Neural Networks 116 (2019) 237–245

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Multivariate LSTM-FCNs for time series classification

Fazle Karim [a], Somshubra Majumdar [b], Houshang Darabi [a,*], Samuel Harford [a]

[a] Mechanical and Industrial Engineering, University of Illinois at Chicago, 900 W. Taylor St., Chicago, IL, 60607, USA
[b] Computer Science, University of Illinois at Chicago, 900 W. Taylor St., Chicago, IL, 60607, USA

ARTICLE INFO

Article history:
Received 21 August 2018
Received in revised form 16 April 2019
Accepted 17 April 2019
Available online 4 May 2019

Keywords:
Convolutional neural network
Long short term memory
Recurrent neural network
Multivariate time series classification

ABSTRACT

Over the past decade, multivariate time series classification has received great attention. We propose transforming the existing univariate time series classification models, the Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN), into a multivariate time series classification model by augmenting the fully convolutional block with a *squeeze-and-excitation* block to further improve accuracy. Our proposed models outperform most state-of-the-art models while requiring minimum preprocessing. The proposed models work efficiently on various complex multivariate time series classification tasks such as activity recognition or action recognition. Furthermore, the proposed models are highly efficient at test time and small enough to deploy on memory constrained systems.

1. Introduction

Time series data is used in various fields of studies, ranging from weather readings to psychological signals (Cui, Shi, & Wang, 2015; Kadous, 2002; Kehagias & Petridis, 1997; Sharabiani et al., 2017). A time series is a sequence of data points in a time domain, typically in a uniform interval (Wang, Wang, & Liu, 2016). There is a significant increase of time series data being collected by sensors (Spiegel, Gaebler, Lommatzsch, De Luca, & Albayrak, 2011). A time series dataset can be univariate, where a sequence of measurements from the same variable are collected, or multivariate, where a sequence of measurements from multiple variables or sensors are collected (Prieto, Alonso-González, & Rodríguez, 2015). Over the past decade, multivariate time series classification has received significant interest. Multivariate time series classifications are applied in healthcare (Kang & Choi, 2014), phoneme classification (Graves & Schmidhuber, 2005), activity recognition, object recognition, and action recognition (Fu, 2015; Geurts, 2001; Pavlovic, Frey, & Huang, 1999; Yu & Lee, 2015). In this paper, we propose two deep learning models that outperform existing algorithms.

Several time series classification algorithms have been developed over the years. Distance-based methods along with k-nearest neighbors have proven to be successful in classifying multivariate time series (Orsenigo & Vercellis, 2010). Plenty of research indicates Dynamic Time Warping (DTW) as the best distance-based measure to use along k-NN (Seto, Zhang, & Zhou, 2015).

In addition to distance-based metrics, other algorithms are used. Typically, feature-based classification algorithms rely heavily on the features being extracted from the time series data (Xing, Pei, & Keogh, 2010). However, feature extraction is arduous because intrinsic features of time series data are challenging to capture. For this reason, distance-based approaches are more successful in classifying multivariate time series data (Zheng, Liu, Chen, Ge, & Zhao, 2014). Hidden State Conditional Random Field (HCRF) and Hidden Unit Logistic Model (HULM) are two successful feature-based algorithms which have led to state-of-the-art results on various benchmark datasets, ranging from online character recognition to activity recognition (Pei, Dibeklioğlu, Tax, & van der Maaten, 2017). HCRF is a computationally expensive algorithm that detects latent structures of the input time series data using a chain of k-nominal latent variables. The number of parameters in the model increases linearly with the total number of latent states required (Quattoni, Wang, Morency, Collins, & Darrell, 2007). Further, datasets that require a large number of latent states tend to overfit the data. To overcome this, HULM proposes using H binary stochastic hidden units to model 2^H latent structures of the data with only O(H) parameters. Results indicate HULM outperforming HCRF on most datasets (Pei et al., 2017).

Traditional models, such as the naive logistic model (NL) and Fisher kernel learning (FKL) (Jaakkola, Diekhans, & Haussler, 2000), show strong performance on a wide variety of time series classification problems. The NL logistic model is a linear logistic model that makes a prediction by summing the inner products between the model weights and feature vectors over time, which is followed by a softmax function (Pei et al., 2017). The FKL model is effective on time series classification problems when based

* Corresponding author.
E-mail addresses: karim1@uic.edu (F. Karim), smajum6@uic.edu (S. Majumdar), hdarabi@uic.edu (H. Darabi), sharfo2@uic.edu (S. Harford).

https://doi.org/10.1016/j.neunet.2019.04.014
0893-6080/

on Hidden Markov Models (HMM). Subsequently, the features or representation from the FKL model is used to train a linear SVM to make a final prediction (Jaakkola et al., 2000; Maaten, 2011).

Another common approach for multivariate time series classification is by applying dimensional reduction techniques or by concatenating all dimensions of a multivariate time series into a univariate time series. Symbolic Representation for Multivariate Time Series (SMTS) (Baydogan & Runger, 2015) applies a random forest on the multivariate time series to partition it into leaf nodes, each represented by a word to form a codebook. Every word is used with another random forest to classify the multivariate time series. Learned Pattern Similarity (LPS) (Baydogan & Runger, 2016) is a similar model that extracts segments from the multivariate time series. These segments are used to train regression trees to find dependencies between them. Each node is represented by a word. Finally, these words are used with a similarity measure to classify the unknown multivariate time series. Ultra Fast Shapelets (UFS) (Wistuba, Grabocka, & Schmidt-Thieme, 2015) obtains random shapelets from the multivariate time series and applies a linear SVM or a Random Forest classifier. Subsequently, UFS was enhanced by computing derivatives as features (dUFS) (Wistuba et al., 2015). The Auto-Regressive (AR) kernel (Cuturi & Doucet, 2011) applies an AR kernel-based distance measure to classify the multivariate time series. Auto-Regressive forests for multivariate time series modeling (mv-ARF) (Tuncel & Baydogan, 2018) uses a tree ensemble, where the trees are trained with different time lags. Most recently, WEASEL+MUSE (Schäfer & Leser, 2017) builds a multivariate feature vector using a classical bag of patterns approach on each variable with various sliding window sizes to capture discrete features, words, and pairs of words. Subsequently, feature selection is used to remove non-discriminative features using a Chi-squared test. The final classification is obtained using a logistic classifier on the final feature vector.

Deep learning has also yielded promising results for multivariate time series classification. In 2014, *Yi et al.* propose using Multi-Channel Deep Convolutional Neural Network (MC-DCNN) for multivariate time series classification. MC-DCNN takes input from each variable to detect latent features. The latent features from each channel are fed into an MLP to perform classification (Zheng et al., 2014).

