Sequence-to-Sequence Model with Attention for Time Series Classification

Yujin Tang, Jianfeng Xu, Kazunori Matsumoto and Chihiro Ono
KDDI R&D Laboratories, Inc.
2-1-15 Ohara, Fujimino, Saitama, 356-8502 Japan
Email: {yu-tang, ji-xu, matsu, ono}@kddilabs.jp

***Abstract*—Encouraged by recent waves of successful applications of deep learning, some researchers have demonstrated the effectiveness of applying convolutional neural networks (CNN) to time series classification problems. However, CNN and other traditional methods require the input data to be of the same dimension which prevents its direct application on data of various lengths and multi-channel time series with different sampling rates across channels. Long short-term memory (LSTM), another tool in the deep learning arsenal and with its design nature, is more appropriate for problems involving time series such as speech recognition and language translation. In this paper, we propose a novel model incorporating a sequence-to-sequence model that consists two LSTMs, one encoder and one decoder. The encoder LSTM accepts input time series of arbitrary lengths, extracts information from the raw data and based on which the decoder LSTM constructs fixed length sequences that can be regarded as discriminatory features. For better utilization of the raw data, we also introduce the attention mechanism into our model so that the feature generation process can peek at the raw data and focus its attention on the part of the raw data that is most relevant to the feature under construction. We call our model S2SwA, as the short for Sequence-to-Sequence with Attention. We test S2SwA on both uni-channel and multi-channel time series datasets and show that our model is competitive with the state-of-the-art in real world tasks such as human activity recognition.**

***Keywords*—time series classification, human activity recognition, deep learning, sequence-to-sequence model, attention mechanism.**

I. Introduction

Time series data mining, with its wide range of applications such as human activity recognition (HAR), has attracted a remarkable amount of interest in both academic and industry communities [1]–[5]. Inspired by the recent success of deep learning applications in computer vision, some researchers have demonstrated the effectiveness of applying CNN to time series classification problems. In those applications, the raw time series data are either encoded into images such as in [6], or chopped into equal length segments [7], and convolutions are then performed on these transformed data. The former methods reduce the problem entirely to image classification and all the relevant tools and parameter tuning techniques can be utilized. In the latter cases, usually 1D filters shared across data channels are applied along each data channel and a fusion mechanism such as voting is adopted to determine the label of the original unchopped time series. Due to the trailing fully-connected (FC) layers in CNNs, these methods require the

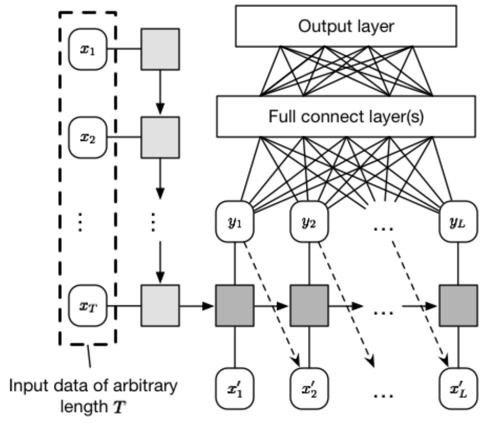

Fig. 1. Architecture of S2SwA. Encoder LSTM is in light grey and decoder LSTM is in dark grey. The network accepts time series data $x_1x_2\cdots x_T$ of arbitrary length T and converts them into features $y_1y_2\cdots y_L$ of fixed length L. The dashed arrow from y_{t-1} to x'_t indicates data replication. The decoder LSTM feeds on its output at the previous time step (With the exception of x'_1 which is set to 0 in our experiments).

input time series to be of the same length. This is prohibitive for applications such as HAR, and though researchers in computer vision incorporated spatial pyramid pooling [8] to eliminate the FC layers, it is not readily to be generalized to time series classifications.

In this paper, we explore the effectiveness of applying LSTM [9], another tool in the deep learning arsenal, to time series classification problems. With its design nature, LSTM is more appropriate for problems involving time series such as language translation [10], image caption generation [11], video representation learning [12], etc. Specifically, we propose a novel model for general time series classification in which we incorporate two LSTMs, one encoder and one decoder. In our model the encoder LSTM accepts input time series of arbitrary lengths and extracts information from the raw data and based on which the decoder LSTM constructs fixed length sequences that can be regarded as automatically extracted features. The same model is applicable to multi-channel time series data with different sampling rates across channels, in which case one can stretch each channel to a common longest length and fill the “holes” with 0’s, as the value has a natural interpretation of “no input” at the specified time step. We call our model

S2SwA, as the abbreviation for **S**equence-to-**S**equence model **w**ith **A**ttention. Fig. 1 shows the architecture of our model.

For the record, the architecture of our model bears a heavy resemblance to that used in language translations [10], there are two subtle distinctions though. In language translation the decoder LSTM outputs a sentence in the target language, thus the length of the target sequence is strongly correlated with the length of the input sequence. In our model the length of target sequences' is a design choice. The second difference lies in the training process. When training a language translation model, the decoder LSTM is fed with sentences in the target language and the difference between the generated and the supplied sentences provides error signals for network training. In contrast, our decoder feeds on its output from the previous time step ($x'_t \triangleq y_{t-1}$) in both training and test phase and the error signal comes from the softmax/hinge loss after the FC layers. In our architecture, it is also possible to incorporate hand-crafted features by inputting them into the decoder LSTM (that is, replace $x'_1 x'_2 \cdots x'_L$ with hand-crafted features), we discuss this in Sec. V. Furthermore, for better utilization of the raw data, we introduce the attention mechanism [13] into our model such that besides the encoded information from the encoder LSTM, the decoder LSTM can also peek at the raw data and focus its attention on the part of the raw data that is most relevant to the feature under construction.

