Multivariate LSTM-FCNs for Time Series Classification

Fazle Karim[1], Somshubra Majumdar[2], Houshang Darabi[1], *Senior Member, IEEE,* and Samuel Harford[1]

Abstract—Over the past decade, multivariate time series classification has been receiving a lot of attention. We propose augmenting the existing univariate time series classification models, LSTM-FCN and ALSTM-FCN with a squeeze and excitation block to further improve performance. Our proposed models outperform most of the 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.

Keywords—*Convolutional neural network, long short term memory, recurrent neural network, multivariate time series classification*

I. Introduction

Time series data is used in various fields of studies, ranging from weather readings to psychological signals [1, 2]. A time series is a sequence of data points in a time domain, typically in a uniform interval [3]. There is a significant increase of time series data being collected by sensors [4]. A time series dataset can be univariate, where a sequence of measurements from the same variable are collected, or multivariate, where sequence of measurements from multiple variables are collected [5]. Over the past decade, multivariate time series classification has received significant interest. Some of the applications where multivariate time series classification is used are in healthcare, activity recognition, object recognition, and action recognition [6]–[8]. In this paper, we propose two deep learning models that outperform existing state of the art 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 [9]. Plenty of research indicates Dynamic Time Warping (DTW) as the best distance based measure to use along k-NN [10].

In addition to distance based metrics, other traditional feature based algorithms are used. Typically, feature based classification algorithms rely heavily on the features being extracted from the time series data [11]. However, feature extraction is very difficult 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 [12]. Hidden State Conditional Random Field (HCRF) and Hidden Unit Logistic Model (HULM) are

two successful feature based algorithms that have led to state of the art results on various benchmark datasets, ranging from online character recognition to activity recognition [13]. HCRF is a computationally expensive algorithm that detects latent structures of the input time series data using a chain of k-nomial latent variables. The number of parameters in the model increases linearly with the total number of latent states required [14]. Further, datasets that require 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 [13].

Traditional models, such as the naive logistic model (NL) and Fisher kernel learning (FKL) [15], 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 [13]. The FKL model is effective on time series classification problems when based 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. [15, 16]

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) [17] applies a random forest on the multivariate time series to partition it into leaf nodes, which are each represented by a word to form a codebook. These words are used with another random forest to classify the multivariate time series. Learned Pattern Similarity (LPS) [18] 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) [19] obtains random shapelets from the multivariate time series and applies a linear SVM or a Random Forest classifier. Subsequently, UFS was enhanced by additionally computing derivatives as features (dUFS) [19]. The Auto-Regressive (AR) kernel [20] applies an AR kernel-based distance measure to classify the multivariate time series. Auto-Regressive forests for multivariate time series modelling (mv-ARF) [21] uses a tree ensemble. Each tree is trained with a different time lags. Most recently, WEASEL+MUSE [22] 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

---

[1]Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago,IL
[2]Computer Science, University of Illinois at Chicago, Chicago, IL

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 a MLP to perform classification [12].

This paper proposes two deep learning models for multivariate time series classification. The proposed models require minimal preprocessing and are tested on 35 datasets, obtaining strong performance in most of them. The rest of the paper is ordered as follows. The background works are discussed in Section II. We present the architecture of the two proposed models in Section III. In Section IV, we discuss the dataset, test the models on, present our results and analyze our findings. In Section V we draw our conclusion.

II. BACKGROUND WORKS

A. Temporal Convolutions

In the proposed models, Temporal Convolutional Networks are used as a feature extraction module of the Fully Convolutional Network (FCN) branch. Typically, a basic convolution block contains a convolution layer, which is accompanied by a batch normalization [23]. The batch normalization is followed by an activation function of either a Rectified Linear Unit or a Parametric Rectified Linear Unit [24].

Generally, Temporal Convolutional Networks have an input of a time series signal. *Lea et al.* [25] defines $X_t \in \mathbb{R}^{F_0}$ to be the input feature vector of length F_0 for time step t. t is > 0 and $\leq T$, where T is the number of time steps of a sequence. Each frame has an action label, y_t, where $y_t \in \{1, ..., C\}$. C is the number of classes.

Each of the L convolutional layers has a 1D filter applied to it, such that the evolution of the input signals over time is captured. *Lea et al.* [25] uses a tensor $W^{(l)} \in \mathbb{R}^{F_l \times d \times F_{l-1}}$ and biases $b^{(l)} \in \mathbb{R}^{F_l}$ to parameterize the 1D filter. The layer index is defined as $l \in \{1, ..., L\}$ and the filter duration is defined by d. The i-th component of the unnormalized activation $\hat{\mathbf{E}}_t^{(l)} \in \mathbb{R}^{F_l}$ for the l-th layer is a function of the incoming normalized activation matrix $E^{(l-1)} \in \mathbb{R}^{F_{l-1} \times T_{l-1}}$ from the previous layer

$$\hat{\mathbf{E}}_{i,t}^{(l)} = f\left(b_i^{(l)} + \sum_{t'=1}^{d} \left\langle W_{i,t',.}^{(l)}, E_{.,t+d-t'}^{(l-1)} \right\rangle\right) \tag{1}$$

for each time t where $f(\cdot)$ is a Rectified Linear Unit.