This paper proposes two deep learning models for multivariate time series classification. These proposed models require minimal preprocessing and are tested on 35 datasets, obtaining strong performances in most of them. Performance is the classification accuracy of a model on a particular dataset. The rest of the paper is ordered as follows. Background works are discussed in Section 2. We present the architecture of the two proposed models in Section 3. In Section 4, we discuss the dataset, evaluate the models on them, present our results and analyze our findings. In Section 5, we draw our conclusion.

2. Background works

2.1. Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a form of neural networks that display temporal behavior through the direct connections between individual layers. Pascanu, Gulcehre, Cho, and Bengio (2013) implement RNN to maintain a hidden vector $\mathbf{h}$ that is updated at time step t,

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{I}\mathbf{x}_t), \tag{1}$$

where the hyperbolic tangent function is represented by *tanh*, the input vector at time step t is denoted as $\mathbf{x}_t$, the recurrent weight matrix is denoted by $\mathbf{W}$ and the projection matrix is signified by $\mathbf{I}$. A prediction, $\mathbf{y}_t$ can be made using a hidden state, $\mathbf{h}$, and a weight matrix, $\mathbf{W}$,

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}\mathbf{h}_{t-1}). \tag{2}$$

The softmax function normalizes the output predictions of the model to be a valid probability distribution and the logistic sigmoid function is declared as σ. RNNs can be stacked to create deeper networks by using the hidden state, $\mathbf{h}^{l-1}$ of a RNN layer $l-1$ as an input to the hidden state, $\mathbf{h}^l$ of another RNN layer l,

$$\mathbf{h}_t^l = \sigma(\mathbf{W}\mathbf{h}_{t-1}^l + \mathbf{I}\mathbf{h}_t^{l-1}). \tag{3}$$

2.2. Long short-term memory RNNs

A major issue with RNNs is that they often have to face the issue of vanishing gradients. Long short-term memory (LSTM) RNNs address this problem by integrating gating functions into their state dynamics (Hochreiter & Schmidhuber, 1997). An LSTM maintains a hidden vector, $\mathbf{h}$, and a memory vector, $\mathbf{m}$, which control state updates and outputs at each time step, respectively. The computation at each time step is depicted by Graves et al. (2012) as the following:

$$\begin{aligned}
\mathbf{g}^u &= \sigma(\mathbf{W}^u\mathbf{h}_{t-1} + \mathbf{I}^u\mathbf{x}_t)\\
\mathbf{g}^f &= \sigma(\mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{I}^f\mathbf{x}_t)\\
\mathbf{g}^o &= \sigma(\mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{I}^o\mathbf{x}_t)\\
\mathbf{g}^c &= \tanh(\mathbf{W}^c\mathbf{h}_{t-1} + \mathbf{I}^c\mathbf{x}_t)\\
\mathbf{m}_t &= \mathbf{g}^f \odot \mathbf{m}_{t-1} + \mathbf{g}^u \odot \mathbf{g}^c\\
\mathbf{h}_t &= \tanh(\mathbf{g}^o \odot \mathbf{m}_t)
\end{aligned} \tag{4}$$

where $\mathbf{g}^u$, $\mathbf{g}^f$, $\mathbf{g}^o$, $\mathbf{g}^c$ are the activation vectors of the input, forget, output and cell state gates respectively, h_t is the hidden state vector of the LSTM unit, the logistic sigmoid function is defined by σ, the elementwise multiplication is represented by $\odot$. The recurrent weight matrices are depicted using the notation $\mathbf{W}^u, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^c$ and the projection matrices are portrayed by $\mathbf{I}^u, \mathbf{I}^f, \mathbf{I}^o, \mathbf{I}^c$.

LSTMs can learn temporal dependencies. However, long-term dependencies of long sequence are challenging to learn using LSTMs. Bahdanau, Cho, and Bengio (2014) proposed using an attention mechanism to learn these long-term dependencies.

2.3. Attention mechanism

An attention mechanism conditions a context vector V on the target sequence y. This method is commonly used in neural translation of texts. Bahdanau et al. (2014) argue the context vector v_i depends on a sequence of *annotations* $(b_1, \ldots, b_{T_x})$, of length T_x which is the maximum length of the input sequence x, where an encoder maps the input sequence. Each annotation, b_i, comprises information on the whole input sequence, while focusing on regions surrounding the ith word of the input sequence. The weighted sum of each annotation, b_i, is used to compute the context vector as follows:

$$v_i = \sum_{j=1}^{T_x} \alpha_{ij} b_j. \tag{5}$$

The weight, α_{ij}, of each annotation is calculated through :

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}, \tag{6}$$

where the energy of alignment, e_{ij}, is given by $a(v_{i-1}, b_j)$, which measures how well the input position, j, and the output at position, i, match using the RNN hidden state, v_{i-1}, and the jth annotation, b_j, of the input sequence. Bahdanau et al. (2014)

use a feedforward neural network to parameterize the alignment model, *a*. The feedforward neural network is trained jointly with all other components of the model. In addition, the alignment model calculates a soft alignment that can backpropagate the gradient of the cost function. The gradient of the cost function trains the alignment model and the translation model simultaneously (Bahdanau et al., 2014).

2.4. Squeeze-and-excitation block

Hu, Shen, and Sun (2017) propose a *squeeze-and-excitation* block that acts as a computational unit for any transformation $\mathbf{F}_{tr} : \mathbf{X} \rightarrow \mathbf{U}, \mathbf{X} \in \mathbb{R}^{W' \times H' \times C'}, \mathbf{U} \in \mathbb{R}^{W \times H \times C}$. The outputs of $\mathbf{F}_{tr}$ are represented as $\mathrm{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_C]$ where

$$\mathbf{u}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s \tag{7}$$

The convolution operation is depicted by $*$, and the 2D spatial kernel is depicted by $\mathbf{v}_c^s$. The single channel of v_c acts on the corresponding channel of Hu et al. (2017) models the channel interdependencies to adjust the filter responses in two steps, *squeeze* and *excitation*.

