to improve the classification accuracy. Different from the above feature learning methods, our research goal is not to extract/learn/select good features from the given data, but to find an effective way to represent original instances. Specifically, the output of GeLSTM reorders the original features and remains the original features. As far as we know, there is no existing research focusing on making LSTM available to generic dataset classification task. Accordingly, our work intends to open a new paradigm to allow LSTM module to be broadly applied to generic machine-learning datasets.

3 PROBLEM DEFINITION AND SYSTEM OVERVIEW

3.1 Problem Definition

Considering a generic dataset $\mathcal{D}$, represented in an instance-feature tabular format with n instances and m features, we denote the k^{th} instance as $\boldsymbol{x_k} = \{x_{k,1}, \ldots, x_{k,m}; y_k\}$, where $x_{k,i}$ and

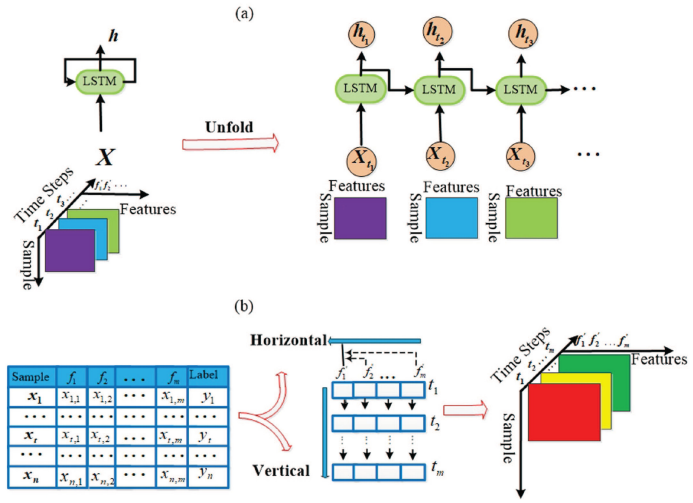

Fig. 4. (a): A conceptual view of the LSTM network. Left panel shows an LSTM network with three-dimensional tensor input (*i.e., [samples, time steps, features]*), and the right panel shows the unfolded LSTM where each LSTM unit corresponds to one time point; and (b) a conceptual review of the proposed GeLSTM which converts a generic dataset (left panel) into three dimensional tensor input (right panel) using horizontal-vertical feature alignment (middle panel).

y_k denote the i^{th} feature and label of the instance $\boldsymbol{x_k}$ accordingly. The **aim** of GeLSTM is to find an LSTM compatible representation of instance $\boldsymbol{x_k}$, denoted by $\mathcal{F}(\boldsymbol{x_k})$. By doing so, LSTM can directly utilize $\mathcal{F}(\boldsymbol{x_k})$ to learn classifiers with a better classification accuracy, compared to classifiers trained from the original feature space.

It is worth noting that our research goal is not to extract/learn/select good features from the given data, but to find an effective way to represent original instances such that LSTM can be directly applied to train deep-learning classifiers from generic machine learning tasks. Therefore, our solutions and experiments are large different from the existing feature extraction/selection research [42, 55].

In Figure 4(a), we briefly show the LSTM structure and its unfolded units, where input data of LSTM are three-dimensional tensor $\boldsymbol{X}$ (*i.e., [samples, time steps, features]*). On the right panel, the LSTM loop structure is unfolded into multiple LSTM units along the time axis, so the input data at time step t_i is $\boldsymbol{X_{t_i}}$ and the output of the last LSTM unit is return to the input of the current LSTM unit.

Since it is the temporal/sequential correlation of the data that LSTM utilizes to obtain meaningful features, we propose a GeLSTM method to convert each instance $\boldsymbol{x_k}$ into a synthetic sentence to create "artificial sequential correlation data." Specifically, as shown in Figure 4(b), for each instance in the original feature space (left panel), our conversion (middle panel) first seeks to horizontally align original features into a sequentially correlated feature vector, resembling to the letter coherence within each single word. In addition, a vertical alignment is also carried out to create multiple time steps and simulates word correlation within a sentence. The two-dimensional horizontal-and-vertical alignments converts the original dataset into a three-dimensional tensor input (right panel) for LSTM model to learn effective features.

3.2 Overall Framework of GeLSTM

The workflow of applying the proposed GeLSTM method to the generic dataset is shown in Figure 5. Overall, GeLSTM includes the following three major steps:

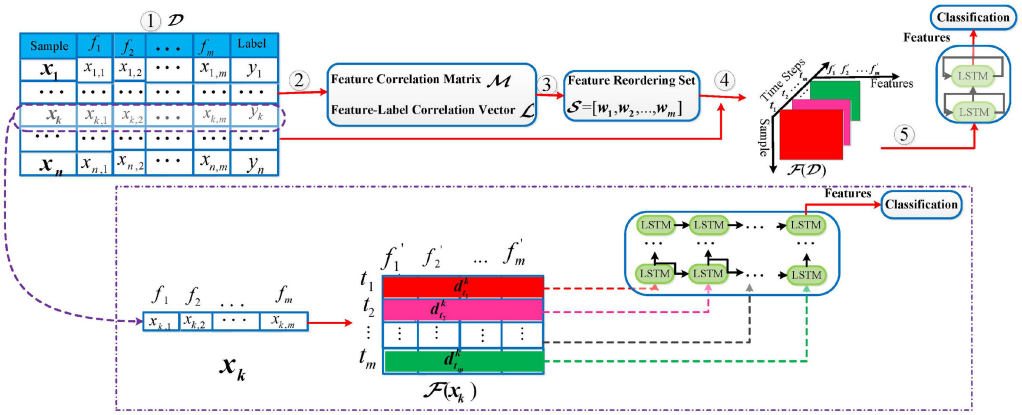

Fig. 5. An overview of the proposed GeLSTM which generalizes LSTM for deep learning from generic data. For a generic dataset represented in instance-feature tabular format in ①, GeLSTM first calculates feature-feature correlation matrix and feature-label correlation vector ②. Next, GeLSTM builds a feature reordering set ③, and converts the original two-dimensional data dataset into three-dimensional tensor ④ which is fed into LSTM module ⑤ for learning features for classification. Lower panel (dashed-line box) shows the conversion of instance $\boldsymbol{x_k}$ into a synthetic sentence format with m words $\mathcal{F}(\boldsymbol{x_k}) = [\boldsymbol{d}^{\boldsymbol{k}}_{t_1}, \boldsymbol{d}^{\boldsymbol{k}}_{t_2}, \ldots, \boldsymbol{d}^{\boldsymbol{k}}_{t_m}]$. The i^{th} word $\boldsymbol{d}^{\boldsymbol{k}}_{t_i}$ is fed into the unfolded LSTM unit at time step t_i.

- **Feature-feature correlation matrix and feature-label correlation vector acquisition:** We utilize Pearson correlation and chi-square correlation to acquire a pairwise feature-feature correlation matrix and a correlation vector, denoted by $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{L}}$, respectively.
- **Feature reordering set construction:** To build LSTM compatible instance representation and remain each instance's original features, we utilize correlation matrix $\boldsymbol{\mathcal{M}}$ and correlation vector $\boldsymbol{\mathcal{L}}$ to construct a feature reordering set $\boldsymbol{\mathcal{S}} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}]$, through a two dimensional horizontal-and-vertical alignment, where $\boldsymbol{w_i} = [w_{i,1}, w_{i,2}, \ldots, w_{i,m}] \in \mathcal{R}^{1\times m}$ stands for the i^{th} feature reordering vector in $\boldsymbol{\mathcal{S}}$ and $\mathcal{R}$ is the spaces real-valued numbers spaces. $\boldsymbol{w_i}$ indicates each instance converting its initial features into the i^{th} word of the synthetic sentence.
- **New presentation of instances generation:** By using feature reordering set $\boldsymbol{\mathcal{S}}$, GeLSTM reorders original feature values of instance $\boldsymbol{x_k}$ and converts instance $\boldsymbol{x_k}$ into a synthetic sentence format $\mathcal{F}(\boldsymbol{x_k}) = [\boldsymbol{d}^{\boldsymbol{k}}_{t_1}, \boldsymbol{d}^{\boldsymbol{k}}_{t_2}, \ldots, \boldsymbol{d}^{\boldsymbol{k}}_{t_m}]$, where $\boldsymbol{d}^{\boldsymbol{k}}_{t_i} = [d^k_{t_i,1}, d^k_{t_i,2}, \ldots, d^k_{t_i,m}] \in \mathcal{R}^{1\times m}$ denotes the i^{th} word in the synthetic sentence $\mathcal{F}(\boldsymbol{x_k})$, which will be fed into the LSTM network at time step t_i.

Based on the above three steps, an original instance will be converted into a synthetic sentence. Then, the LSTM can be utilized to learn features from the converted synthetic sentence for performance improving.

4 GELSTM ALGORITHM

Algorithm 1 briefly describes the proposed GeLSTM algorithm. In the following, we will discuss how to construct correlation matrix $\boldsymbol{\mathcal{M}}$, correlation vector $\boldsymbol{\mathcal{L}}$, and feature reordering set $\boldsymbol{\mathcal{S}}$, followed by an example showing the process of using GeLSTM algorithm to convert instances for LSTM learning.