B. 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 et al.* [26] state RNN to maintain a hidden vector $\mathbf{h}$ that is update at time step t,

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{I}\mathbf{x}_t), \tag{2}$$

where the hyperbolic tangent function is represented by $tanh$, 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, $\mathbf{W}$,

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}\mathbf{h}_{t-1}). \tag{3}$$

The softmax creates a normalized probability distribution over all possible classes using a logistic sigmoid function, σ. RNNs can be stacked to create deeper networks by using the hidden state, $\mathbf{h}$ as an input to another RNN,

$$\mathbf{h}_t^l = \sigma(\mathbf{W}\mathbf{h}_{t-1}^l + \mathbf{I}\mathbf{h}_t^{l-1}). \tag{4}$$

C. Long Short-Term Memory RNNs

A major issue with RNNs is they contain a vanishing gradient problem. Long short-term memory (LSTM) RNNs address this problem by integrating gating functions into their state dynamics [27]. 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.* [28] as the following:

$$\begin{aligned}
\mathbf{g}^u &= \sigma(\mathbf{W}^u\mathbf{h}_{t-1} + \mathbf{I}^u\mathbf{x}_t) \\
\mathbf{g}^f &= \sigma(\mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{I}^f\mathbf{x}_t) \\
\mathbf{g}^o &= \sigma(\mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{I}^o\mathbf{x}_t) \\
\mathbf{g}^c &= \tanh(\mathbf{W}^c\mathbf{h}_{t-1} + \mathbf{I}^c\mathbf{x}_t) \\
\mathbf{m}_t &= \mathbf{g}^f \odot \mathbf{m}_{t-1} \; + \; \mathbf{g}^u \odot \mathbf{g}^c \\
\mathbf{h}_t &= \tanh(\mathbf{g}^o \odot \mathbf{m}_t)
\end{aligned} \tag{5}$$

where 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 $\mathbf{W}^u, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^c$.

LSTMs can learn temporal dependencies. However, long term dependencies of long sequence are challenging to learn using LSTMs. *Bahdanau et al.* [29] propose using an attention mechanism to learn these long term dependencies.

D. Attention Mechanism

An attention mechanism conditions a context vector C on the target sequence y. This method is commonly used in neural translation of texts. *Bahdanau et al.* [29] argues the context vector c_i depends on a sequence of *annotations* $(h_1, ..., h_{T_x})$, where an encoder maps the input sequence. Each annotation, h_i, comprises of information on the whole input sequence, while focusing on regions surrounding the i-th word of the input sequence. The weighted sum of each annotation, h_i, is used to compute the context vector as follows:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{6}$$

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

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}, \tag{7}$$

where the *alignment model*, e_{ij}, is $a(s_{i-1}, h_j)$. The *alignment model* measures how well the input position, j, and the output at position, i, match using the RNN hidden state, s_{i1}, and the j-th annotation, h_j, of the input sentence. *Bahdanau et al.* [29] uses a feedforward neural network to parameterize the alignment model, a. The feedforward neural network is trained 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 whole translation model simultaneously [29].

E. Squeeze and Excite Block

Hu et al. [30] propose a *Squeeze-and-Excitation* block that acts as a computational unit for any transformation $\mathbf{F}_{tr} : \mathbf{X} \to \mathbf{U}, \mathbf{X} \in \mathbb{R}^{W' \times H' \times C'}, \mathbf{U} \in \mathbb{R}^{W \times H \times C}$. The outputs of $\mathbf{F}_{tr}$ are represented as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_C]$ where

$$\mathbf{u}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s. \qquad (8)$$

The convolution operation is depicted by *, and the 2D spatial kernel is depicted by $\mathbf{v}_c^s$. The single channel of $\mathbf{v}_c$ acts on the corresponding channel of X. *Hu et al.* [30] 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, $\mathbf{z} \in \mathbb{R}^C$. The c-th element of z is calculated by:

$$\mathbf{z}_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} u_c(i, j). \qquad (9)$$

For temporal sequence data, the transformation output, U, is shrunk through the temporal dimension T to compute the channel-wise statistics, $\mathbf{z} \in \mathbb{R}^C$. The c-th element of z is then calculated by:

$$\mathbf{z}_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{T} \sum_{t=1}^{T} u_c(t). \qquad (10)$$

The aggregated information 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:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 z)), \qquad (11)$$

where δ is a ReLU activation function, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{\frac{C}{r} \times C}$. $\mathbf{W}_1$ and $\mathbf{W}_2$ is 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:

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, \mathbf{s}_c) = \mathbf{s}_c \cdot \mathbf{u}_c, \qquad (12)$$

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

III. MULTIVARIATE LSTM FULLY CONVOLUTIONAL NETWORK

A. Network Architecture

LSTM-FCN and ALSTM-FCN have been successful in classifying univariate time series [31]. The models we propose, Multivariate LSTM-FCN (MLSTM-FCN) and Multivariate ALSTM-FCN (MALSTM-FCN) augment LSTM-FCN and ALSTM-FCN respectively.