The *squeeze* operation exploits the contextual information outside the local receptive field by using a global average pool to generate channel-wise statistics. The transformation output, **U**, is shrunk through spatial dimensions, $W \times H$, to compute the channel-wise statistics, $\mathrm{z} \in \mathbb{R}^C$. The c-th element of $\mathbf{z}$ is calculated by computing $\mathbf{F}_{sq}(\mathbf{u}_c)$, which is the channel-wise global average over the spatial dimensions $W \times H$, defined as:

$$\mathbf{z}_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} u_c(i, j) \tag{8}$$

For temporal sequence data, the transformation output, **U**, is shrunk through the temporal dimension T to compute the channel-wise statistics, $\mathrm{z} \in \mathbb{R}^C$. The c-th element of $\mathbf{z}$ is then calculated by computing $\mathbf{F}_{sq}(\mathbf{u}_c)$, which is the channel-wise global average over the temporal dimension T, defined as:

$$\mathbf{z}_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{T} \sum_{t=1}^{T} u_c(t) \tag{9}$$

The aggregated information obtained from the *squeeze* operation is followed by an *excite* operation, whose objective is to capture the channel-wise dependencies. To achieve this, a simple gating mechanism is applied with a sigmoid activation, as follows:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 z)), \tag{10}$$

where $\mathbf{F}_{ex}$ is parameterized as a neural network, σ is the Sigmoid activation function, δ is the ReLU activation function, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{\frac{C}{r} \times C}$ are learnable parameters of $\mathbf{F}_{ex}$ and r is the reduction ratio. $\mathbf{W}_1$ and $\mathbf{W}_2$ are used to limit model complexity and aid with generalization. $\mathbf{W}_1$ are the parameters of a dimensionality-reduction layer and $\mathbf{W}_2$ are the parameters of a dimensionality-increasing layer.

Finally, the output of the block is rescaled as follows:

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, \mathbf{s}_c) = \mathbf{s}_c \cdot \mathbf{u}_c, \tag{11}$$

where $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \ldots, \widetilde{\mathbf{x}}_C]$ and $\mathbf{F}_{scale}(\mathbf{u}_c, \mathbf{s}_c)$ denotes the channel-wise multiplication between the feature map $\mathbf{u}_c \in \mathbb{R}^T$ and the scale s_c.

3. Multivariate LSTM fully convolutional network

3.1. Network architecture

Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN) have been successful in classifying univariate time series (Karim, Majumdar, Darabi, & Chen, 2017). However, they have never been applied to on a multivariate time series classification problem. The models we propose, Multivariate LSTM-FCN (MLSTM-FCN) and Multivariate Attention LSTM-FCN (MALSTM-FCN), convert their respective univariate models into multivariate variants. We extend the *squeeze-and-excite* block to the case of 1D sequence models and augment the fully convolutional blocks of the LSTM-FCN and ALSTM-FCN models to enhance classification accuracy.

As the datasets now consist of multivariate time series, we can define a time series dataset as a tensor of shape (N, Q, M), where N is the number of samples in the dataset, Q is the maximum number of time steps amongst all variables and M is the number of variables processed per time step. Therefore a univariate time series dataset is a special case of the above definition, where M is 1. The alteration required to the input of the LSTM-FCN and ALSTM-FCN models is to accept M inputs per time step, rather than a single input per time step.

Similar to LSTM-FCN and ALSTM-FCN, the proposed models comprise a fully convolutional block and a LSTM block, as depicted in Fig. 1. The fully convolutional block contains three temporal convolutional blocks, used as a feature extractor, which is replicated from the original fully convolutional block by Wang, Yan, and Oates (2017). The convolutional blocks contain a convolutional layer with a number of filters (128, 256, and 128) and a kernel size of 8, 5, and 3 respectively. Each convolutional layer is succeeded by batch normalization, with a momentum of 0.99 and epsilon of 0.001. The batch normalization layer is succeeded by the ReLU activation function. In addition, the first two convolutional blocks conclude with a *squeeze-and-excite* block, which sets the proposed model apart from LSTM-FCN and ALSTM-FCN. Fig. 2 summarizes the process of how the *squeeze-and-excite* block is computed in our architecture. For all *squeeze and excitation* blocks, we set the reduction ratio r to 16. The final temporal convolutional block is followed by a global average pooling layer.

The *squeeze-and-excite* block is an addition to the FCN block which adaptively recalibrates the input feature maps. Due to the reduction ratio r set to 16, the number of parameters required to learn these self-attention maps is reduced such that the overall model size increases by just 3%–10%. This can be computed as below:

$$P = \frac{2}{r} \sum_{s=1}^{S} R_s \cdot G_s^2$$

where P is the total number of additional parameters, r denotes the reduction ratio, S denotes the number of stages (each stage refers to the collection of blocks operating on feature maps of a common spatial dimension), G_s denotes the number of output feature maps for stage s and R_s denotes the repeated block number for stage s. Since the FCN blocks are kept consistent for all models, we can directly compute the additional number of parameters as $P = \frac{2}{16} * (128^2 + 256^2) = 10{,}240$ for all models. *Squeeze* and *excitation* is essential to the enhanced performance on multivariate datasets, as not all feature maps may impact the subsequent layers to the same degree. This adaptive recalibration of the feature maps can be considered as a form of learned self-attention on the output feature maps of prior layers. This adaptive rescaling of the filter maps is of utmost importance to the improved performance of the MLSTM-FCN model compared

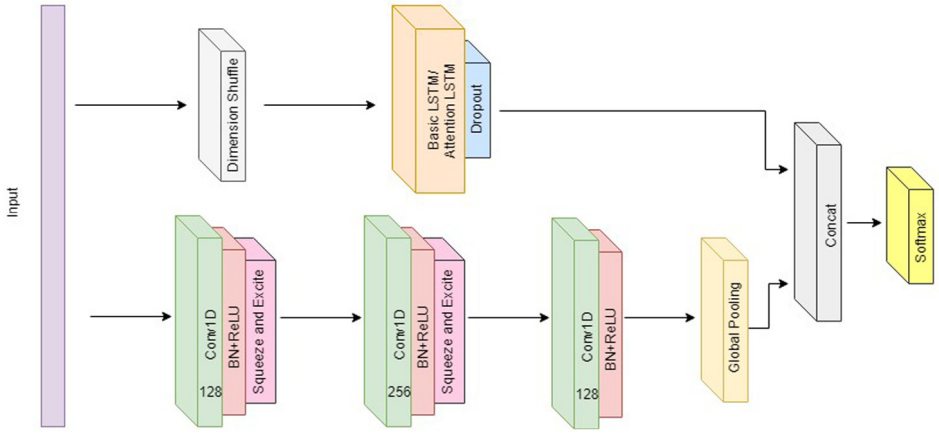

Fig. 1. The MLSTM-FCN architecture. LSTM cells can be replaced by Attention LSTM cells to construct the MALSTM-FCN architecture.