ALGORITHM 1: GeLSTM Algorithm

Input:
$\mathcal{D}$: A generic dataset;
Output:
The converted synthetic sentence format $\mathcal{F}(\mathcal{D})$;
1: $\mathcal{M} \leftarrow$ Construct feature-feature correlation matrix;
2: $\mathcal{L} \leftarrow$ Construct feature-label correlation vector;
3: Feature reordering set $\mathcal{S} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}]$ construction:
(a) $\mathcal{S} = \emptyset$;
(b) $h \leftarrow$ Obtain the index of the feature that has the strongest correlation with the class label according to feature-label correlation vector $\mathcal{L}$;
(c) $\mathcal{V} \leftarrow$ Sort features in a descending order according to their correlation to the h^{th} feature by utilizing feature-feature correlation matrix $\mathcal{M}$;
(d) $\boldsymbol{w_1} = \mathcal{V}$: Determine the elements in the feature reordering vector $\boldsymbol{w_1}$;
(e) **for each** $i \in [2, \ldots, m]$ **do**
(i) **for each** $j \in [1, 2, \ldots, m]$ **do**
$w_{i,j} \leftarrow$ Apply $\mathcal{M}$ to obtain the j^{th} element in vector $\boldsymbol{w_i}$ using Equation (6).
(ii) **end**
(f) **end**
4: **for all** $\boldsymbol{x_k} \in \mathcal{D}$ **do**
5: $\mathcal{F}(\boldsymbol{x_k}) \leftarrow$ Algorithm 2 $(\boldsymbol{x_k}, \mathcal{S})$
6: **end for**
7: **return** $\mathcal{F}(\mathcal{D})$

4.1 Feature-feature Correlation Matrix $\mathcal{M}$ and Feature-label Correlation Vector $\mathcal{L}$

To obtain higher order feature correlation, we first utilize Pearson correlation [29] to obtain the feature-feature correlation matrix $\mathcal{M}$, to capture pair-wise correlation between features, which can be computed by Equation (1).

$$\mathcal{M}_{i,j} = \frac{\sum_{c=1}^{N} (x_{c,i} - \bar{f}_i)(x_{c,j} - \bar{f}_j)}{\sqrt{\sum_{c=1}^{N} (x_{c,i} - \bar{f}_i)^2} \sqrt{\sum_{c=1}^{N} (x_{c,j} - \bar{f}_j)^2}} \tag{1}$$

ALGORITHM 2: Synthetic sentence $\mathcal{F}(\boldsymbol{x_k})$ of instance $\boldsymbol{x_k}$ generation

Input:
$\mathcal{S} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}]$: The feature reordering set;
$\boldsymbol{x_k}$: The k^{th} instance in the original dataset;
Output:
The synthetic sentence $\mathcal{F}(\boldsymbol{x_k}) = [\boldsymbol{d^k_{t_1}}, \boldsymbol{d^k_{t_2}}, \ldots, \boldsymbol{d^k_{t_m}}]$;
1: $i = 1, j = 1$;
2: **while** $i \neq m$ **do**
3: **while** $j \neq m$ **do**
4: $d^k_{t_{i,j}} = x_{k,w_{i,j}}$: Determine the j^{th} element in the i^{th} word of the synthetic sentence $\mathcal{F}(\boldsymbol{x_k})$;
5: $i = i + 1$
6: **end while**
7: $j = j + 1$
8: **end while**

In Equation (1), $\bar{f}_i$ and $\bar{f}_j$ stand for the sample means of features i and j, respectively, which can be computed by

$$\bar{f}_i = \sum_{c=1}^{N} x_{c,i}, \quad \bar{f}_j = \sum_{c=1}^{N} x_{c,j}. \tag{2}$$

Similarity, in order to capture the correlation between each feature and the class label, we obtain the correlation vector $\boldsymbol{\mathcal{L}}$ by utilizing the Chi-Square method [12, 43]

$$\boldsymbol{\mathcal{L}}_i = \sum_{k \in \boldsymbol{f_i}} \sum_{d \in \boldsymbol{L}} \left(\frac{N_{kd} - E_{kd}}{E_{kd}} \right)^2, \tag{3}$$

where $\boldsymbol{L}$ is the label vector of the dataset, N_{kd} is the observed frequency of feature k and label d appearing simultaneously in the dataset, and E_{kd} is the expected frequency of k and d simultaneously in the dataset. The greater the value of $\boldsymbol{\mathcal{L}}_i$, the stronger the correlation between feature $\boldsymbol{f_i}$ and label $\boldsymbol{L}$.

Note that, we only utilize the magnitude of the correlation to assess the feature-feature correlation and feature-label correlation. Therefore, our calculation, in the remaining part of the article, uses the absolute values of $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{L}}$.

4.2 Feature Reordering Set $\mathcal{S}$ Construction

Once the correlation matrix $\boldsymbol{\mathcal{M}}$ and the correlation vector $\boldsymbol{\mathcal{L}}$ are available, the feature reordering set $\boldsymbol{\mathcal{S}} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}]$ is constructed to create temporal correlation features for LSTM.

The i^{th} element in the feature reordering set $\boldsymbol{\mathcal{S}}$ (*i.e.,* $\boldsymbol{w_i}$) is a feature index vector, which indicates all the instances to convert their initial features into the i^{th} word of their synthetic sentence according to Algorithm 2.

In order to create a "synthetic sentence" with characteristics similar to genuine word correlation and order information, we use two-gram model, which is a generally used model in the natural language modeling [2, 19]. In summary, two-gram model is a model of assigning a probability value to a sentence, indicating how likely the sentence is from that language. In other words, different orders of the words in a sentence correspond to different probability values. The higher the probability value of a sentence, the more likely the sentence belongs to the language. Therefore, the order of words in the synthetic sentence can be determined by maximizing the probability of the synthetic sentence. Furthermore, two-gram model assumes that the current word is associated with the previous word, such that the correlation is added in the synthetic sentence. Thus, our aim is to maximize the following probability to create sequential correlation sentence by utilizing the two-gram model.

$$\begin{aligned} &\underset{\boldsymbol{\mathcal{S}}=[\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}]}{\arg\max} \quad p(\boldsymbol{\mathcal{S}}) = p(\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}) \\ &\overset{(a)}{=} p(\boldsymbol{w_1}) p(\boldsymbol{w_2}|\boldsymbol{w_1}) \ldots p(\boldsymbol{w_m}|\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_{m-1}}) \\ &\overset{(b)}{=} p(\boldsymbol{w_1}) p(\boldsymbol{w_2}|\boldsymbol{w_1}) \ldots p(\boldsymbol{w_m}|\boldsymbol{w_{m-1}}), \end{aligned} \tag{4}$$

In Equation (4), (a) is derived by the chain rule, and (b) relies on the two-gram approximation where the current probability of the word depends on the previous word. $p(w_i|w_{i-1})$ means the probability of w_{i-1} transiting into w_i to construct the sequence correlation.

Finding feature reordering set $\boldsymbol{\mathcal{S}} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}]$ satisfying Equation (4) is equal to find each feature index vector in turn such that

$$\begin{aligned} &\underset{\boldsymbol{w_i}=[w_{i,1}, w_{i,2}, \ldots, w_{i,m}]}{\arg\max} \quad p(\boldsymbol{w_i}|\boldsymbol{w_{i-1}}) \\ \overset{(a)}{\approx} &\underset{\boldsymbol{w_i}=[w_{i,1}, w_{i,2}, \ldots, w_{i,m}]}{\arg\max} \prod_{k=1}^{m} \mathcal{M}_{w_{i-1,k}, w_{i,k}}, \end{aligned} \tag{5}$$

The proof and Example of (a) is described in Appendix A.

Apparently, since there are m^m cases for $\boldsymbol{w_1}$, it is impossible to derive solutions to solve Equation (5) directly. To make Equation (5) solvable, we add two constraints to Equation (5) as follows:

- (1) In order to make the feature reordering set $\boldsymbol{\mathcal{S}}$ be label-targeting, we utilize the correlation vector $\boldsymbol{\mathcal{L}}$ and correlation matrix $\boldsymbol{\mathcal{M}}$ to determine the elements in $\boldsymbol{w_1}$. Specifically, we first find the index of the feature that has the strongest correlation with the class label according to feature-label correlation vector $\boldsymbol{\mathcal{L}}$, denoted by h. Then, the feature index vector $\boldsymbol{\mathcal{V}} = [\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m]$ is equal to $\text{Sort}(\boldsymbol{\mathcal{M}}_{h,:})$, where Sort stands for returning the corresponding feature indices by sorting vector $\boldsymbol{\mathcal{M}}_{h,:}$ in a descending order. The feature reordering in $\boldsymbol{w_1}$ is $\boldsymbol{\mathcal{V}} = [\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m]$, *i.e.*, $w_{1,j} = \mathcal{V}_j$ $(1 \le j \le m)$.
- (2) In order to ensure the fairness between features and preserve all initial feature values, we make each $\boldsymbol{w_i}$ contain all feature indices but in different order by adding constraints on $w_{i,j}$, which will be described at the end of this Section.

Therefore, the problem in Equation (5) can be transformed as follows.

$$\begin{aligned} &\underset{w_{i,j}}{\arg\max}\, \mathcal{M}_{w_{i-1,j}, w_{i,j}} \\ &\text{s.t. } w_{i,1} \notin [w_{1,1}, w_{2,1}, \ldots, w_{i-1,1}], \\ &\quad w_{i,j} \notin [w_{i,1}, w_{i,2}, \ldots, w_{i,j-1}], \\ &\quad \boldsymbol{w_1} = \boldsymbol{\mathcal{V}}, (1 \le j \le m). \end{aligned} \tag{6}$$

Based on Equation (6), we can derive the elements in $\boldsymbol{w_i}$ $(i = 2, \ldots, m)$ one by one. According to the constraint $w_{i,1} \notin [w_{1,1}, w_{2,1}, \ldots, w_{i-1,1}]$, we can easily derive that the maximum number of time spans is the number of features of the generic data m. In order to preserve complete information of the original dataset, we utilize m as the number of time spans in our feature reordering mechanism. Actually, the procedure of determining $\boldsymbol{w_1}$ is to horizontally align original features into a sequential correlated feature vector, resembling to the letter coherence within each single word. The procedure of determining $\boldsymbol{w_i}(i = 2, \ldots, m)$ is to vertically align to create multiple time steps and simulates sequential order of words (i.e., word correlation) within a sentence. The two-dimensional horizontal-and-vertical alignments not only ensures the correlation within a signal word but also the correlation among words.

In the following, we briefly explain why the constraints on $\boldsymbol{w_{i,j}}$ (*i.e.,* $w_{i,1} \notin [w_{1,1}, w_{2,1}, \ldots, w_{i-1,1}]$ and $w_{i,j} \notin [w_{i,1}, w_{i,2}, \ldots, w_{i,j-1}]$) can ensure that each $\boldsymbol{w_i}$ contains all feature indices but in different order. The first constraint $w_{i,1} \notin [w_{1,1}, w_{2,1}, \ldots, w_{i-1,1}]$ means that the first element in the feature reordering vector $\boldsymbol{w_i}$ is different from those of the preceding feature reordering vectors. This ensures that the elements in the feature reordering set $\boldsymbol{\mathcal{S}}$ (*i.e.,* $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots \boldsymbol{w_m}$) are different from each other, *i.e.,* $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots \boldsymbol{w_m}$ are in different orders. The second constraint $w_{i,j} \notin [w_{i,1}, w_{i,2}, \ldots, w_{i,j-1}]$ means that the j^{th} element in w_i is different from its preceding $j-1$ elements. This ensures that the $\boldsymbol{w_i}$ contains all feature indices.