Similar to LSTM-FCN and ALSTM-FCN, the proposed models comprise of 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. Each convolutional block contains a convolutional layer, with filter size of 128 or 256, and is succeeded by a batch normalization, with a momentum of 0.99 and epsilon of 0.001. The batch normalization layer is succeeded by the ReLU activation. 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. The added *squeeze and excite* block enhances the performance of the LSTM-FCN and ALSTM-FCN models. The final temporal convolutional block is followed by a global average pooling layer.

On the other hand, the multivariate time series input is passed through a dimension shuffle layer, explained more in section III-B, followed by the LSTM block. The LSTM block is identical to the block from the LSTM-FCN or ALSTM-FCN models [31], comprising of either a LSTM layer or an Attention LSTM layer, which is followed by a dropout. Since the datasets are padded by zeros at the end to make their size consistent, we use a mask prior to the LSTM or Attention LSTM layer to skip time steps for which we have no information.

B. 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 *N* timesteps having *M* distinct variables per timestep. If there is a time series with *M* variables and *N* time steps, the fully convolutional block will receive the data as such.

On the other hand, 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 input to a LSTM does not go through the dimension shuffle, the LSTM will require *N* time steps to process *M* variables at each timestep. However, if the dimension shuffle is applied, the LSTM will require *M* time steps to process *N*
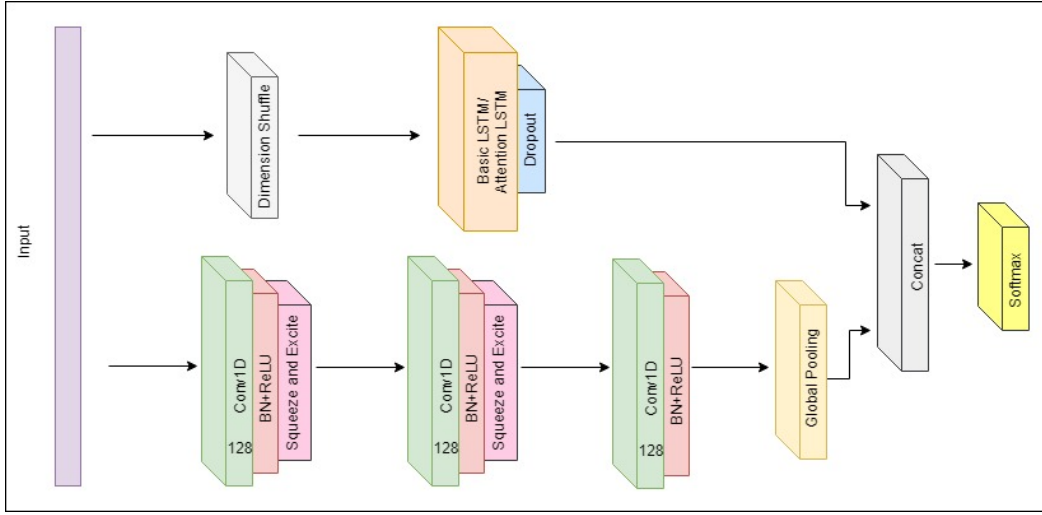
Fig. 1: The MLSTM-FCN architecture. LSTM cells can be replaced by Attention LSTM cells to construct the MALSTM-FCN architecture.
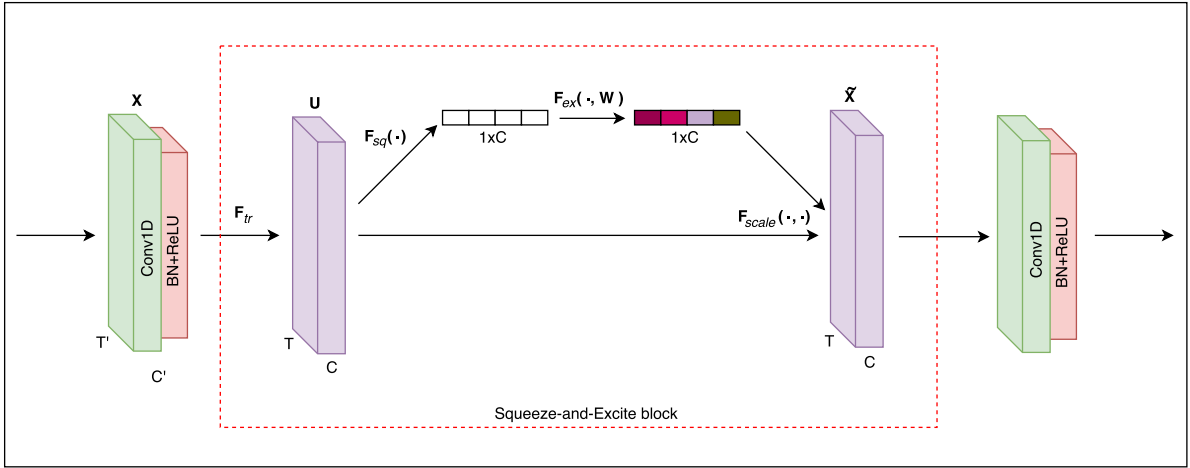