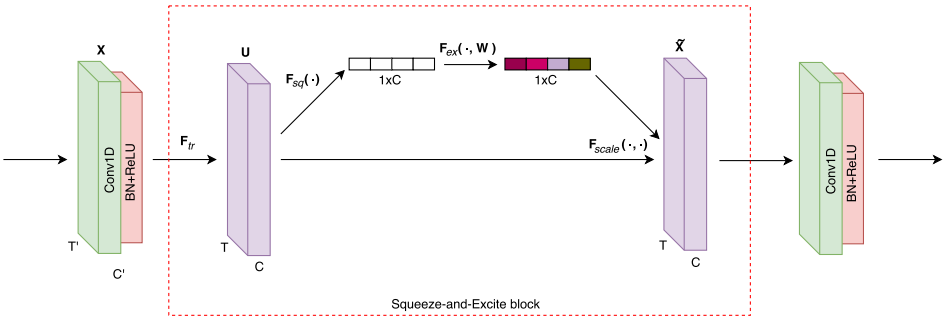

Fig. 2. The computation of the temporal *squeeze-and-excite* block.

to LSTM-FCN, as it incorporates learned self-attention to the inter-correlations between multiple variables at each time step, which was inadequate with the LSTM-FCN.

In addition, the multivariate time series input is passed through a dimension shuffle layer (explained more in Section 3.2), followed by the LSTM block. The LSTM block is identical to the block from the LSTM-FCN or ALSTM-FCN models (Karim et al., 2017), comprising either an LSTM layer or an Attention LSTM layer, which is followed by a dropout layer.

3.2. Network input

Depending on the dataset, the input to the fully convolutional block and LSTM block vary. The input to the fully convolutional block is a multivariate variate time series with Q time steps having M distinct variables per time step. If there is a time series with M variables and Q time steps, the fully convolutional block will receive the data as such. Variables are defined as the channels of interconnected data streams.

In addition, the input to the LSTM can vary depending on the application of dimension shuffle. The dimension shuffle transposes the temporal dimension of the input data. If the dimension shuffle operation is not applied to the LSTM path, the LSTM will require Q time steps to process M variables at each time step. However, if the dimension shuffle is applied, the LSTM will require M time steps to process Q variables per time step. In other words, the dimension shuffle improves the efficiency of the model when the number of variables M is less than the number of time steps Q.

After the dimension shuffle, at each time step t, where $1 \leq t \leq M$, M being the number of variables, the input provides the LSTM the entire history of that variable (data of that variable over all Q time steps). Thus, the LSTM is given the global temporal information of each variable at once. As a result, the dimension shuffle operation reduces the computation time of training and inference without losing accuracy for time series classification problems. An ablation test is performed to show the performance of a model with the dimension shuffle operation is statistically the same as a model without using it (further discussed in Section 4.4).

The proposed models take a total of 13 h to process the MLSTM-FCN and a total of 18 h to process the MALSTM-FCN on a single GTX 1080 Ti GPU. While the time required to train these models is significant, one can note their inference time is comparable with other standard models.

4. Experiments

MLSTM-FCN and MALSTM-FCN have been tested on 35 datasets, in Section 4.2. The optimal number of LSTM cells for each dataset was found via grid search over 3 distinct choices - 8, 64 or 128, and all other hyper parameters are kept constant. The FCN block is comprised of 3 blocks of 128-256-128 filters for all models, with kernel sizes of 8, 5, and 3 respectively, comparable with the original models proposed by Wang et al. (2017). Additionally, the first two FCN blocks are succeeded by the *squeeze-and-excitation* block. We consistently chose 16 as the reduction ratio r for all *squeeze-and-excitation* blocks, as suggested by Hu et al. (2017). During the training phase, we set the total number of training epochs to 250 unless explicitly stated and the dropout rate is set to 80% to mitigate overfitting. Each of the proposed models is trained using a batch size of 128. The convolution kernels are initialized by the *Uniform He* initialization scheme proposed by He, Zhang, Ren, and Sun (2015), which samples from the uniform distribution $U \in \left(-\sqrt{\frac{6}{d}}, \sqrt{\frac{6}{d}}\right)$, where d is the number of input units to the weight tensor. For datasets

Table 1
Properties of all datasets. The yellow cells are datasets used by Pei et al. (2017), the purple cells are datasets used by Schäfer and Leser (2017), and the blue cells are datasets from the UCI repository Lichman (2013). (See Carnegie Mellon University, 0000; Hammami & Bedda, 2010; Li, Zhang, & Liu, 2010; Lichman, 2013; Olszewski, 2012; Sübakan, Kurt, Cemgil, & Sankur, 2014; van der Maaten & Hendriks, 2012; Wang, Liu, Wu, & Yuan, 2012; Williams, Toussaint, & Storkey, 2008.)

Dataset	Num. of Classes	Num. of Variables	Max Training Length	Task	Train-Test Split	Source
OHC	20	30	173	Handwriting Classification	10-fold	Williams, Toussaint, and Storkey (2008)
Arabic Voice	88	39	91	Speaker Recognition	75-25 split	Hammami and Bedda (2010)
Cohn-Kanade AU-coded Expression (CK+)	7	136	71	Facial Expression Classification	10-fold	van der Maaten and Hendriks (2012)
MSR Action	20	570	100	Action Recognition	5 ppl in train; rest in test	Li, Zhang, and Liu (2010)
MSR Activity	16	570	337	Activity Recognition	5 ppl in train; rest in test	Wang, Liu, Wu, and Yuan (2012)
ArabicDigits	10	13	93	Digit Recognition	75-25 split	Lichman (2013)
AUSLAN	95	22	96	Sign Language Recognition	44-56 split	Lichman (2013)
CharacterTrajectories	20	3	205	Handwriting Classification	10-90 split	Lichman (2013)
CMU_MOCAP_S16	2	62	534	Action Recognition	50-50 split	Carnegie mellon university (0000)
DigitShape	4	2	97	Action Recognition	60-40 split	Sübakan, Kurt, Cemgil, and Sankur (2014)
ECG	2	2	147	ECG Classification	50-50 split	Olszewski (2012)
JapaneseVowels	9	12	26	Speech Recognition	42-58 split	Lichman (2013)
KickvsPunch	2	62	761	Action Recognition	62-38 split	Carnegie mellon university (0000)
LIBRAS	15	2	45	Sign Language Recognition	38-62 split	Lichman (2013)
LP1	4	6	15	Robot Failure Recogntion	43-57 split	Lichman (2013)
LP2	5	6	15	Robot Failure Recogntion	36-64 split	Lıchman (2013)
LP3	4	6	15	Robot Failure Recogntion	36-64 split	Lıchman (2013)
LP4	3	6	15	Robot Failure Recogntion	36-64 split	Lichman (2013)
LP5	5	6	15	Robot Failure Recogntion	39-61 split	Lichman (2013)
NetFlow	2	4	994	Action Recognition	60-40 split	Sübakan et al. (2014)
PenDigits	10	2	8	Digit Recognition	2-98 split	Lichman (2013)
Shapes	3	2	97	Action Recognition	60-40 split	Sübakan et al. (2014)
Uwave	8	3	315	Gesture Recognition	20-80 split	Lichman (2013)
Wafer	2	6	198	Manufacturing Classification	25-75 split	Olszewski (2012)
WalkVsRun	2	62	1918	Action Recognition	64-36 split	Carnegie mellon university (0000)
AREM	7	7	480	Activity Recognition	50-50 split	Lichman (2013)
HAR	6	9	128	Activity Recognition	71-29 split	Lichman (2013)
Daily Sport	19	45	125	Activity Recognition	50-50 split	Lichman (2013)
Gesture Phase	5	18	214	Gesture Recognition	50-50 split	Lichman (2013)
EEG	2	13	117	EEG Classification	50-50 split	Lichman (2013)
EEG2	2	64	256	EEG Classification	20-80 split	Lichman (2013)
HT Sensor	3	11	5396	Food Classification	50-50 split	Lichman (2013)
Movement AAL	2	4	119	Movement Classification	50-50 split	Lichman (2013)
Occupancy	2	5	3758	Occupancy Classification	35-65 split	Lichman (2013)
Ozone	2	72	291	Weather Classification	50-50 split	Lichman (2013)