4.3 Example: Synthetic Sentence Explanation

In the remaining part of this subsection, an example is utilized to elaborate the following three aspects: (1) why generic data cannot be directly used for LSTM; (2) how to convert a generic

Table 1. Toy Example Demonstrating: (a) Toy Machine-Learning Dataset Contains Six Instances $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, Four Features $\{f_1, f_2, f_3, f_4\}$ and a Binary Class Label $\{1, 0\}$; (b) The Pair-wise Feature Correlation Matrix $\mathcal{M}$ of the Toy Dataset, Calculated Using Equation (1)

(a)

Instance	f_1	f_2	f_3	f_4	Label
x_1	0.3	0.7	0.6	0.1	1
x_2	0.4	0.72	0.3	0.9	0
x_3	0.13	0.55	0.3	0.4	0
x_4	0.22	0.42	0.14	0.25	1
x_5	0.34	0.51	0.48	0.79	0
x_6	0.28	0.37	0.59	0.32	1

(b)

Feature	f_1	f_2	f_3	f_4
f_1	1.0	0.43	0.32	0.58
f_2	0.43	1.0	0.07	0.27
f_3	0.31	0.07	1.0	0.19
f_4	0.58	0.27	0.19	1.0

dataset into synthetic sentence for LSTM module; and (3) why the feature reordering can mimic real word sequential data. Table 1(a) presents a toy dataset of six instances, four features, and two class labels. Let's first discuss why generic data cannot be directly used for LSTM.

4.3.1 Why Generic Data cannot be Directly Used for LSTM. Since LSTM is specially designed for sequential data, the inputs of LSTM should be more like sentences. LSTM relies on correlation in the sentence to learn new feature representation to represent data for better classification. In our problem settings, instances are represented in the form of instance-feature table with weak or no correlation. The proposed GeLSTM algorithm converts each original instance into a "synthetic sentence" format by using two-gram model which assigns a probability value to a sentence, and ensures that the correlation and order information are added in the synthetic sentence, such that LSTM can be utilized as the deep learning model. The higher the probability value of a sentence, the more likely the sentence is from that language.

In the following, we use the toy dataset as an example, and compare the probability values between the initial sentence and that of the synthetic sentence. Our goal is to show that by taking correlation into consideration to rearrange instances, we can convert original instance into a format better fit the LSTM learning.

For simplicity, we consider a simple case that a word only corresponds to one feature and demonstrate why generic data cannot be directly used for LSTM (In our proposed feature reordering mechanisms, each word actually considers more features).

Based on Equation (5)(a), the probability value of the initial sentence (without taking the ordering and correlation into account) of the toy dataset is

$$\begin{aligned} p(\mathcal{S}) &= p(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4) = p(\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4) \\ &= p(\boldsymbol{f}_1)p(\boldsymbol{f}_2|\boldsymbol{f}_1)p(\boldsymbol{f}_3|\boldsymbol{f}_2)p(\boldsymbol{f}_4|\boldsymbol{f}_3) \approx 1 \times 0.43 \times 0.07 \times 0.19 = 0.005719. \end{aligned} \tag{7}$$

Alternatively, if we set the first word of the sentence to $\boldsymbol{f}_1$, and utilize two-gram model to obtain the synthetic sentence ($[\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4] = [\boldsymbol{f}_1, \boldsymbol{f}_4, \boldsymbol{f}_2, \boldsymbol{f}_3]$) with sequential correlation, we have

$$\begin{aligned} p(\mathcal{S}') &= p(\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4) = p(\boldsymbol{f}_1, \boldsymbol{f}_4, \boldsymbol{f}_2, \boldsymbol{f}_3) \\ &= p(\boldsymbol{f}_1)p(\boldsymbol{f}_4|\boldsymbol{f}_1)p(\boldsymbol{f}_2|\boldsymbol{f}_4)p(\boldsymbol{f}_3|\boldsymbol{f}_4) \approx 1 \times 0.58 \times 0.43 \times 0.07 = 0.017458. \end{aligned} \tag{8}$$

The results from Equations (7) and (8) show that the probability value of the initial sentence $(\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4)$ is much lower than that of the synthetic sentence $(\boldsymbol{f}_1, \boldsymbol{f}_4, \boldsymbol{f}_2, \boldsymbol{f}_3)$. This means that, compared to the initial sentence $(\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4)$, the synthetic sentence $(\boldsymbol{f}_1, \boldsymbol{f}_4, \boldsymbol{f}_2, \boldsymbol{f}_3)$ is more like a real sentence. The reason is that, compared to the the generic data (without taking the ordering and correlation into account), the two-gram model ensures that the correlation and order information are added in the synthetic sentence, such that the synthetic sentence is more like a

genuine sentence and LSTM can be utilized to learn new feature representation. Therefore, generic data may not be directly used for LSTM, but should be transformed into synthetic sentence, which is described in the following subsection, to support deep learning.

4.3.2 How to Convert a Generic Dataset into Synthetic Sentence for LSTM Module. In order to convert the toy dataset in Table 1 into synthetic sentence format, GeLSTM first build the correlation matrix $\mathcal{M}$, which is shown in Table 1(b), and correlation vector $\mathcal{L} = [0.003, 0.0257, 0.0259, 0.731]$. Then, GeLSTM finds $h = 4$, *i.e.,* the index of the feature with the strongest correlation to the class label according to $\mathcal{L}$. The feature index vector $\mathcal{V} = [4, 3, 2, 1]$ equals Sort($\mathcal{M}_{h,:}$), where Sort stands for returning the corresponding feature indices by sorting vector $\mathcal{M}_{4,:}$ in a descending order. Thus, the first feature reordering vector $\boldsymbol{w_1}$ is $[4, 1, 2, 3]$.

Next, by substituting $\mathcal{M}$ and $\boldsymbol{w_1}$ into Equation (6), we derive elements in $\boldsymbol{w_2}$ one by one (*i.e.,* $\boldsymbol{w_2} = [1, 4, 2, 3]$). Then, by substituting $\mathcal{M}$ and $\boldsymbol{w_2}$ into Equation (6), $\boldsymbol{w_3} = [2, 4, 1, 3]$ can be further obtained. Finally, we can derive $\boldsymbol{w_4} = [3, 4, 1, 2]$ using the same way. As a result, the feature reordering set is $\mathcal{S} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \boldsymbol{w_4}] = [[4, 1, 2, 3], [1, 4, 2, 3], [2, 4, 1, 3], [3, 4, 1, 2]]$.

Following Algorithm 2, the constructed synthetic sentence of instance $\boldsymbol{x_1}$ is as follows,

$$\begin{aligned}\mathcal{F}(\boldsymbol{x_1}) &= \left[d^1_{t_1}, d^1_{t_2}, d^1_{t_3}, d^1_{t_4}\right] \\ &= [[0.1, 0.3, 0.7, 0.1], [0.3, 0.1, 0.7, 0.6], [0.7, 0.1, 0.3, 0.6], [0.6, 0.1, 0.3, 0.3, 0.7]].\end{aligned} \tag{9}$$

Finally, we discuss why the feature reordering can mimic real word sequential data.

4.3.3 Rationality of Feature Reordering to Mimic Real-word Sequential Data. According to Equation (5), the probability value of the synthetic sentence of the toy dataset is

$$\begin{aligned}p(\mathcal{S}) &= p(\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \boldsymbol{w_4}) \\ &= p(\boldsymbol{w_1})p(\boldsymbol{w_2}|\boldsymbol{w_1})p(\boldsymbol{w_3}|\boldsymbol{w_2})p(\boldsymbol{w_4}|\boldsymbol{w_3}) \approx 0.003.\end{aligned} \tag{10}$$

If we employ random ordering to obtain the feature reordering set $\mathcal{S}$, where each feature reordering vector $\boldsymbol{w_i}$ includes all feature indices, the random feature reordering set is $\mathcal{S}' = [\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \boldsymbol{w_4}] = [[1, 3, 2, 4], [2, 4, 1, 3], [2, 3, 1, 4], [4, 3, 2, 1]]$. Similarly, the probability of the random sentence of the toy dataset is

$$\begin{aligned}p(\mathcal{S}') &= p(\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \boldsymbol{w_4}) \\ &= p(\boldsymbol{w_1})p(\boldsymbol{w_2}|\boldsymbol{w_1})p(\boldsymbol{w_3}|\boldsymbol{w_2})p(\boldsymbol{w_4}|\boldsymbol{w_3}) \approx 1.62 \times 10^{-5}.\end{aligned} \tag{11}$$

The calculations from Equations (10) and (11) show that the sentence derived from our proposed feature reordering mechanism much more resembles to real-world sequential data. Furthermore, the random feature reordering mechanism uses random order to create a synthetic sentence, without taking correlation into consideration. In the next Section (Section 5.4), our experiments will show that the proposed feature reordering mechanism (*i.e., GeLSTM*) outperforms random feature reordering mechanism (*i.e., RoLSTM*), which means that constructing correlation by reordering features is preferable for LSTM.

5 EXPERIMENTS

In this section, we validate the proposed GeLSTM in terms of its effectiveness of enabling LSTM learning for generic datasets. In the experiments, we implement the proposed framework and baselines using Tensorflow [1], and mainly report the effectiveness of the GeLSTM enabled deep-learning classifiers, and compare their accuracy with traditional machine learning classifiers, including support vector machine (SVM), decision trees (DT), random forests (RF), and the like. We did not compare GeLSTM with any feature extraction/selection methods [42, 55], because our research goal is not to extract/learn/select good features from the given data, but to find effective

Table 2. A Brief Introduction of the 20 Benchmark Datasets

ID	Datasets	number of Instances	number of Features	number of Classes
1	vehicle	946	18	4
2	vowel	990	9	6
3	breast cancer wisconsin (Diagnostic)	569	30	2
4	wine	178	13	3
5	banknote authentication	1372	4	2
6	vertebral column	310	6	2
7	yeast	1484	8	10
8	seeds	210	7	3
9	climate model simulation crashes	540	18	2
10	glass identification	214	9	6
11	plrx	182	12	2
12	pima	768	9	2
13	iris	150	4	3
14	wireless indoor localization	2000	7	4
15	sonar	208	60	2
16	LSVT voice rehabilitation	126	309	2
17	parkinsons	197	22	2
18	fertility	100	9	2
29	waveform	5000	21	3
20	leaf	340	14	30

ways to represent original instances for training deep learning classifiers. If needed, one can always employ feature selection/extraction methods prior to apply GeLSTM to their data to enable deep learning.