Fig. 2: The computation of the temporal *Squeeze-and-Excite* block.

variables. 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 *N*. In the proposed model, the dimension shuffle operation is only applied when the number of time steps, *N*, is greater than the number of variables, *M*.

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

IV. EXPERIMENTS

MLSTM-FCN and MALSTM-FCN have been tested on 35 datasets, further explained in section IV-B1. The optimal num-

ber of LSTM cells for each dataset was found via grid search ranging from 8 cells to 128 cells. In most experiments, we use an initial batch size of 128. The FCN block is comprised of 3 blocks of 128-256-128 filters, so as to be comparable with the original models. During the training phase, we set the total number of training epochs to 250 unless explicitly stated and dropout rate set to 80% to mitigate overfitting. The convolution kernels are initialized with the initialization proposed by *He et al.* [32]. Each of the proposed models is trained using a batch size of 128. For datasets with class imbalance, a class weighing schemed inspired by *King et al.* is utilized [33].

We use the Adam optimizer [34], with an initial learning rate set to 1e-3 and the final learning rate set to 1e-4 to train all models. The datasets were normalized and preprocessed to have zero mean and unit variance. We then append variable

length time series with zeros so as to obtain a time series dataset with a constant length *N*, where *N* is the maximum length of the time series. In addition, after every 100 epochs, the learning rate is reduced by a factor of $1/\sqrt[3]{2}$. We use the Keras [35] library with the Tensorflow backend [36] to train the proposed models. [1]

A. Evaluation Metrics

In this paper, various models, including the proposed models, are evaluated using accuracy, arithmetic rank, geometric rank, Wilcoxon signed rank test, and mean per class error. The arithmetic and geometric rank are the arithmetic and geometric mean of the ranks. The Wilcoxon signed rank test is a non-parametric statistical test that hypothesizes that the median of the rank between the compared models are the same. The alternative hypothesis of the Wilcoxon signed rank test is that the median of the rank between the compared models are not the same. Finally, the mean per class error is the mean of the per class error (PCE) of all the datasets,

$$PCE_k = \frac{1 - accuracy}{number\ of\ unique\ classes}$$
$$MPCE = \frac{1}{K} \sum PCE_K.$$

B. Datasets

A total of 35 datasets are used to test the proposed models. Five of the 35 datasets are benchmark datasets used by *Pei et al.* [13], where the training and testing sets are provided online. In addition, we test the proposed models on 20 benchmark datasets, most recently by utilized *Schäfer and Leser* [22] The remaining datasets are from the UCI repository [37]. The dataset is summarized in Table I.

1) Standard Datasets: The proposed models were tested on twenty-five multivariate datasets. Each of the datasets that were trained were on the same training and testing datasets most resently mentioned by *Pei et al.* [13] and *Schäfer and Leser* [22]. These benchmark datasets are from varying fields. Some of the fields the datasets encompass are in the medical, speech recognition and motion recognition fields. Further details of each dataset are depicted in Table I.

2) Multivariate Datasets from UCI: The remaining 10 datasets of various classification tasks were obtained from the UCI repository. "HAR", "EEG2", and the "Occupancy" datasets have training and testing sets provided. All the remaining datasets are partitioned into training and testing sets with a split ratio of 50-50. Each of the datasets is normalized with a mean of 0 and a standard deviation of 1, and padded with zeros as required.

C. Results

MLSTM-FCN and MALSTM-FCN is applied on three sets of experiments, benchmark datasets used by *Pei et al.*,

---

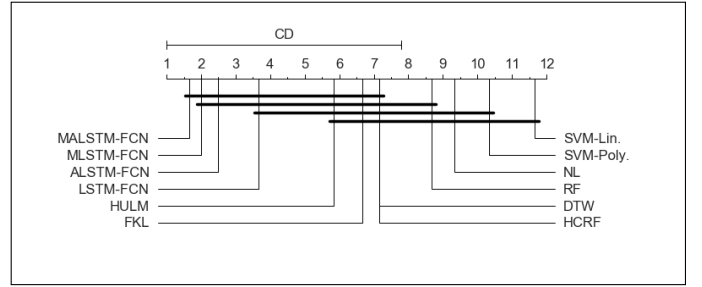[1]The codes and weights of each models are available at https://github.com/houshd/MLSTM-FCN

Fig. 3: Critical difference diagram of the arithmetic means of the ranks on benchmark datasets utilized by *Pei et al.*

benchmark datasets used by *Schäfer and Leser* and datasets found in a variety of repositories. We compare our performance with HULM [13], HCRF [14], NL and FKL [15] only on the datasets used by *Pei et al.* The proposed models are also compared to the results of ARKernel [20], LPS [18], mv-ARF [21], SMTS [17], WEASEL+MUSE [22], and dUFS [19] on the benchmark datasets used by *Schäfer and Leser*. Additionally, we compare our performance with LSTM-FCN [31], ALSTM-FCN [31]. Alongside these models, we also obtain baselines for these dataset 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.