with class imbalance, a class weighing scheme inspired by *King et al.* is utilized (King & Zeng, 2001), weighing the contribution of each class C_i $(1 \leq i \leq C)$ to the loss by the factor $Gw_i = \frac{N}{C \times N_{C_i}}$, where Gw_i is the loss scaling weight for the i-th class, N is the number of samples in the dataset, C is the number of classes and N_{C_i} is the number of samples which belong to class C_i.

We use the Adam optimizer (Kingma & Ba, 2014), with an initial learning rate set to 1e-3 and the final learning rate set to 1e-4 to train all models. In addition, after every 100 epochs, the learning rate is reduced by a factor of $1/\sqrt[3]{2}$. The datasets were normalized and preprocessed to have zero mean and unit variance. We append variable length time series with zeros afterwards to obtain a time series dataset with a constant length Q, where Q is the maximum length of the time series. Mean-standard deviation normalization and zero padding are the only preprocessing steps performed for all models. We compute the mean and standard deviation of the training dataset and apply these values to normalize both the train and test datasets. We use the Keras (Chollet et al., 2015) library with the Tensorflow backend (Abadi et al., 2015) to train the proposed models.

4.1. Evaluation metrics

In this paper, various models, including the proposed models, are evaluated using accuracy, arithmetic rank, geometric rank, the Wilcoxon signed-rank test, and mean per class error. The arithmetic and geometric rank are the arithmetic and geometric mean of the ranks,

$$\text{Arithmetic Mean}_K = \frac{\sum_K^{\mathbf{N}} rank_K}{\mathbf{N}}$$

$$\text{Geometric Mean}_K = \frac{\prod_K^{\mathbf{N}} rank_K}{\mathbf{N}},$$

where K is the dataset and $\mathbf{N}$ is the number of datasets.

The Wilcoxon signed-rank test is a non-parametric statistical test that hypothesizes the median of the rank between the compared models is the same. The alternative hypothesis of the Wilcoxon signed-rank test is that the median of the rank between the compared models is not the same. Finally, the mean per class error is the average error of each class for all the datasets,

$$PCE_k = \frac{1 - accuracy}{number\ of\ unique\ classes}$$

$$MPCE = \frac{1}{\mathbf{N}} \sum PCE_K.$$

4.2. Datasets

A total of 35 datasets are used to test the proposed models. Five of the 35 datasets are benchmark datasets used by

Table 2
Performance comparison of proposed models with the rest on benchmark datasets. Green cells denote model with best performance. Red font indicates models that have a strided convolution prior to the LSTM block. (See Cuturi & Doucet, 2011; Pei et al., 2017; Schäfer & Leser, 2017; Tuncel & Baydogan, 2018.)

Datasets	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	SOTA
Action 3d	71.72	75.42	72.73	74.74	70.71 [DTW]
Activity	53.13	61.88	55.63	58.75	66.25 [DTW]
ArabicDigits	100.00	100.00	99.00	99.00	99.00 (Schäfer & Leser, 2017)
Arabic-Voice	98.00	98.00	98.55	98.27	94.55 (Pei et al., 2017)
AREM	76.92	84.62	76.92	84.62	76.92 [DTW]
AUSLAN	97.00	97.00	96.00	96.00	98.00 (Schäfer & Leser, 2017)
CharacterTraject	99.00	100.00	99.00	100.00	99.00 (Schäfer & Leser, 2017)
CK+	96.07	96.43	97.10	97.50	93.56 (Pei et al., 2017)
CMUsubject16	100.00	100.00	100.00	100.00	100.00 (Tuncel & Baydogan, 2018)
Daily Sport	99.65	99.65	99.63	99.72	98.42 [DTW]
DigitShapes	100.00	100.00	100.00	100.00	100.00 (Tuncel & Baydogan, 2018)
ECG	85.00	86.00	86.00	86.00	93.00 (Schäfer & Leser, 2017)
EEG	60.94	65.63	64.06	64.07	62.50 [RF]
EEG2	90.67	91.00	90.67	91.33	77.50 [RF]
Gesture Phase	50.51	53.53	52.53	53.05	40.91 [DTW]
HAR	96.00	96.71	95.49	96.71	81.57 [RF]
HT Sensor	68.00	78.00	72.00	80.00	72.00 [DTW]
JapaneseVowels	99.00	100.00	99.00	99.00	98.00 (Cuturi & Douce, 2011)
KickvsPunch	90.00	100.00	90.00	100.00	100.00(Schäfer & Leser, 2017)
Libras	99.00	97.00	98.00	97.00	95.00 (Cuturi & Douce, 2011)
LP1	74.00	86.00	80.00	82.00	96.00 (Schäfer & Leser, 2017)
LP2	77.00	83.00	80.00	77.00	76.00 (Schäfer & Leser, 2017)
LP3	67.00	80.00	80.00	73.00	79.00 (Schäfer & Leser, 2017)
LP4	91.00	92.00	89.00	93.00	96.00 (Schäfer & Leser, 2017)
LP5	61.00	66.00	62.00	67.00	71.00 (Schäfer & Leser, 2017)
Movement AAL	73.25	79.63	70.06	78.34	65.61 [SVM-Poly]
NetFlow	94.00	95.00	93.00	95.00	98.00 (Schäfer & Leser, 2017)
Occupancy	71.05	76.31	71.05	72.37	67.11 [DTW]
OHC	99.96	99.96	99.96	99.96	99.03 (Pei et al., 2017)
Ozone	67.63	81.50	79.19	79.78	75.14 [DTW]
PenDIgits	97.00	97.00	97.00	97.00	95.00 (Cuturi & Douce, 2011)
Shapes	100.00	100.00	100.00	100.00	100.00 (Schäfer & Leser, 2017)
UWave	97.00	98.00	97.00	98.00	98.00 (Schäfer & Leser, 2017)
Wafer	99.00	99.00	99.00	99.00	99.00 (Schäfer & Leser, 2017)
WalkvsRun	100.00	100.00	100.00	100.00	100.00 (Schäfer & Leser, 2017)
Arith Mean	4.63	3.29	4.17	3.17	-
Geo Mean	3.72	2.58	3.21	2.42	-
Count	22.00	28.00	26.00	27.00	-
MPCE	5.01	4.21	4.93	4.19	-