We compare the baseline and the proposed GeLSTM method on 20 machine-learning benchmark datasets from UCI data repository [46]. The 20 benchmark datasets cover a variety of data characteristics (*e.g.*, the number of classes vary from 2 to 30, and the number of features vary from several to several hundreds). We intentionally select such diverse datasets to ensure that our approach is effective for different kinds of generic datasets. For UCI benchmark datasets, we assume that data examples are independent and identically distributed (*i.i.d.*), such that the model trained from the training dataset can be evaluated on the test set. Meanwhile, we perform a case study by utilizing time series dataset from the UCR Time Series Classification Archive dataset [14], to explain features learned from GeLSTM converted sentence. Table 2 presents a brief description of the 20 benchmark datasets used in the experiments.

In our experiments, we use cross validations to validate the algorithm performance. For small datasets, more splits mean fewer samples in the test set for each fold. Considering several small datasets in the experiments, we carry out the experiments using five-fold cross validation, and report the classification accuracy in the article.

5.1 Experimental Settings

In the experiments, we take the output of the last time step of the LSTM network as the learned features. Then, the learned features are (fully connected) fed to the output layer with m nodes and a softmax activation function is used to produce the correctly normalized probability values [18]. Furthermore, we use the Adam optimizer [34] and set the learning rate to 0.001 at the training process.

5.2 Baseline Methods

To the best of our knowledge, there is no work on generalizing LSTM for generic datasets. Therefore, we implement two baseline approaches, random feature reordering LSTM (RoLSTM) and label-feature correlation feature reordering LSTM (LoLSTM), to validate the effectiveness of the feature reordering method of GeLSTM. The difference between the two baselines and GeLSTM is the ways of constructing feature reordering set $\mathcal{S}$.

RoLSTM: utilizes random ordering to obtain the feature reordering set $\mathcal{S}$, where each vector $\boldsymbol{w_i}$ of $\mathcal{S}$ includes all feature indices. The random ordering results in the constructed synthetic sentence with no correlation. If GeLSTM can achieve performance gain, compared to RoLSTM, this means that constructing correlation by reordering features is preferable for the LSTM.

LoLSTM: This method obtains the set $\mathcal{S}$ according to the feature–feature correlation matrix and feature-label correlation vector without taking sequential correlation into account. In other words, each feature reordering vector $\boldsymbol{w_i}$ includes all feature indices. We utilize the index of feature that has the ith strongest correlation with the class label according to feature-label correlation vector $\mathcal{L}$ as the first element in $\boldsymbol{w_i}$. The other $m-1$ elements in $\boldsymbol{w_i}$ are the indices of features in a descending order according to their correlation to the first feature by utilizing feature–feature correlation matrix $\mathcal{M}$. Obviously, LoLSTM only considers a local correlation within word, while GeLSTM considers both global (*i.e.*, temporal) and local correlation.

5.2.1 Conventional Machine-Learning Classifiers. In order to validate the benefit of using GeLSTM for generic machine-learning tasks, some popular machine-learning methods are considered as the baseline. The methods used include k-nearest neighbors algorithm (k-NN), dense neural network (NN), SVM, DT, RF, XGBoost (XG) which is completed by using the sklearn module in Tensorflow [47]. If GeLSTM shows better performance than conventional machine learning algorithms, it will confirm that enabling LSTM learning for generic *i.i.d.* datasets is useful for improving generic machine-learning tasks using deep learning.

- ***k*-NN** aims to find k nearest neighbors from the training set for each test instance [4], and labels the instance as the class that the most k nearest neighbors agree.
- **Support Vector Machines (SVM)** constructs a hyperplane by utilizing kennel function [17]. Then, the original instances are mapped into high-dimensional space and labeled as the class according to the side they falling to.
- **Decision Tree (DT)** builds a decision tree to classify the instance. The leaves and branches of the decision tree stand for the labels and conjunctions of features accordingly [10].
- **Random Forest (RF)** is an ensemble learning method for classification. It relies on multiple decision trees, constructed from random feature subspaces, and combines different trees to generate final outputs [33].
- **Multilayer Neural Network (NN)** is a kind of multi-layered neural network, including input layer, hidden layer(s), and output layer [16]. The output layer outputs the predicted results. In our experiments, we use dense networks for training and classification. Because, in theory, two hidden layers can approximate arbitrary shaped decision boundaries to achieve sufficiently high classification accuracy (with suitable learning rates), more hidden layers are unnecessary and require much more training time [31]. A dense network with one hidden layer (denoted by NN-1) or two hidden layers (denoted by NN-2) is utilized in our experiments.
- **XGBoost (XG)** is a DT-based ensemble machine-learning algorithm, and is commonly used for classification and regression tasks [13].

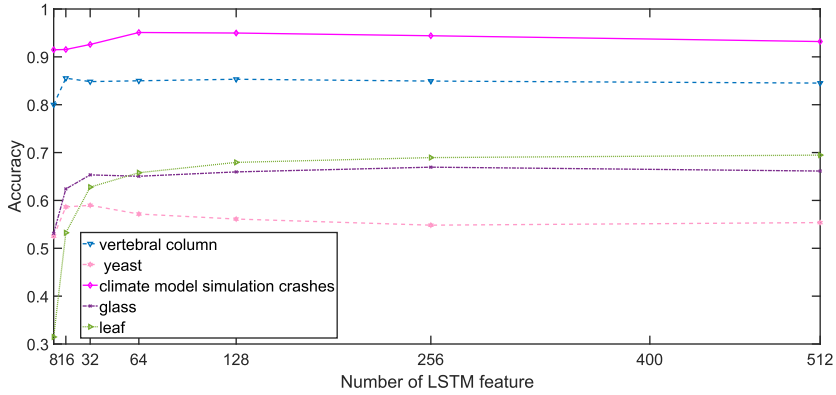

Fig. 6. Accuracy comparisons *w.r.t.* different number of LSTM output features, where the x-axis denotes the number of learned features, varying from 8 to 512, and the y-axis denotes the accuracy reported in decimal numbers.

5.2.2 Deep-Learning Classifiers. To validate the performance of deep-learning classifiers on generic machine-learning tasks, we mainly use LSTM as the deep-learning baseline. Meanwhile, to demonstrate that GeLSTM is also useful for other deep-learning algorithms, we also apply CNN and attention-based LSTM (A-LSTM) as the deep-learning models. For all these deep-learning classifiers, we use GeLSTM to convert original instances into "synthetic sentence" format, and then use the sentences to train deep-learning classifiers.

- **Convolutional Neural Network (CNN)** uses convolutional layers to explore spatial/temporal adjacency to construct new feature representation, which is the go-to method for any type of prediction problem involving image data as an input [40].
- **Attention-based LSTM (A-LSTM)** obtains the performance gain of the LSTM network by utilizing the attention-based models, so A-LSTM can concentrate on different parts of a sentence for learning [53].

In the following subsections, we first study classification accuracy of the GeLSTM with respect to different number of learned features. After that, we compare different feature reordering method in terms of the classification accuracy. Next, we report the classification accuracy of all benchmark datasets. Finally, a case study is performed to validate the effectiveness of applying the GeLSTM for LSTM learning.

5.3 Results *w.r.t.* Different Number of LSTM Features

In our experiments, we fix the number of LSTM layers to two, and take the output of the final LSTM network as the learned features. In Figure 6, we report the performance of GeLSTM *w.r.t.* different number of learned features on five datasets, where the x-axis denotes the number of learned features and the y-axis denotes the accuracy. The results show that increasing the number of learned features from 8 to 32 will continuously improve the classification accuracy, but increasing the feature size further (from 32 to 512) does not impose additional performance gain and the results fluctuate within a small range.

5.4 Feature Reordering Method Comparisons

For all experiments in the following, we utilize the parameter settings below. We set the number of LSTM layers to 1 and 2, and the hidden layer sizes 20 and 64.

Table 3. Classification Accuracy Comparisons between GeLSTM and Different Feature Reordering Methods on the 20 Benchmark Datasets, where the Digit 1 and 2 Denote the Number of LSTM Layers, and the Hidden Layer Sizes is Set to 20 (the Results are Reported in Percentage)

Dataset	RoLSTM		LoLSTM		GeLSTM	
	1	2	1	2	1	2
vehicle	60.18	63.01	60.96	60.14	62.97	64.23
vowel	83.46	83.67	83.85	85.36	86.66	87.63
breast cancer wisconsin (Diagnostic)	96.61	96.96	96.27	96.52	96.62	96.57
wine	96.09	95.9	95.57	95.16	97.18	96.07
banknote authentication	99.91	99.95	99.93	99.96	99.93	99.94
vertebral column	80.42	83.55	83.09	85.23	83.48	84.84
yeast	57.77	58.04	57.61	58.5	58.64	59.13
seeds	94.24	94.29	94.19	93.76	95.14	94.95
climate model simulation crashes	91.48	91.48	91.48	91.48	91.52	91.78
glass identification	60.33	60.79	59.2	60.04	62.83	63.5
plrx	71.44	71.44	71.44	71.44	71.44	71.44
pima	76.79	76.64	76.72	76.51	76.43	76.23
iris	96.93	96.47	97	96.4	96.73	96.47
wireless indoor localization	98.15	98.14	98.07	98.14	98.16	98.2
sonar	77.72	79.13	79.56	78.26	78.08	80.91
LSVT voice rehabilitation	83.01	83.67	82.42	84.51	84.21	83.38
parkinsons	87.15	86.7	89.68	83.07	88.15	89.63
fertility	86.07	86.12	86.07	84.02	87.12	84.12
waveform	85.18	85.12	85.88	85.88	86.68	86.26
leaf	50	52.02	50.33	49.84	56.97	55.53
Average Accuracy	81.65	82.15	81.97	81.7	82.95	83.04

Tables 3–4 compare different feature reordering methods, with respect to different number of LSTM layers and hidden nodes. Specifically, Table 3 shows the classification accuracy performance for different feature reordering methods on 20 benchmark datasets, where the number of LSTM layers and hidden nodes are (1,2) and 64. The results of one and two LSTM layers with 64 hidden nodes are reported in Table 4.

We can see from Tables 3 and 4 that GeLSTM outperforms other feature reordering methods for different parameter settings. This indicates that, for the generic dataset, creating sequential correlation by reordering features is preferable for the LSTM to produce good classification accuracy. In other words, it is the sequential correlation that the LSTM utilizes to obtain effective features. While LoLSTM only considers correlation within each signal word, RoLSTM ignores any correlation in the synthetic sentence.