1) Standard Datasets:

a) Multivariate Datasets Used By Pei et al.: Table III compares the performance of various models with MLSTM-FCN and MALSTM-FCN. Two datasets, "Activity" and "Action 3d", required a strided temporal convolution (stride 3 and stride 2 respectively) 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 the proposed models outperform the state of the art models (SOTA) on five out of the six datasets of this experiment. "Activity" is the only dataset 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 2 and 1.4 respectively, and a geometric rank of 2.05 and 1.23 respectively. Fig. 3 depicts the superiority of the proposed models over existing models through a critical difference diagram of the average arithmetic ranks. The MPCE of both the proposed models are below 0.71 percent. In comparison, HULM, the prior SOTA on these datasets, has a MPCE of 1.03 percent.

b) Multivariate Datasets Used By Schäfer and Leser: MALSTM-FCN and MLSTM-FCN are compared to seven state of the art models, using results reported by their respective authors in their publications. The results are summarized in Table III. Over the 20 datasets, MLSTM-FCN and MALSTM-FCN far outperforms all of the existing state of the art models.

TABLE I: Properties of all datasets. The yellow cells are datasets used by *Pei et al.* [13], the purple cells are datasets used by *Schäfer and Leser* [22], and the blue cells are datasets from the UCI repository [37].

Dataset	Num. of Classes	Num. of Variables	Max Training Length	Task	Train-Test Split	Source
OHC	20	30	173	Handwriting Classification	10-fold	[38]
Arabic Voice	88	39	91	Speaker Recognition	75-25 split	[39]
Cohn-Kanade AU-coded Expression (CK+)	7	136	71	Facial Expression Classification	10-fold	[40]
MSR Action	20	570	100	Action Recognition	5 ppl in train; rest in test	[41]
MSR Activity	16	570	337	Activity Recognition	5 ppl in train; rest in test	[42]
ArabicDigits	10	13	93	Digit Recognition	75-25 split	[37]
AUSLAN	95	22	96	Sign Language Recognition	44-56 split	[37]
CharacterTrajectories	20	3	205	Handwriting Classification	10-90 split	[37]
CMU_MOCAP_S16	2	62	534	Action Recognition	50-50 split	[43]
DigitShape	4	2	97	Action Recognition	60-40 split	[44]
ECG	2	2	147	ECG Classification	50-50 split	[45]
JapaneseVowels	9	12	26	Speech Recognition	42-58 split	[37]
KickvsPunch	2	62	761	Action Recognition	62-38 split	[43]
LIBRAS	15	2	45	Sign Language Recognition	38-62 split	[37]
LP1	4	6	15	Robot Failure Recogntion	43-57 split	[37]
LP2	5	6	15	Robot Failure Recogntion	36-64 split	[37]
LP3	4	6	15	Robot Failure Recogntion	36-64 split	[37]
LP4	3	6	15	Robot Failure Recogntion	36-64 split	[37]
LP5	5	6	15	Robot Failure Recogntion	39-61 split	[37]
NetFlow	2	4	994	Action Recognition	60-40 split	[44]
PenDigits	10	2	8	Digit Recognition	2-98 split	[37]
Shapes	3	2	97	Action Recognition	60-40 split	[44]
Uwave	8	3	315	Gesture Recognition	20-80 split	[37]
Wafer	2	6	198	Manufacturing Classification	25-75 split	[45]
WalkVsRun	2	62	1918	Action Recognition	64-36 split	[43]
AREM	7	7	480	Activity Recognition	50-50 split	[37]
HAR	6	9	128	Activity Recognition	71-29 split	[37]
Daily Sport	19	45	125	Activity Recognition	50-50 split	[37]
Gesture Phase	5	18	214	Gesture Recognition	50-50 split	[37]
EEG	2	13	117	EEG Classification	50-50 split	[37]
EEG2	2	64	256	EEG Classification	20-80 split	[37]
HT Sensor	3	11	5396	Food Classification	50-50 split	[37]
Movement AAL	2	4	119	Movement Classification	50-50 split	[37]
Occupancy	2	5	3758	Occupancy Classification	35-65 split	[37]
Ozone	2	72	291	Weather Classification	50-50 split	[37]

TABLE II: 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. * denotes best model.