Pei et al. (2017), where the training and testing sets are provided online. In addition, we test the proposed models on 20 benchmark datasets, most recently utilized by Schäfer and Leser (2017). These 20 datasets are trained on the same training and testing datasets as Schäfer and Leser (2017). These benchmark datasets are from various fields. Some datasets encompass the domains of medical care, speech recognition and motion recognition. Further details of each dataset are depicted in Table 1. The max training length in Table 1 is the maximum number of time steps for the entire sequence. The remaining 10 datasets of various classification tasks were obtained from the UCI repository (Lichman, 2013). "HAR", "EEG2", and the "Occupancy" datasets have predefined training and testing sets. All the remaining datasets are partitioned into training and testing sets with a split ratio of 50:50. Each of the datasets is normalized to have zero mean and unit standard deviation. Furthermore, the datasets are padded with zeros, such that each time series length is equivalent to the maximum length of all variables in the training dataset. The dataset is summarized in Table 1.

4.3. Results

MLSTM-FCN and MALSTM-FCN are applied on all 35 datasets. We compare our results to the existing reported state-of-the-art models (HULM (Pei et al., 2017), HCRF (Quattoni et al., 2007), NL (Jaakkola et al., 2000), FKL (Jaakkola et al., 2000), ARKernel (Cuturi & Doucet, 2011), LPS (Baydogan & Runger, 2016), mv-ARF (Tuncel & Baydogan, 2018), SMTS (Baydogan & Runger, 2015), WEASEL+MUSE (Schäfer & Leser, 2017), and dUFS (Wistuba et al., 2015)) of each dataset. Additionally, we compare our models with LSTM-FCN (Karim et al., 2017), ALSTM-FCN (Karim et al., 2017). Alongside these models, we also obtain baselines for these datasets by testing them on DTW, Random Forest, SVM with a linear kernel, SVM with a 3rd degree polynomial kernel and choose the highest score as the baseline.

Due to the general variance of deep learning algorithms, reproducing exact results is particularly onerous. For replicability, we ran the experiments 3–5 times on various datasets. All the results are similar, where the maximum variance of the accuracy is 3%. The results presented in Table 2 are obtained when the training loss is a minimum. The weights of the models trained on all of

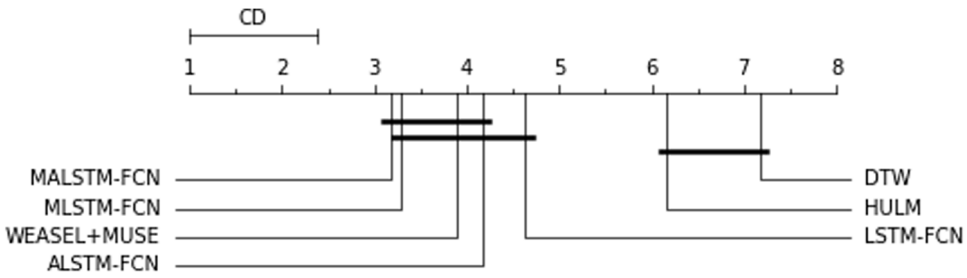

Fig. 3. Critical difference diagram of the arithmetic means of the ranks on all 35 datasets.

Table 3
Wilcoxon signed-rank test comparison of Each Model. Red cells denote models where we fail to reject the hypothesis and claim that the models have similar performance.

	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	DTW	SVM Lin.	SVM Poly	RF	NL	FKL	HCRF	HULM	dUFS	SMTS	LPS	mv-ARF	ARKernel
MLSTM-FCN	2.76E-04																
ALSTM-FCN	2.41E-01	1.93E-03															
MALSTM-FCN	4.40E-04	2.48E-01	3.76E-04														
DTW	8.22E-08	4.72E-09	6.98E-08	3.67E-09													
SVM Lin.	1.11E-10	1.11E-10	1.31E-10	1.11E-10	1.31E-10												
SVM Poly	1.54E-10	1.11E-10	1.54E-10	1.11E-10	2.14E-10	2.81E-10											
RF	3.68E-10	1.54E-10	4.56E-10	2.26E-10	1.88E-09	1.17E-10	1.54E-10										
NL	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.24E-10	1.17E-10	1.46E-10	1.82E-10									
FKL	1.11E-10	1.31E-10	1.24E-10	1.24E-10	1.63E-10	1.11E-10	1.17E-10	1.17E-10	1.31E-10								
HCRF	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.72E-10	1.11E-10	1.11E-10	1.31E-10	1.31E-10	1.63E-10							
HULM	1.17E-10	1.38E-10	1.31E-10	1.11E-10	1.63E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.46E-10	1.46E-10						
dUFS	1.06E-09	2.09E-09	2.20E-09	2.09E-09	1.17E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10					
SMTS	5.22E-09	1.05E-08	1.05E-08	7.42E-09	4.32E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	5.35E-10				
LPS	1.41E-08	1.34E-08	1.80E-08	8.19E-09	4.94E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	4.09E-10	1.16E-08			
mv-ARF	1.10E-08	4.27E-09	6.39E-09	2.44E-09	4.56E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	2.88E-10	7.60E-09	7.80E-09		
ARKernel	4.27E-09	1.61E-09	1.61E-09	1.38E-09	5.79E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	5.64E-10	1.28E-08	1.05E-08	9.05E-09	
WEASEL MUSE	2.84E-09	1.05E-08	6.71E-09	9.51E-09	1.31E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	1.11E-10	2.32E-09	2.44E-09	1.79E-09	1.18E-09	5.07E-10

Table 4
Comparison of MLSTM-FCN With and Without Dimension Shuffle.

	MLSTM-FCN with dimension shuffle	MLSTM-FCN without dimension shuffle
MPCE	4.21	4.86
Time (h)	13	32

these datasets are provided online. In addition, we provide our training and evaluation scripts that will simplify the replication of similar results.[1]

Table 2 compares the performance of various models with MLSTM-FCN and MALSTM-FCN. We define performance as the classification accuracy of a model on a particular dataset. Two datasets, "Activity" and "Action 3d", required a strided temporal convolution (stride 2) prior to the LSTM branch to reduce the amount of memory consumed when using the MALSTM-FCN model, because the models were too large to fit on a single GTX 1080 Ti processor otherwise. Both of the proposed models, MLSTM-FCN and MALSTM-FCN, outperform the state-of-the-art models (SOTA) on 28 and 27 out of the 35 datasets of this experiment respectively. "Activity" is one of the few datasets where the proposed models did not outperform the SOTA model. We postulate that the low performance is due to the large stride of the convolution prior to the LSTM branch, which led to a loss of valuable information.