The results in Table 3 shows that, compared to RoLSTM and LoLSTM, the average accuracy gains of GeLSTM are 1.3% and 0.98% for one LSTM layer, and 0.89% and 1.34% for two LSTM layer accordingly. The results in Table 4 shows that, compared to RoLSTM and LoLSTM, the average accuracy gains of GeLSTM are 1.23% and 0.84% for one LSTM layer, and 1.09% and 1.04% for two LSTM layer accordingly. In addition, the best accuracy gain of GeLSTM is higher than those of RoLSTM and LoLSTM. Specifically, the best accuracy gain of GeLSTM, RoLSTM and LoLSTM are 84.26% in Table 4, 83.17% in Table 4, and 83.39% in Table 4.

Table 4. Classification Accuracy Comparisons between GeLSTM and Different Feature Reordering Methods on the 20 Benchmark Datasets, where the Digit 1 and 2 Denote the Number of LSTM Layers and the Hidden Layer Sizes is Set to 64 (the Results are Reported in Percentage)

Dataset	RoLSTM		LoLSTM		GeLSTM	
	1	2	1	2	1	2
vehicle	64.82	65.21	64.46	66.26	66.65	65.66
vowel	88.21	90.42	87.81	90.4	92.93	93.15
breast cancer wisconsin (Diagnostic)	97.08	97.01	96.76	96.57	96.48	96.43
wine	95.39	96.03	96.01	95.95	96.95	96.84
banknote authentication	99.96	99.93	99.97	99.91	99.95	99.9
vertebral column	84.58	84.48	84.48	84.77	84.35	85.03
yeast	59.28	58.98	59.74	58.69	58.5	57.64
seeds	94.52	94.52	94.38	93.62	95.33	94.79
climate model simulation crashes	91.59	91.9	91.59	91.58	94	94.09
glass identification	64.14	64.94	64.39	64.43	66.66	66.13
plrx	71.44	71.44	71.39	71.44	71.39	71.44
pima	76.44	76.71	76.36	76.29	76.24	76.22
iris	96.4	96.27	96.6	96.47	96.51	96.07
wireless indoor localization	98.04	98.04	98.1	97.97	98.17	98.14
sonar	80.14	80.25	80.4	81.41	80.94	82.43
LSVT voice rehabilitation	82.77	84.29	84.02	84.58	81.12	83.2
parkinsons	83.58	84.29	89.75	84.58	90.45	89.24
fertility	83.07	84.12	82.17	82.01	86.06	86.97
waveform	85.82	83.28	86.3	86.18	85.12	85.25
leaf	62.9	61.27	63.03	61.24	66.73	66.55
Average Accuracy	83.00	83.17	83.39	83.22	84.23	84.26

The results from Tables 3 and 4 demonstrate that using the feature reordering matrix in GeLSTM to create sequential correlation is the key for LSTM to learn informative features to improve the classification accuracy.

The above results show that, the LSTM structure settings to achieve the best accuracy performance for GeLSTM are as follows: the number of the LSTM layer and hidden nodes are set to 2 and 64 accordingly. Therefore, we use the same LSTM parameter settings in the following subsections.

5.5 Comparisons of Different Learning Methods

Table 5 shows comparisons between CNNs, GeLSTM, A-LSTM and conventional learning algorithms (*i.e.*, k-NN, SVM, DT, RF, and NN) on 20 benchmark datasets, where NN-i indicates that there are i hidden layers in the neural network.

The parameter settings of conventional machine-learning methods in this subsection are described as follows. For k-NN, we set the parameter k to 5. For SVM, we utilize the linear kennel function. For DT, the decision tree is generated by CART (Classification And Regression Tree) algorithm. For NN, the mumber of hidden layers is set to 1 and 2, and the corresponding number of nodes is set to to 16 and (32,16). For XG, we use trees as the booster.

For the GeLSTM deep-learning algorithms, we take the output of the last time step of the LSTM network as the learned features. Then, the learned features are fully connected to the output layer with m nodes and a softmax activation function is used to produce the correctly normalized

Table 5. Accuracy Comparisons between GeLSTM and Some Traditional Machine Learning Methods on Benchmark Datasets (the Results are Reported in Percentage)

Dataset	*k*-NN	SVM	DT	RF	NN-1	NN-2	XG	CNN	GeLSTM	A-LSTM
vehicle	57.2	58.78	63.38	63.93	53.65	65.68	63.26	66.45	65.66	68.04
vowel	92.05	66.88	79.16	90.71	55.78	74.42	86	93.12	93.15	91.06
breast cancer wisc.	96.59	97.47	92.41	95.23	97.03	97.01	95.98	96.15	96.43	96.32
wine	80.5	98.32	91.18	97.54	97.07	98.03	96.34	96.89	96.84	97.19
banknote authent.	99.86	97.96	98.39	99.23	95.67	99.81	99.39	99.94	99.9	99.93
vertebral column	79.06	79.29	80.29	82.97	71.67	80.25	81.93	83.67	85.03	85.16
yeast	56.97	56.74	51.14	57.86	47.04	58.42	60.52	59.34	57.64	56.73
seeds	93.24	93.29	90.29	91.61	91.48	91.48	92.9	92.66	94.79	94.85
climate model sim.	92.76	95.33	90.77	92.39	93.89	95.24	94.53	93.4	94.09	94.68
glass identification	65.48	57.51	67.21	74.51	40.16	58.43	73.75	68.45	66.13	67.55
plrx	62.1	71.44	56.78	66.98	71.44	71.38	60.99	70.92	71.44	70.68
pima	73.76	76.95	70.08	73.87	70.42	76.38	75.84	75.63	76.22	76.42
iris	95.53	96.2	95.07	94.93	95.27	96.27	95.93	96.13	96.07	96.33
wireless indoor local.	98.42	98.01	97.22	97.96	96.57	97.62	98.08	98.01	98.14	98.02
sonar	79.6	76.46	71.1	77.39	80.6	81.78	79.99	82.44	82.43	82.06
LSVT voice rehab.	81.41	83.86	76.22	80.73	84.23	84.58	82.24	81.54	83.2	83.03
parkinsons	92.15	86.85	85.6	88.62	86.24	90.36	89.96	89.55	89.24	89.85
fertility	86.44	88.08	83.12	85.93	88.08	88.08	83.22	83.02	86.97	84.45
waveform	82.08	86.89	75.26	82.5	86.17	86.24	85.09	84.25	85.25	86.18
leaf	67.29	54.98	63.43	69.47	60.72	57.63	68.45	73.12	66.55	71.28
Average Accuracy	81.62	81.06	78.9	83.22	78.16	82.45	83.22	84.23	84.26	84.49

probability values. The CNN deep-learning method contains two convolutional layers of size [32,16] followed by a max pooling layer and the filter size of each convolutional layer is set to 2. Features extracted by CNN are further fed into a two hidden layer NN of size [32,16] to classify test data. For the A-LSTM deep-learning model, we take the output of the last layer of the LSTM as the input of the attention layer, and then the outputs of the attention layer are connected to a softmax layer to produce the correctly normalized probability values.

The results are reported in Table 5, where CNN, GeLSTM, and A-LSTM mean using CNN, LSTM, and attention-based LSTM as the deep learning model, respectively.

Our experiments and comparison on 20 generic datasets show that the performance gain of A-LSTM (84.49%) is marginally better than that of GeLSTM (84.26%), and LSTM (84.26%) also slightly outperforms CNN (84.23%). This is consistent with the literature which shows that attention mechanism can help improve the performance of the LSTM model.

Meanwhile, the results show that LSTM and its subtypes are slightly more suitable to deal with the "synthetic sentence" compared to the CNN, partially because that CNNs are more suitable for images which have spatial correlation, whereas LSTM's recurrent model is more effective for sequences. Because GeLSTM converts original instances as "synthetic sentences," it makes GeLSTM's outputs more suitable for LSTM based deep-learning modules. Meanwhile, because LSTM and its subtypes (including attention-based LSTM) belong to the same type of deep-learning model, and our research focuses on investigating whether the "synthetic sentence" can be utilized by such type of deep-learning model to improve the classification performance, we mainly utilize the LSTM as the deep-learning model for validation.

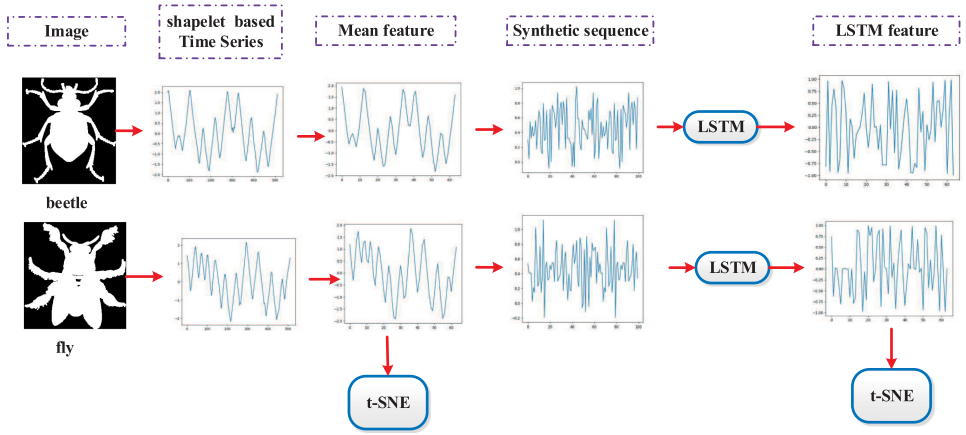

Fig. 7. Comparisons between original time mean features *vs.* LSTM features learned from GeLSTM converted sentence for the instances from different classes ("beetle" *vs.* "fly"). The data on time series curve is obtained from UCR time series archive [14] which are mapped from the outline of the image, and then extract the 64-dimensional mean features from the time series to obtain the data on the mean feature curve. After that, by applying GeLSTM method to convert mean represented instances, data on the synthetic sequence curve is available, which are further fed into the LSTM model to learn LSTM feature. Finally, we employ the *t*-SNE tool [52] to make a comparison between the original mean features and the LSTM learned features to further explain the benefits of our proposed GeLSTM method.

The results from Table 5 show that the performance of the methods of using LSTM and attention-based LSTM after GeLSTM are much higher than those of the conventional machine methods. Specifically, the accuracy gains of CNN are 2.61%, 3.17%, 5.33%, 1.01%, 6.07%, 1.78%, and 1.01% accordingly. The performance gains of GeLSTM are 2.64%, 3.2%, 5.36%, 1.04%, 6.1%, 1.81%, and 1.04%, and the accuracy gains of A-LSTM are 2.87%, 3.43%, 5.69%, 1.27%, 6.33%, 2.04%, and 1.27%, respectively. This means that converting each instance from the generic dataset into a synthetic sentence format by utilizing the GeLSTM, and then apply LSTM or other deep-learning models to the converted data, will improve the classification accuracy for generic dataset.