	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	SOTA
Activity	53.13	61.88	55.63	58.75	66.25 [DTW]
Action 3d	71.72	75.42*	72.73	74.74	70.71 [DTW]
CK+	96.07	96.43	97.10	97.50*	93.56 [13]
Arabic-Voice	98.00	98.00	98.55*	98.27	94.55 [13]
OHC	99.96*	99.96*	99.96*	99.96*	99.03 [13]
Count	4.00	4.00	4.00	4.00	
Arith. Mean	3.20	2.00	2.00	1.40	-
Geom. Mean	2.74	2.05	1.64	1.23	-
MPCE	0.83	0.70	0.77	0.71	-

The average arithmetic rank of MLSTM-FCN is 2.25 and its average geometric rank is 1.79. The average arithmetic rank and average geometric rank of MALSTM-FCN is 3.15 and 2.47, respectively. The current existing state of the art model, WEASEL+MUSE, has an average arithmetic rank of 4.15 and an average geometric rank of 2.98. Fig. 4, a critical difference diagram of the average arithmetic ranks, indicates the performance dominance of MLSTM-FCN and MALSTM-FCN over all other state of the art models. Furthermore, the MPCE of MLSTM-FCN and MALSTM-FCN are 0.016 and

0.018. WEASEL+MUSE has a MPCE of 0.016.

2) Multivariate Datasets from UCI: The performance of MLSTM-FCN and MALSTM-FCN on various multivariate datasets is summarized in Table IV. The colored cells represent the highest performing model on a particular dataset. Both the proposed models outperform the baseline models on all the datasets in this experiment. The arithmetic rank of MLSTM-FCN is 1.33 and the arithmetic rank of MALSTM-FCN is 1.42. In addition, the geometric rank of MLSTM-FCN and MALSM-FCN is 1.26 and 1.33 respectively. MLSTM-

TABLE III: Performance comparison of proposed models with the rest on 20 benchmark datasets. Green cells denote model with best performance. * denotes best model.

Datasets	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	SOTA
ArabicDigits	1.00*	1.00*	0.99	0.99	0.99 [22]
AUSLAN	0.97	0.97	0.96	0.96	0.98 [22]
CharacterTrajectories	0.99	1.00*	0.99	1.00*	0.99 [22]
CMUsubject16	1.00*	1.00*	1.00*	1.00*	1.00 [21]
DigitShapes	1.00*	1.00*	1.00*	1.00*	1.00 [21]
ECG	0.85	0.86	0.86	0.86	0.93 [22]
JapaneseVowels	0.99	1.00*	0.99	0.99	0.98 [20]
KickvsPunch	0.90	1.00*	0.90	1.00*	1.00 [20]
Libras	0.99*	0.97	0.98	0.97	0.95 [20]
LP1	0.74	0.86	0.80	0.82	0.96 [22]
LP2	0.77	0.83*	0.80	0.77	0.76 [22]
LP3	0.67	0.80*	0.80*	0.73	0.79 [22]
LP4	0.91	0.92	0.89	0.93	0.96 [22]
LP5	0.61	0.66	0.62	0.67	0.71 [22]
NetFlow	0.94	0.95	0.93	0.95	0.98 [22]
PenDIgits	0.97*	0.97*	0.97*	0.97*	0.95 [20]
Shapes	1.00*	1.00*	1.00*	1.00*	1.00 [22]
UWave	0.97	0.98*	0.97	0.98*	0.98 [22]
Wafer	0.99*	0.99*	0.99*	0.99*	0.99 [22]
WalkvsRun	1.00*	1.00*	1.00*	1.00*	1.00 [22]
Count	11.00	14.00	12.00	13.00	-
Arith. Mean	4.05	2.25	3.85	3.15	-
Geom. Mean	2.88	1.79	2.68	2.47	-
MPCE	0.023	0.016	0.021	0.018	-

TABLE IV: Performance comparison of proposed models with the rest on multivariate datasets from various online database. Green cells denote model with best performance. * denotes best model.

	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	Baseline
AREM	89.74	92.31*	82.05	92.31*	76.92 [DTW]
Daily Sport	99.65	99.65	99.63	99.72*	98.42 [DTW]
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]
Movement AAL	73.25	79.63*	70.06	78.34	65.61 [SVM-Poly]
Occupancy	71.05	76.31*	71.05	72.37	67.11 [DTW]
Ozone	67.63	81.50*	79.19	79.78	75.14 [DTW]
Count	7.00	10.00	10.00	10.00	-
Arith. Mean	3.92	1.33	3.67	1.42	-
Geom. Mean	3.70	1.26	3.61	1.33	-
MPCE	8.28	6.49	7.72	6.81	-

TABLE V: 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.