MLSTM-FCN and MALSTM-FCN have an average arithmetic rank of 3.29 and 3.17 respectively, and a geometric rank of 2.58 and 2.42 respectively. Fig. 3 depicts the superiority of the proposed models over the top existing models through a critical difference diagram (that applies a Nemenyi test (Nemenyi, 1962) of the average arithmetic ranks.

We perform a Wilcoxon signed-rank test to compare all models that were tested on all 35 datasets, as shown in Table 3. A Dunn–Sidak correction (Šidák, 1967) is applied to control the familywise error rate, resulting in the adjusted significance of 0.0028. We statistically conclude that the proposed models have a performance score higher than the remaining model as the p-values are below 0.28 percent. The Wilcoxon signed-rank test also demonstrates the performance of MLSTM-FCN and MALSTM-FCN to be the same. Both MLSTM-FCN and MALSTM-FCN perform significantly better than LSTM-FCN and ALSTM-FCN. This indicates the *squeeze-and-excitation* block enhances performance significantly on multivariate time series classification through modeling the inter-dependencies between the variables.

4.4. Ablation tests

An ablation study is conducted to determine the effect of dimension shuffle on the input to the LSTM block of the proposed models. We compare the MLSTM-FCN with and without dimension shuffle on all 35 datasets, keeping the number of LSTM cells the same as obtained via grid search for the original models. All other parameters are kept constant. MLSTM-FCN without dimension shuffle took approximately 32 h to process all the datasets on a GTX 1080 Ti GPU. In comparison, MLSTM-FCN with dimension shuffle required 13 h to process all the datasets.

The purpose of this study is to determine the impact of the dimension shuffle operation on classification accuracy. Due to the dimension shuffle operation, the time required for training and evaluation of models is significantly reduced in several cases where the number of variables is less than the number of time steps. A Wilcoxon signed-rank test obtains a p-value of 0.136, indicating that we cannot successfully reject the null-hypothesis of the test. This demonstrates the performance of a model when the dimension shuffle operation is applied is statistically the same as when not applied. MLSTM-FCN with dimension shuffle has an MPCE of 4.21. In contrast, an MLSTM-FCN without dimension shuffle obtained a higher MPCE of 4.86. Table 4 summarizes

[1] The codes and weights of all models are available at https://github.com/houshd/MLSTM-FCN.

Table A.5
Definition of all variables.

Variable	Definition	First Introduced
$\mathbf{h}_t$	Hidden vector at time step t	2.1
$\mathbf{I}$	Projection matrix	2.1
l	Layer	2.1
σ	Sigmoid function	2.1
t	Time step	2.1
$tanh$	Hyperbolic tangent function	2.1
$\mathbf{W}$	Weight matrix	2.1
$\mathbf{x}_t$	Input vector at time step t	2.1
$\mathbf{y}_t$	Prediction at time step t	2.1
$\odot$	Elementwise multiplication	2.2
$\mathbf{c}$	Cell gate	2.2
$\mathbf{f}$	Forget gate	2.2
$\mathbf{g}$	Activation function	2.2
$\mathbf{h}$	Hidden vector	2.2
$\mathbf{m}$	Memory vector	2.2
$\mathbf{o}$	Output gate	2.2
$\mathbf{u}$	Input gate	2.2
a	Alignment	2.3
α_{ij}	Weight	2.3
b_i	Annotation	2.3
e_{ij}	Energy of element	2.3
i	Output position	2.3
j	Input position	2.3
T_x	Maximum length of input sequence x	2.3
V	context vector	2.3
v_i	context vector	2.3
ν	RNN hidden state	2.3
$*$	Convolution operation	2.4
$\mathbf{F}_{sq}(\mathbf{u}_c)$	Channel-wise multiplication between the feature map and the scale	2.4
$\mathbf{F}_{tr}$	Computational unit for any transformation	2.4
$\mathbf{F}_{ex}$	Parameterized as a neural network	2.4
$\mathbf{s}_c$	Channel-wise global average over the temporal dimension T	2.4
H	Spatial dimension	2.4
r	Reduction ratio	2.4
T	Temporal dimension	2.4
$\mathbf{U}$	Outputs of $\mathbf{F}_{tr}$	2.4
$\mathbf{v}_c^s$	2D spatial kernel on channel c	2.4
W	Spatial dimension	2.4
$\mathbf{W}_1$	Learnable parameters of $\mathbf{F}_{ex}$	2.4
$\mathbf{W}_2$	Learnable parameters of $\mathbf{F}_{ex}$	2.4
$\mathbf{X}$	Image of shape $HxWxC$	2.4
$\widetilde{\mathbf{x}}_c$	Output of the block rescaled	2.4
$\mathbf{z}$	Channel wise statistic	2.4
$\mathbf{z}_c$	cth element of $\mathbf{z}$	2.4
δ	ReLU activation function	2.4
σ	Sigmoid activation function	2.4
G_s	Number of output feature maps for stage s	3.1
M	Number of variables processed per time step	3.1
P	Total number of additional parameters	3.1
Q	Maximum number of time steps amongst all variables	3.1
R_s	Repeated block number for stage s	3.1
S	Number of stages	3.1
s	Stage	3.1
C_i	Contribution of each class	4
d	Number of input units to the weight tensor	4
Gw_i	loss scaling weight for the ith class	4
N	number of samples in the dataset	4
N_{C_i}	number of samples that belong to class C_i	4
U	uniform distribution	4
K	dataset	4.1
$MPCE$	mean per class error	4.1
$\mathbf{N}$	number of datasets	4.1
PCE_k	per class error for dataset k	4.1

how dimension shuffle affects MLSTM-FCN. In other words, the dimension shuffle operation reduces the processing time by 59 percent while maintaining the same classification accuracy.

5. Conclusion & future work

The two proposed models attain state-of-the-art results in most of the datasets tested, 28 out of 35 datasets. Each of the proposed models requires minimal preprocessing and feature extraction. Furthermore, the addition of the *squeeze-and-excitation* block improves the performance of LSTM-FCN and ALSTM-FCN significantly. We provide a comparison of our proposed models to other existing state-of-the-art algorithms.

The proposed models will be beneficial in various multivariate time series classification tasks, such as activity recognition, or action recognition. The proposed models can quickly be deployed in real-time systems and embedded systems because the proposed models are small and efficient. Further research is being done to better understand why the *squeeze-and-excitation* block does not

match the performance of the general LSTM-FCN or ALSTM-FCN models on a couple of datasets.