5.6 GeLSTM Learning Case Study

In this subsection, we comparatively study the original features and features learned from GeLSTM converted synthetic sentence. This study aims to describe what can be learned when feeding the converted synthetic sentence into the LSTM model.

In our experiments, we use two time series datasets (*i.e.,* Beetle/Fly and Bird/Chicken) from the UCR time series archive [14], and extract 64-dimensional mean features as original features to represent each time series. After that, the synthetic sequence represented instances are obtained by applying GeLSTM method to convert mean represented instances, and are fed into the LSTM model to learn LSTM feature. Because these two datasets are mapped from the outlines of Beetle/Fly and Bird/Chicken images accordingly, we report the corresponding images, in the article, to visualize the algorithm performance. It is understandable that mean features might not be the best features to represent time series data. However, our goal in this study is to describe how GeLSTM converted synthetic sentence works for the LSTM model instead of finding effective features for better classification. Hence, we utilize mean features as the original feature in the case study.

The workflow of the case study is shown in Figures 7–10. First, the mean features are extracted from time series which are mapped from the outlines of the corresponding images. Next, the mean

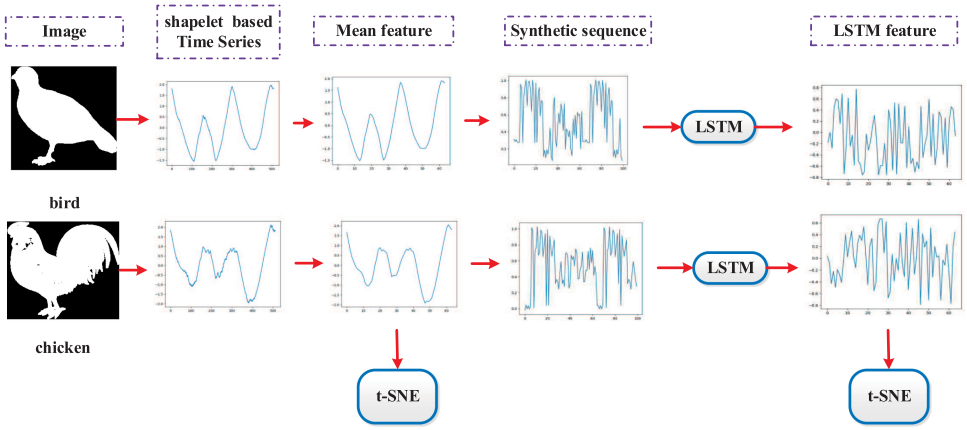

Fig. 8. Comparisons between original time mean features *vs.* LSTM features learned from GeLSTM converted sentence for the instances from different classes ("bird" *vs.* "chicken').

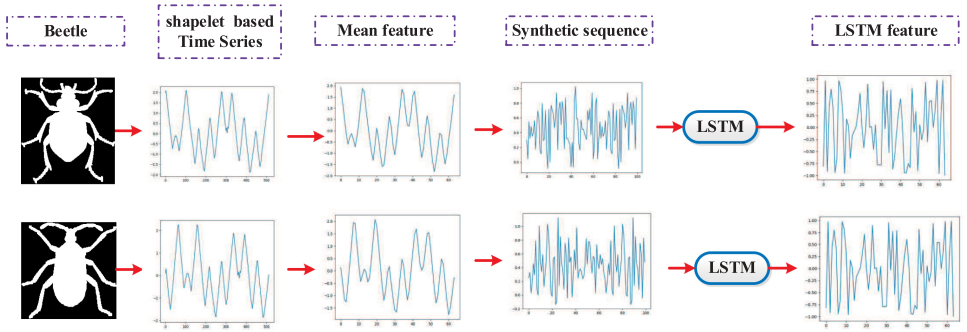

Fig. 9. Comparisons between original time mean features *vs.* LSTM features learned from GeLSTM converted sentences for the instances without image rotation from the same classes ("beetle").

features are fed into our proposed GeLSTM method to output a synthetic sentence. Then, we input the synthetic sentence into an LSTM model to learn new features fit for classification. Finally, the *t*-SNE tool [52] is utilized to visualize the difference between original features and LSTM-learned features (For LSTM model, we use two LSTM layers with 64 hidden nodes).

Figures 7 and 8 show that the original mean features are rather less discriminative to differentiate "Beetle" *vs.* "Fly" in Figure 7 and "Bird" *vs.* "Chicken" in Figure 8, noticing high similarity in mean feature curve. The reason is that the outline of Beetle (Bird) image is similar to that of Fly (Chicken) image. The synthetic sentence, converted by our proposed GeLSTM method, illustrates a better distinctiveness of "Beetle" *vs.* "Fly" in Figure 7 and "Bird" *vs.* "Chicken" in Figure 8. Then, the LSTM utilizes this difference to learn discriminative features to classify "Beetle" *vs.* "Fly" in Figure 7 and "Bird" *vs.* "Chicken" in Figure 8, noticing high discrimination in LSTM feature curve. In other words, the results from Figures 7 and 8 show that, compared to the original mean feature, the features learned by LSTM is more discriminative to differentiate "Beetle" *vs.* "Fly" in Figure 7 and "Bird" *vs.* "Chicken" in Figure 8.

Figures 9–12 show that, the mean feature of instances in the same class vary from each other. After applying GeLSTM to convert each mean represented instance into synthetic sentence, the synthetic sentence shows the similarity among instances in the same class. This is further captured

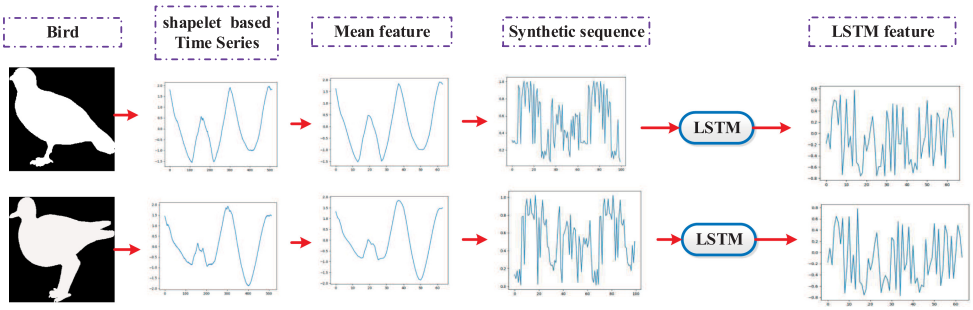

Fig. 10. Comparisons between original time mean features *vs.* LSTM features learned from GeLSTM converted sentences for same class images with rotation ("bird").

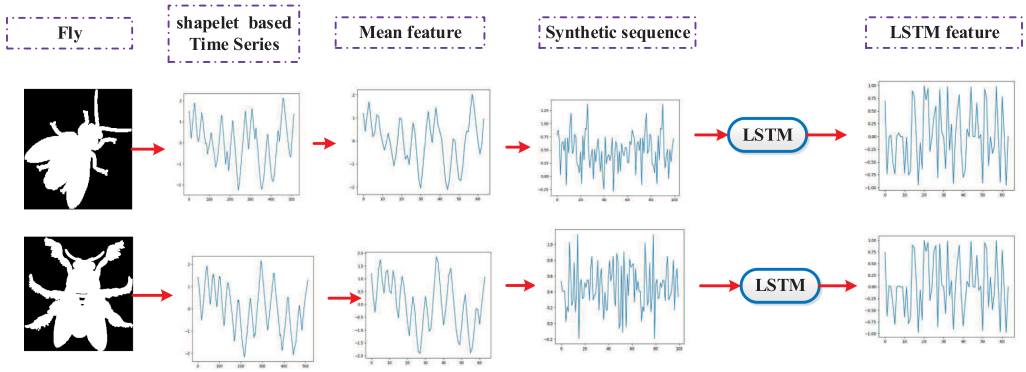

Fig. 11. Comparisons between original time mean features *vs.* LSTM features learned from GeLSTM converted sentence for same class images with rotation ("fly").

by the LSTM model to learn the similarity among instances in the same class. More specifically, Figures 9 and 10 compare the original time mean feature vs. LSTM feature in the same class without image rotation for "beetle" and "bird" accordingly. Through the mean features are different for the same class, the LSTM model learns the similarity from the synthetic sentence represented instance converted by our GeLSTM method (noting the high similarity in LSTM feature curve). In addition, Figures 11 and 12 compare the original time mean feature *vs.* LSTM feature in the same class with image rotation for "fly" and "chicken" accordingly. The synthetic sentence represented instance converted by our GeLSTM method can still preserve the similarity in the same class, which is further captured by the LSTM model to learn the similarity in the same class.

Figures 13 and 14 illustrate the feature representation learning results of original mean features and the GeLSTM converted features for Beetle/Fly and Bird/Chicken, respectively, by utilizing the $t-$SNE tool. The results show that LSTM features have a relatively better discrimination capability, than mean features, to separate two types of samples. This indicates that GeLSTM converted features are more suitable to the LSTM for classification, compared to the original mean features.

Table 6 compare the GeLSTM to different machine-learning methods for "Beetle *vs.* Fly" and "Bird *vs.* Chicken" classification tasks. Compared to KNN, SVM, DT, RF, NN-1, and NN-2, the performance gains of GeLSTM are 40%, 10%, 25%, 25%, 10%, and 5% for "Beetle *vs.* Fly" classification task, and 35%, 25%, 5%, 20%, 20%, and 5% for "Bird *vs.* Chicken" classification task.

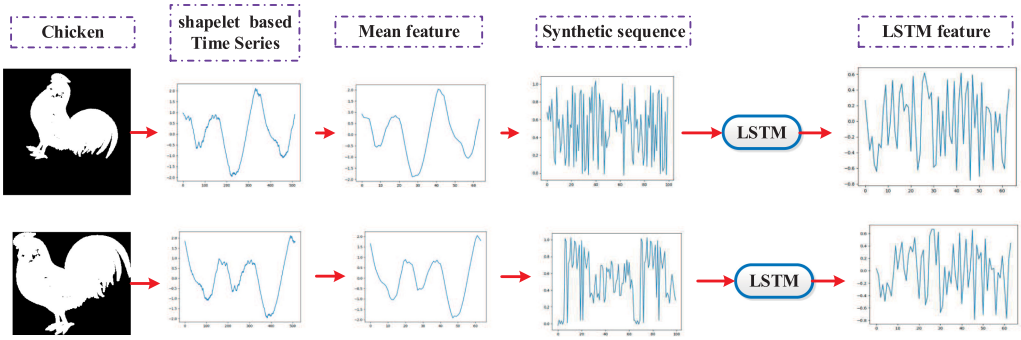

Fig. 12. Comparisons between original time mean features *vs.* LSTM features learned from GeLSTM converted sentences for same class images with rotation ("chicken"). The results show that mean features are sensitive to rotation, whereas LSTM features learned from GeLSTM converged sentences is robust to the rotation, and both images show similar LSTM features.