|	ALSTM-FCN	LSTM-FCN	MALSTM-FCN	MLSTM-FCN	ARKernel	DTW	FKL	HCRF	HULM	LPS	NL	RF	SMTS	SVM-Lin.	SVM-Poly.	WEASEL+MUSE	dUFS
LSTM-FCN	1.49E-01																
MALSTM-FCN	2.47E-04	8.65E-06															
MLSTM-FCN	1.25E-04	8.47E-06	3.13E-01														
ARKernel	2.93E-06	3.75E-06	1.25E-06	7.44E-07													
DTW	2.38E-07	4.33E-07	1.25E-07	9.09E-08	7.74E-08												
FKL	9.09E-08	9.09E-08	7.74E-08	7.74E-08	7.74E-08	1.35E-07											
HCRF	7.74E-08	7.74E-08	7.74E-08	7.74E-08	7.74E-08	1.46E-07	1.35E-07										
HULM	7.74E-08	8.39E-08	7.74E-08	7.74E-08	7.74E-08	1.35E-07	1.15E-07	1.15E-07									
LPS	7.79E-06	2.70E-05	1.92E-06	1.37E-06	3.26E-05	7.74E-08	7.74E-08	7.74E-08	7.74E-08								
NL	7.74E-08	7.74E-08	7.74E-08	7.74E-08	7.74E-08	9.09E-08	9.84E-08	9.84E-08	7.74E-08	7.74E-08							
RF	1.14E-07	1.58E-07	7.74E-08	7.74E-08	7.74E-08	9.52E-08	8.39E-08	9.84E-08	7.74E-08	7.74E-08	1.58E-07						
SMTS	6.16E-06	2.38E-05	2.39E-06	4.17E-07	3.67E-05	7.74E-08	7.74E-08	7.74E-08	7.74E-08	4.38E-05	7.74E-08	7.74E-08					
SVM-Lin.	7.74E-08	8.39E-08	7.74E-08	7.74E-08	7.74E-08	9.84E-08	7.74E-08	7.74E-08	7.74E-08	7.74E-08	8.39E-08	8.39E-08	7.74E-08				
SVM-Poly.	7.74E-08	7.74E-08	7.74E-08	7.74E-08	7.74E-08	2.95E-07	8.39E-08	7.74E-08	7.74E-08	7.74E-08	1.15E-07	1.72E-07	7.74E-08	2.95E-07			
WEASEL+MUSE	7.04E-05	3.65E-05	3.81E-05	1.87E-05	1.76E-06	7.74E-08	7.74E-08	7.74E-08	7.74E-08	5.65E-06	7.74E-08	7.74E-08	8.37E-06	7.74E-08	7.74E-08		
dUFS	2.42E-06	5.40E-06	2.42E-06	1.55E-06	8.41E-07	7.74E-08	7.74E-08	7.74E-08	7.74E-08	5.70E-07	7.74E-08	7.74E-08	7.32E-07	7.74E-08	7.74E-08	6.05E-06	
mv-ARF	4.80E-06	2.32E-05	1.06E-06	4.37E-07	4.90E-05	7.74E-08	7.74E-08	7.74E-08	7.74E-08	3.93E-05	7.74E-08	7.74E-08	2.29E-05	7.74E-08	7.74E-08	2.81E-06	3.69E-07
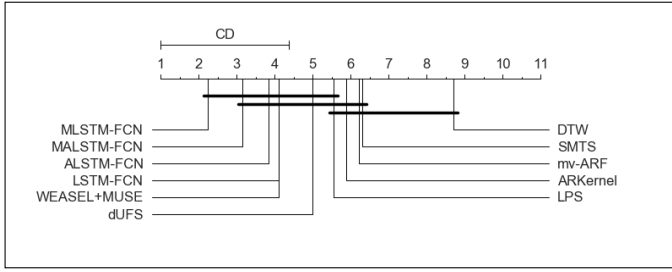
Fig. 4: Critical difference diagram of the arithmetic means of the ranks on 20 benchmark datasets used by *Schäfer and Leser*
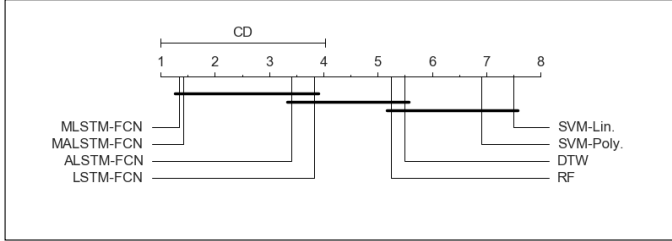

Fig. 5: Critical difference diagram of the arithmetic means of the ranks on multivariate datasets from various online database.

FCN and MALSTM-FCN have a MPCE of 6.49 and 6.81, respectively. In juxtaposition, DTW, a successful algorithm for multivariate time series classification, has an arithmetic rank, geometric rank, and a MPCE of 5.5, 5.34, and 9.68 respectively. A visual representation comparing the arithmetic ranks of each model is depicted in Fig. 5. Fig. 5 depicts both the proposed models outperforming the remaining models by a large margin.

Further, we perform a Wilcoxon signed rank test to compare all models that were tested on all 35 datasets. We statistically conclude that the proposed models have a performance score higher than the remaining model as the p-values are below 5 percent. However, 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 excite layer enhances performance significantly on multivariate time series classification through modelling the inter-dependencies between the channels.