Appendix. Variable definitions

See Table A.5.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org, URL https://www.tensorflow.org/.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. ArXiv preprint arXiv:1409.0473.

Baydogan, M. G., & Runger, G. (2015). Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, *29*(2), 400–422.

Baydogan, M. G., & Runger, G. (2016). Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, *30*(2), 476–509.

Carnegie Mellon University - CMU Graphics Lab - motion capture library. (2000). URL http://mocap.cs.cmu.edu/.

Chollet, F., et al. (2015). *Keras*. GitHub, https://github.com/fchollet/keras.

Cui, Y., Shi, J., & Wang, Z. (2015). Complex rotation quantum dynamic neural networks (CRQDNN) using complex quantum neuron (CQN): applications to time series prediction. *Neural Networks*, *71*, 11–26.

Cuturi, M., & Doucet, A. (2011). Autoregressive kernels for time series. ArXiv preprint arXiv:1101.0673.

Fu, Y. (2015). *Human activity recognition and prediction*. Springer.

Geurts, P. (2001). Pattern extraction for time series classification. In *PKDD, Vol. 1* (pp. 115–127). Springer.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610.

Graves, A., et al. (2012). *Supervised sequence labelling with recurrent neural networks, Vol. 385*. Springer.

Hammami, N., & Bedda, M. (2010). Improved tree model for arabic speech recognition. In *Computer science and information technology, 2010 3rd IEEE international conference on, Vol. 5* (pp. 521–526). IEEE.

He, K., Zhang, X., Ren, S., & Sun, J. (2015) .Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hu, J., Shen, L., & Sun, G. (2017). Squeeze-and-excitation networks. ArXiv preprint arXiv:1709.01507.

Jaakkola, T., Diekhans, M., & Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, *7*(1–2), 95–114.

Kadous, M. W. (2002). Temporal classification: extending the classification paradigm to multivariate time series. New South Wales, Australia.

Kang, H., & Choi, S. (2014). Bayesian common spatial patterns for multi-subject EEG classification. *Neural Networks*, *57*, 39–50.

Karim, F., Majumdar, S., Darabi, H., & Chen, S. (2017). Lstm fully convolutional networks for time series classification. *IEEE Access*, 1–7.

Kehagias, A., & Petridis, V. (1997). Predictive modular neural networks for time series classification. *Neural Networks*, *10*(1), 31–49.

King, G., & Zeng, L. (2001). Logistic regression in rare events data. *Political Analysis*, *9*(2), 137–163.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. ArXiv preprint arXiv:1412.6980.

Li, W., Zhang, Z., & Liu, Z. (2010). Action recognition based on a bag of 3d points. In *Computer vision and pattern recognition workshops, 2010 IEEE computer society conference on* (pp. 9–14). IEEE.

Lichman, M. (2013). *UCI machine learning repository*. University of California, Irvine, School of Information and Computer Sciences, URL http://archive.ics.uci.edu/ml.

Maaten, L. (2011). Learning discriminative fisher kernels. In: *Proceedings of the 28th international conference on machine learning* (pp. 217–224).

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics, Vol. 18* (p. 263). International Biometric Soc 1441 I ST, NW, SUITE 700, Washington, DC 20005-2210.

Olszewski, R. T. (2012). Bobski's world. [link]. URL http://www.cs.cmu.edu/~bobski/.

Orsenigo, C., & Vercellis, C. (2010). Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification. *Pattern Recognition*, *43*(11), 3787–3794.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. ArXiv preprint arXiv:1312.6026.

Pavlovic, V., Frey, B. J., & Huang, T. S. (1999). Time-series classification using mixed-state dynamic Bayesian networks. In *Computer vision and pattern recognition, 1999. IEEE computer society conference on, Vol. 2* (pp. 609–615). IEEE.

Pei, W., Dibeklioğlu, H., Tax, D. M., & van der Maaten, L. (2017). Multivariate time-series classification using the hidden-unit logistic model. *IEEE Transactions on Neural Networks and Learning Systems*.

Prieto, O. J., Alonso-González, C. J., & Rodríguez, J. J. (2015). Stacking for multivariate time series classification. *Pattern Analysis and Applications*, *18*(2), 297–312.

Quattoni, A., Wang, S., Morency, L.-P., Collins, M., & Darrell, T. (2007). Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(10).

Schäfer, P., & Leser, U. (2017). Multivariate time series classification with WEASEL+ MUSE. ArXiv preprint arXiv:1711.11343.

Seto, S., Zhang, W., & Zhou, Y. (2015). Multivariate time series classification using dynamic time warping template selection for human activity recognition. In *Computational Intelligence, 2015 IEEE Symposium Series on* (pp. 1399–1406). IEEE.

Sharabiani, A., Darabi, H., Rezaei, A., Harford, S., Johnson, H., & Karim, F. (2017). Efficient classification of long time series by 3-d dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Šidák, Z. (1967). Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, *62*(318), 626–633.

Spiegel, S., Gaebler, J., Lommatzsch, A., De Luca, E., & Albayrak, S. (2011). Pattern recognition and classification for multivariate time series. In *Proceedings of the fifth international workshop on knowledge discovery from sensor data* (pp. 34–42). ACM.

Sübakan, Y. C., Kurt, B., Cemgil, A. T., & Sankur, B. (2014). Probabilistic sequence clustering with spectral learning. *Digital Signal Processing* , *29*, 1–19.

Tuncel, K. S., & Baydogan, M. G. (2018). Autoregressive forests for multivariate time series modeling. *Pattern Recognition*, *73*, 202–215.

van der Maaten, L., & Hendriks, E. (2012). Action unit classification using active appearance models and conditional random fields. *Cognitive Processing*, *13*(2), 507–518.

Wang, J., Liu, Z., Wu, Y., & Yuan, J. (2012). Mining actionlet ensemble for action recognition with depth cameras. In *Computer vision and pattern recognition, 2012 IEEE conference on* (pp. 1290–1297). IEEE.

Wang, L., Wang, Z., & Liu, S. (2016). An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm. *Expert Systems with Applications*, *43*, 237–249.

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *Neural networks, 2017 international joint conference on* (pp. 1578–1585). IEEE.

Williams, B., Toussaint, M., & Storkey, A. J. (2008). Modelling motion primitives and their timing in biologically executed movements. In *Advances in neural information processing systems* (pp. 1609–1616).

Wistuba, M., Grabocka, J., & Schmidt-Thieme, L. (2015). Ultra-fast shapelets for time series classification. ArXiv preprint arXiv:1503.05018.

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, *12*(1), 40–48.

Yu, Z., & Lee, M. (2015). Real-time human action classification using a dynamic neural model. *Neural Networks*, *69*, 29–43.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management* (pp. 298–310). Springer.