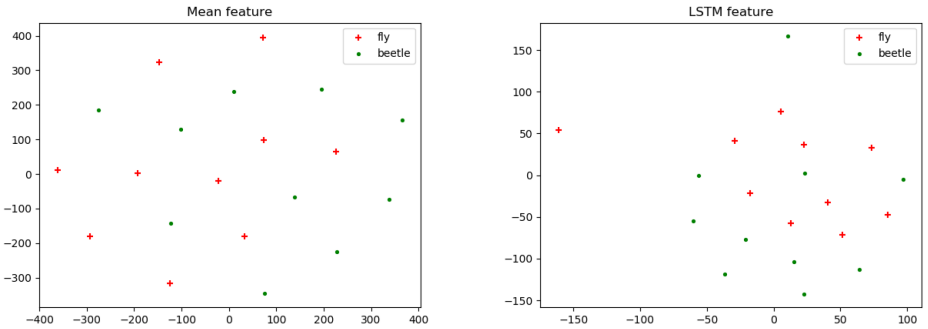

Fig. 13. Comparisons between original features (mean) *vs.* GeLSTM learned LSTM features ("Beetle" *vs.* "Fly"). Instances are color-and-shape coded, according to their labels. The plot on the left panel shows the instances with respect to the original features, and the plot on the right panel shows the GeLSTM learned LSTM features. Both plots are produced using $t-$SNE feature visualization tool [52].

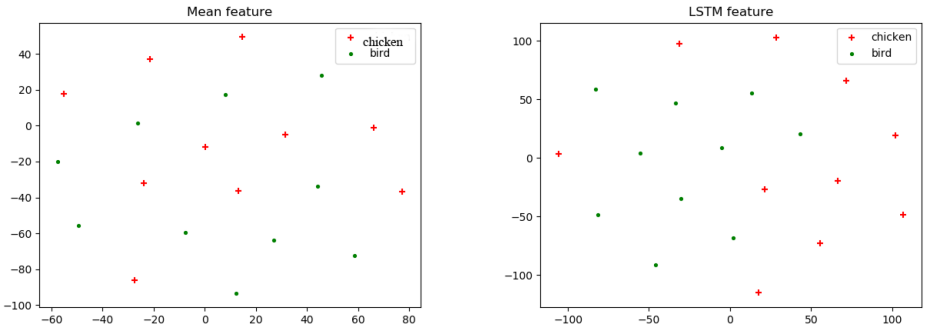

Fig. 14. Comparisons between original features (mean) *vs.* GeLSTM learned LSTM features ("Bird" *vs.* "Chicken"). Similar to Figure 13, both plots are produced using $t-$SNE feature visualization tool.

Table 6. Classification Accuracy Comparisons between Different Learning Methods on the "Beetle *vs.* Fly" and "Bird *vs.* Chicken" Classification Tasks (the Results are Reported in Percentage)

Classification Tasks	k-NN	SVM	DT	RF	NN-1	NN-2	GeLSTM
Beetle *vs.* Fly	55	85	70	70	85	90	95
Bird *vs.* Chicken	50	60	80	60	65	80	85

6 CONCLUSION

In this article, we proposed to generalize LSTM to generic machine-learning tasks where data are represented in instance-feature tabular format. Our goal is to convert each instance into an LSTM compatible three-dimensional tensor representation, such that LSTM can be directly applied to the converted data to learn new features to improve classification accuracy, compared to the classifier learned from the original instances. To this end, we proposed to reorder features of each instance as a synthetic sentence format with sequential correlations. The horizontal and vertical feature aliment creates maximized feature correlations, through which LSTM can learn new feature values. Experiments and comparisons on 20 generic datasets confirm that generalizing LSTM to generic datasets can improve the classification accuracy of conventional machine learning methods, including random forests, XGBoost, K-NN, and the like.

APPENDICES

A THE PROOF OF EQUATION (5)(a)

Proof. Since $\boldsymbol{w_i} = [w_{i,1}, \ldots, w_{i,k}, \ldots, w_{i,m}]$ and $\boldsymbol{w_{i-1}} = [w_{i-1,1}, \ldots, w_{i-1,k}, \ldots, w_{i-1,m}]$ are not genuine words, we cannot derive $p(\boldsymbol{w_i}|\boldsymbol{w_{i-1}})$ using text corpus. Alternatively, $\boldsymbol{w_{i-1}}$ can be regarded as the received codewords considering the discrete memoryless channel (DMC) in the wireless communication system, and $\boldsymbol{w_i} \in \boldsymbol{C}$ can be regarded as the likely transmitted codewords where $\boldsymbol{C}$ stands for the available codewords set. In addition, we use $\boldsymbol{w_t} = [w_{t,1}, \ldots, w_{t,k}, \ldots, w_{t,m}] \in \boldsymbol{C}$ to denote the transmitted codewords. $w_{i,k}$, $w_{i-1,k}$ and $w_{t,k}$ are the k^{th} codeword in codewords $\boldsymbol{w_i}$, $\boldsymbol{w_{i-1}}$ and $\boldsymbol{w_t}$ accordingly. The relationship between these three kinds of codewords is shown in Figure 15. Specifically, the transmitter, such as the smartphone, generates the transmitted codewords $\boldsymbol{w_t} \in \boldsymbol{C}$. Then, $\boldsymbol{w_t}$ passes through the DMC, and DMC outputs the received codewords $\boldsymbol{w_{i-1}}$. For such system, by utilizing the likely transmitted codeword $w_{i,k}$ and the received codeword $w_{i-1,k}$, the maximum a posterior probability (MAP) decoder estimates the k^{th} transmitted codeword (the estimated result is denoted by $w'_{t,k}$). We can utilize MAP channel decoding rule to justify Equation (5)(a).

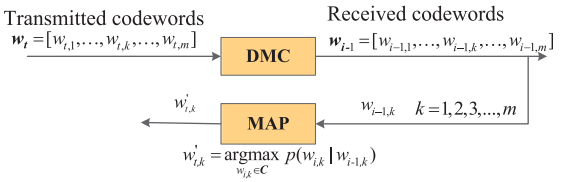

Fig. 15. A conceptual view of communication system considering the discrete memoryless channel (DMC) and the maximum a posterior probability (MAP) decoder: The transmitted codewords $\boldsymbol{w_t} \in \boldsymbol{C}$ passes through the DMC, and DMC outputs the received codewords $\boldsymbol{w_{i-1}}$. By utilizing the likely transmitted codeword $w_{i,k}$ and the received codeword $w_{i-1,k}$, the MAP decoder estimates the k^{th} transmitted codeword (the estimated result is denoted by $w'_{t,k}$).

By utilizing the feature of the DMC, Equation (5) can be rewritten as [3]

$$\begin{aligned}
&\underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad p(\boldsymbol{w_i}|\boldsymbol{w_{i-1}}) \\
&= \underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad p(w_{i,1},w_{i,2},\ldots,w_{i,m}|w_{i-1,1},w_{i-1,2},\ldots,w_{i-1,m}) \\
&= \underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad \prod_{k=1}^{m} p(w_{i,k}|w_{i-1,k}).
\end{aligned} \tag{12}$$

Given the received codeword $w_{i-1,k}$, the MAP decoding rule regards the most likely transmitted codeword $w_{i,k}$, as the estimated codeword $w'_{t,k}$, as shown in Figure 15 [3].

$$w'_{t,k} = \underset{w_{i,k}\in C}{\arg\max}\, p(w_{i,k}|w_{i-1,k}) \tag{13}$$

Then, based on the Bayesian formula $p(w_{i-1,k}|w_{i,k}) = \frac{p(w_{i,k}|w_{i-1,k})p(w_{i-1,k})}{p(w_{i,k})}$, if the codewords are transmitted with the same probability, the MAP $p(w_{i,k}|w_{i-1,k})$ is equivalent to the maximum likelihood (MLD) $p(w_{i-1,k}|w_{i,k})$, *i.e.,*

$$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \max p(w_{i-1,k}|w_{i,k}) \tag{14}$$

where the symbol "⇒" means equivalent. Our feature reordering mechanism satisfies this condition in the sense that the feature reordering mechanism ensures the fairness between features. Next, considering the white noise σ^2, the MLD can be written as [3]

$$p(w_{i-1,k}|w_{i,k}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d(w_{i-1,k},w_{i,k})}{\sigma^2}\right) \tag{15}$$

where $d(w_{i,k},w_{i-1,k})$ stands for the Euclidean distance between $w_{i,k}$ and $w_{i-1,k}$.

We can see from Equation (15) that $\max p(w_{i-1,k}|w_{i,k})$ is equivalent to finding the minimum distance between $w_{i,k}$ and $w_{i-1,k}$ (*i.e.,* $\max p(w_{i-1,k}|w_{i,k}) \Rightarrow \min d(w_{i-1,k},w_{i,k})$). Due to the fact that "$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \max p(w_{i-1,k}|w_{i,k})$", we have "$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \min d(w_{i-1,k},w_{i,k})$". Furthermore, the smaller the distance between $w_{i,k}$ and $w_{i-1,k}$, the stronger the correlation between $w_{i,k}$ and $w_{i-1,k}$ (*i.e.,* $\min d(w_{i-1,k},w_{i,k}) \Rightarrow \max \mathcal{M}_{w_{i-1,k},w_{i,k}}$). Hence, we have "$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \max \mathcal{M}_{w_{i-1,k},w_{i,k}}$". Then, Equation (12) can be rewritten as

$$\begin{aligned}
&\underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad p(\boldsymbol{w_i}|\boldsymbol{w_{i-1}}) \\
&= \underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad p(w_{i,1},w_{i,2},\ldots,w_{i,m}|w_{i-1,1},w_{i-1,2},\ldots,w_{i-1,m}) \\
&= \underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad \prod_{k=1}^{m} p(w_{i,k}|w_{i-1,k}) \\
&\approx \underset{\boldsymbol{w_i}=[w_{i,1},w_{i,2},\ldots,w_{i,m}]}{\arg\max} \quad \prod_{k=1}^{m} \mathcal{M}_{w_{i-1,k},w_{i,k}},
\end{aligned} \tag{16}$$

Hence, Equation (5)(a) holds. □

B AN EXAMPLE OF EQUATION (5)(a)

Example. We use the channel decoding as an example to justify assumption that "$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \max \mathcal{M}_{w_{i-1,k},w_{i,k}}$". The details are described as follows.