V. 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 require minimal preprocessing and feature extraction. Furthermore, the addition of the squeeze and excite 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 easily be deployed on 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 excite block does not match the performance of the general LSTM-FCN or ALSTM-FCN models on the dataset "Arabic Voice".

REFERENCES

[1] M. W. Kadous, "Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series," *New South Wales, Australia*, 2002.

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

[3] L. Wang, Z. Wang, and S. Liu, "An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm," *Expert Systems with Applications*, vol. 43, pp. 237–249, 2016.

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

[5] O. J. Prieto, C. J. Alonso-González, and J. J. Rodríguez, "Stacking for multivariate time series classification," *Pattern Analysis and Applications*, vol. 18, no. 2, pp. 297–312, 2015.

[6] Y. Fu, *Human Activity Recognition and Prediction*. Springer, 2015.

[7] P. Geurts, "Pattern extraction for time series classification," in *PKDD*, vol. 1. Springer, 2001, pp. 115–127.

[8] V. Pavlovic, B. J. Frey, and T. S. Huang, "Time-series classification using mixed-state dynamic bayesian networks," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999, pp. 609–615.

[9] C. Orsenigo and C. Vercellis, "Combining discrete svm and fixed cardinality warping distances for multivariate time series classification," *Pattern Recognition*, vol. 43, no. 11, pp. 3787–3794, 2010.

[10] S. Seto, W. Zhang, and Y. Zhou, "Multivariate time series classification using dynamic time warping template selection for human activity recognition," in *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE, 2015, pp. 1399–1406.

[11] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 40–48, 2010.

[12] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *International Conference on Web-Age Information Management*. Springer, 2014, pp. 298–310.

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

[14] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 10, 2007.

[15] T. Jaakkola, M. Diekhans, and D. Haussler, "A discriminative framework for detecting remote protein homologies," *Journal of computational biology*, vol. 7, no. 1-2, pp. 95–114, 2000.

[16] L. Maaten, "Learning discriminative fisher kernels," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 217–224.

[17] M. G. Baydogan and G. Runger, "Learning a symbolic representation for multivariate time series classification," *Data Mining and Knowledge Discovery*, vol. 29, no. 2, pp. 400–422, 2015.

[18] M. G. Baydogan and G. Runger, "Time series representation and similarity based on local autopatterns," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 476–509, 2016.

[19] M. Wistuba, J. Grabocka, and L. Schmidt-Thieme, "Ultra-fast shapelets for time series classification," *arXiv preprint arXiv:1503.05018*, 2015.

[20] M. Cuturi and A. Doucet, "Autoregressive kernels for time series," *arXiv preprint arXiv:1101.0673*, 2011.

[21] K. S. Tuncel and M. G. Baydogan, "Autoregressive forests for multivariate time series modeling," *Pattern Recognition*, vol. 73, pp. 202–215, 2018.

[22] P. Schäfer and U. Leser, "Multivariate time series classification with weasel+ muse," *arXiv preprint arXiv:1711.11343*, 2017.

[23] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[24] L. Trottier, P. Giguère, and B. Chaib-draa, "Parametric Exponential Linear Unit for Deep Convolutional Neural Networks," *arXiv*, pp. 1–16, may 2016. [Online]. Available: http://arxiv.org/abs/1605.09332

[25] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks: A Unified Approach to Action Segmentation," in *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 47–54.

[26] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to Construct Deep Recurrent Neural Networks," *arXiv preprint arXiv:1312.6026*, 2013.

[27] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] A. Graves *et al.*, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, vol. 385.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv preprint arXiv:1409.0473*, 2014.

[30] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, 2017.

[31] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE Access*, pp. 1–7, 2017.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[33] G. King and L. Zeng, "Logistic Regression in Rare Events Data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[34] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[37] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[38] B. Williams, M. Toussaint, and A. J. Storkey, "Modelling motion primitives and their timing in biologically executed movements," in *Advances in neural information processing systems*, 2008, pp. 1609–1616.

[39] N. Hammami and M. Bedda, "Improved tree model for arabic speech recognition," in *Computer Science and Information Technology (ICC-SIT), 2010 3rd IEEE International Conference on*, vol. 5. IEEE, 2010, pp. 521–526.

[40] L. van der Maaten and E. Hendriks, "Action unit classification using active appearance models and conditional random fields," *Cognitive processing*, vol. 13, no. 2, pp. 507–518, 2012.

[41] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 9–14.

[42] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1290–1297.

[43] "Carnegie mellon university - cmu graphics lab - motion capture library." [Online]. Available: http://mocap.cs.cmu.edu/

[44] Y. C. Sübakan, B. Kurt, A. T. Cemgil, and B. Sankur, "Probabilistic sequence clustering with spectral learning," *Digital Signal Processing*, vol. 29, pp. 1–19, 2014.

[45] R. T. Olszewski, 2012. [Online]. Available: http://www.cs.cmu.edu/~bobski/