We assume that the available transmitted codewords set is $\boldsymbol{C} = [C_1, C_2, C_3]$ where $C_1 = [100000111000]$, $C_2 = [101100101000]$, and $C_3 = [100000100000]$. For the k^{th} codeword, we

Table 7. The Euclidean Distance between $w_{i-1,k}$ and $w_{i,k}$ and their Correlation Coefficient for the Toy Example

Likely transmitted codeword	$d(w_{i-1,k}, w_{i,k})$	$\mathcal{M}_{w_{i-1,k}, w_{i,k}}$
$w_{i,k} = C_1$	1.414	0.6325
$w_{i,k} = C_2$	1.732	0.5292
$w_{i,k} = C_3$	2	0.4472

assume that the received codeword is $w_{i-1,k} = [100000100000]$, and the transmitted codeword is $w_{t,k} = C_3 = [100000100000]$.

Given the received codeword $w_{i-1,k}$, the MAP decoder estimates the transmitted codeword, according to $w'_{t,k} = \arg\max_{w_{i,k} \in C} P(w_{i,k}|w_{i-1,k})$.

Note that $\max p(w_{i,k}|w_{i-1,k})$ is equivalent to finding the minimum Euclidean distance between $w_{i,k}$ and $w_{i-1,k}$. The Euclidean distance between $w_{i-1,k}$ and $w_{i,k}$ is reported in Table 7. In order to observe the relationship between $p(w_{i,k}|w_{i-1,k})$ and $\mathcal{M}_{w_{i-1,k}, w_{i,k}}$, we also calculate their correlation coefficient in Table 7.

We can see from Table 7 that a lower $d(w_{i-1,k}, w_{i,k})$ value results in a relatively higher $\mathcal{M}_{w_{i-1,k}, w_{i,k}}$. Hence, based on "$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \min d(w_{i-1,k}, w_{i,k})$", we can derive that the higher the conditional probability value $p(w_{i,k}|w_{i-1,k})$, the higher the correlation between them is. In other words, we have "$\max p(w_{i,k}|w_{i-1,k}) \Rightarrow \max \mathcal{M}_{w_{i-1,k}, w_{i,k}}$". □

ACKNOWLEDGMENT

This research is supported by the US National Science Foundation (NSF) through Grant Nos. IIS-1763452 and CNS-1828181.

REFERENCES

[1] M. Abadi, A. Agarwal, and P. Barham. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. 1 (2015). Softwareavailablefromtensorflow.org.

[2] A. Adam Pauls and D. Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. 258–267.

[3] S. Al-Semari, F. Alajaji, and T. E. Fuja. 1999. Sequence MAP decoding of trellis codes for Gaussian and Rayleigh channels. *IEEE Transactions on Vehicular Technology* 48, 4 (1999), 1130–1140.

[4] K. G. Anil. 2006. On optimum choice of k in nearest neighbour classification. *Computational Statistics and Data Analysis* 50, 11 (2006), 3113–3123.

[5] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[6] Y. Bengio, O. Delalleau, and N. Le Roux. 2005. The curse of highly variable functions for local kernel machines. In *Proceedings of the Advances in Neural Information Processing Systems, British Columbia, Canada*. MIT Press, 107–114.

[7] Y. Bengio and P. Simard. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166.

[8] Mairead L. Bermingham, Ricardo Pong-Wong, Athina Spiliopoulou, et al. 2015. Application of high-dimensional feature selection: Evaluation for genomic prediction in man. *Scientific Reports* 5, 10312 (2015).

[9] A. L. Blum and P. Langley. 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence* 97, 1–2 (1997), 245–271.

[10] C. E. Brodley and P. E. Utgoff. 1995. Multivariate decision trees. *Machine Learning* 19, 1 (1995), 45–77.

[11] Xiaojun Chang, Feiping Nie, Yi Yang, Chengqi Zhang, and Heng Huang. 2016. Convex sparse PCA for unsupervised feature learning. *ACM Transactions on Knowledge Discovery from Data* 11, 1 (2016), 3:1–3:16.

[12] L. Changki and L. G. Geunbae. 2006. Information gain and divergence-based feature selection for machine learning-based text categorization. *Information Processing & Management* 42, 1 (2006), 155–165.

[13] T. Chen and C. Guestrin. 2016. XGBoost: A scalable tree boosting System. In *Proceedings of the Conference on Knowledge Discovery and Data Mining*.

[14] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The UCR Time Series Classification Archive. Retrieved from www.cs.ucr.edu/~eamonn/time_series_data/.
[15] Dan Ciresan, U. Meier, and J. Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*. 3642–3649.
[16] C. M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK.
[17] C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.
[18] R. A. Dunne and N. A. Campbel. 1997. On the pairing Of the softmax activation and cross entropy penalty functions and the derivation of the softmax activation function. In *Proceedings of the 8th Australian Conference on Neural Networks*. 181–185.
[19] M. Federico and M. Cettolo. 2007. Efficient handling of n-gram language models for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*. 88–95.
[20] F. Gers, N. Schraudolph, and J. Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research* 3, 1 (2002), 115–143.
[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press, Cambridge, MA.
[22] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. 6645–6649.
[23] A. Graves, A. R. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, Canada*. 6645–6649.
[24] A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18, 5 (2005), 602–610.
[25] I. Guyon and A. Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3, 6 (2003), 1157–1182.
[26] M. F. A. Hady and F. Schwenker. 2013. *Semi-supervised Learning,in Handbook on Neural Information Processing*. Springer, Berlin, Germany.
[27] H. Han, Y. Li, and X. Zhu. 2019. Convolutional neural network learning for generic data classification. *Information Sciences* 477 (2019), 448–465.
[28] H. Han, X. Zhu, and Y. Li. 2018. EDLT: Enabling deep learning for generic data classification. In *Proceedings of the IEEE International Conference on Data Mining*.
[29] J. Hauke and T. Kossowski. 2011. Comparison of values of Pearson's and Spearman's correlation coefficient on the same sets of data. *Quaestiones Geographicae* 31, 2 (2011), 87–93.
[30] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 3 (1997), 1735–1780.
[31] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4, 2 (1991), 251–257.
[32] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Springer.
[33] Adebayo Kolawole John, Luigi Di Caro, and Guido Boella. 2016. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 12th International Conference on Semantic Systems*.
[34] D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.
[35] R. Kohavi and G. H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligent* 97, 12 (1997), 273–324.
[36] Alex Krizhevsky, Ilya Sutskever, and Geoffry Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems, Lake Tahoe, Nevada*.
[37] L. Ladla and T. Deepa. 2011. Feature selection methods and algorithms. *International Journal on Computer Science and Engineering* 3, 5 (2011), 1787–1797.
[38] P. Langley. 1994. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance, New Orleans, Louisiana*. 140–144.
[39] Y. LeCun, G. Bengio, and Y. Hinton. 2015. Deep learning. *Nature* 521 (2015), 436–444.
[40] Y. LeCun, G. Bengio, and Y. Hinton. 2019. Fast video frame correlation analysis for vehicular networks by using CVS-CNN. *IEEE Transactions on Vehicular Technology* 68, 7 (2019), 6286–6296.
[41] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Proceedings of the Advances in Neural Information Processing Systems, Vancouver, Canada*. MIT Press, 396–404.
[42] Huan Liu and Hiroshi Motoda. 1998. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publishers.
[43] H. Liu and R. Setiono. 1995. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*.

[44] D. Lunga, S. Prasad, M. M. Crawford, and O. Ersoy. 2014. Manifold learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning. *IEEE Signal Processing Magazine* 31, 1 (2014), 55–66.
[45] Nasser M. Nasrabadi. 2007. Pattern recognition and machine learning. *Journal of Electronic Imaging* 16, 4 (2007), 049901.
[46] D. Newman, S. Hettich, C. Blake, and C. Merz. 1998. UCI repository of machine learning databases, Irvine. University of California, Department of Information and Computer Science, CA. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html.
[47] F. Pedregosa, G. Varoquaux, A. Gramfort, and V. Michel. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 10 (2011), 2825–2830.
[48] V. Rokhlin, A. Szlam, and M. Tygert. 2009. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications* 31, 3 (2009), 1100–1124.
[49] H. Sak et al. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of the Annual Conference of the International Speech Communication Association*. 338–342.
[50] J. Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61, 1 (2015), 85–117.
[51] B. Scholkopft and K.-R. Mullert. 1999. *Neural Networks for Signal Processing*. Springer.
[52] L. J. P. van der Maaten and G. E. Hinton. 2008. Visualizing High-dimensional data using t-SNE. *Journal of Machine Learning Research* 9, 12 (2008), 2579–2605.
[53] Y. Wang, M. Huang, L. Zhao, and X. Zhu. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the Conference on Conference on Empirical Methods in Natural Language Processing*.
[54] Man Wu, Shirui Pan, Xingquan Zhu, Chuan Zhou, and Lei Pan. 2019. Domain-adversarial graph neural networks for text classification. In *Proceedings of the IEEE International Conference on Data Mining*.
[55] Y. Wu, S. Hio, T. Mei, and N. Yu. 2017. Large-scale online feature selection for ultra-high dimensional sparse data. *ACM Transactions on Knowledge Discovery from Data* 11, 4 (2017), 48:1–48:22.
[56] Kui Yu, Xindong Wu, Wei Ding, and Jian Pei. 2016. Scalable and accurate online feature selection for big data. *ACM Transactions on Knowledge Discovery from Data* 11, 2 (2016), 16:1–16:39.
[57] D. Zhang, J. Wang, F. Wang, and C. Zhang. 2008. Semi-supervised classification with universum. In *Proceedings of the SIAM International Conference on Data Mining, San Diego, CA*. 323–333.
[58] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE Transactions on Big Data* (2018). DOI: https://doi.org/10.1109/TBDATA.2018.2850013
[59] X. Zhu. 2011. Cross-domain semi-supervised learning using feature formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 41, 6 (2011), 1627–1638.

Received December 2018; revised August 2019; accepted October 2019