The contributions of this paper are as follows: *i)* We propose a novel model incorporating encoder-decoder LSTMs with the attention mechanism for general time series classification, our model accepts input time series data of arbitrary length and outputs discriminatory feature sequences the length of which is independent of that of the inputs'. *ii)* We demonstrate the effectiveness of our model by testing it on both uni-channel [14] and multi-channel [15] time series datasets, the performance of our model is competitive with the state-of-the-art in applications such as HAR.

The rest of this paper is organized as follows. In Sec. II, we review literatures on related works. We explain our model in Sec. III, and then present our experiments and results in Sec. IV. We conclude this paper and provide an overview of future works in Sec. V.

II. Related Work

A. Long Short-term Memory (LSTM)

LSTM is an improved version of recurrent neural networks (RNN). Unlike traditional neural networks, RNN/LSTM allows loops in its network and it is these loops that help it memorize previous events such that the network can make better use of its input information. The problem with RNN is that it is unable to handle long term correlations due to fading/exploding error signals during training [9]. LSTM was invented to address this problem. Fig. 2 shows the internal structure of an LSTM cell, it has the ability to remove or add information to the internal state which is carefully regulated the input gate (i_t), the forget gate (f_t) and the output gate (o_t). The three gates determine to which extent data should be transferred. The data flow inside LSTM is modelled as follows:

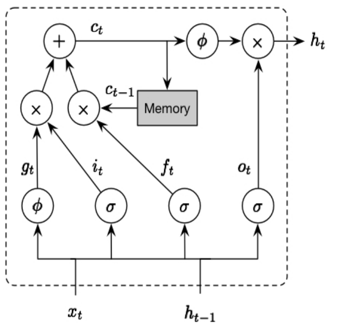

Fig. 2. Internal structure of an LSTM cell, where x_t is the input, h_{t-1} and h_t are the hidden states at time steps $t-1$ and t; i_t, f_t and o_t are activations at time t of the input gate, the forget gate and the output gate. There is a memory cell to store internal state. σ and ϕ units stand for sigmoid and tanh functions.

$$\begin{aligned}
g_t &= \phi(W_g[x_t, h_{t-1}] + b_g)\\
i_t &= \sigma(W_i[x_t, h_{t-1}] + b_i)\\
f_t &= \sigma(W_f[x_t, h_{t-1}] + b_f)\\
o_t &= \sigma(W_o[x_t, h_{t-1}] + b_o)\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t\\
h_t &= o_t \odot \phi(c_t)
\end{aligned} \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, $\phi(\cdot)$ is the tanh function, and $\odot$ indicates element-wise product. The square brackets denote elements concatenation.

B. Sequence-to-sequence Model

Sequence-to-sequence model was first explored in natural language processing [10], [16]. The model consists of two LSTMs, one acts as an encoder that squashes the information of the input sequence $x_1 x_2 \cdots x_T$ to a fixed dimensional representation v in the form of the last LSTM hidden state. Then the other LSTM plays the role of a decoder that computes the probability of $y_1 y_2 \cdots y_{T'}$ with a standard LSTM language modelling formulation in which the initial hidden state is set to v: $p(y_1 y_2 \cdots y_{T'} | x_1 x_2 \cdots x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1 y_2 \cdots y_{t-1})$ where each $p(y_t | v, y_1 y_2 \cdots y_{t-1})$ is a softmax over all the words in the vocabulary. The architecture of a sequence-to-sequence model can be referred in Fig. 1 by removing the FC layer and the output layer. In the training phase, the decoder LSTM's input $x'_1 x'_2 \cdots x'_{L'}$ is set to the target sequence $y_1 y_2 \cdots y_{L'}$ to provide supervised learning signals.

C. The Attention Mechanism

Recently, it is conjectured that squashing the information of the entire input sequence to a vector of fixed length poses bottleneck in improving the performance of the encoder-decoder architecture and a solution mechanism called the attention mechanism is proposed [13]. Concretely, by replacing each probability on the right hand side of the equation $p(y_1 y_2 \cdots y_{T'} | x_1 x_2 \cdots x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1 y_2 \cdots y_{t-1})$ with

$p(y_t|v_t, y_1y_2\cdots y_{t-1})$, each probability is conditioned on a distinct context vector v_t. The context vector v_t depends on a sequence of annotations $a_1a_2\cdots a_T$ to which an encoder maps the input sentence. Each annotation a_t contains information about the whole input sequence with a strong focus on the parts surrounding the t^{th} element of the input sequence. The context vector v_t is then computed as follows:

$$\begin{aligned} v_t &= \sum_{k=1}^{T} \alpha_{tk} a_k \\ \alpha_{tk} &= \frac{exp(e_{tk})}{\sum_{i=1}^{T} exp(e_{ti})} \\ e_{tk} &= f(h_{t-1}, a_k) \end{aligned} \tag{2}$$

where α_{tk} determines the importance of annotation a_k to the context vector v_t when generating the t^{th} element of the output sequence, h_{t-1} is the hidden state of the decoder LSTM at time $t-1$ and $f(\cdot)$ is modelled as a feedforward neural network that is jointly trained with the sequence-to-sequence model.

III. The Proposed Model

A. Challenges in Time Series Classification

Almost all time series classification researches and applications assume input data of equal length [3]. Temporal signals over a long time span are processed into shorter equal length subsequences and those that are shorter than the expected length are, if possible, padded such that the resulting input sequences to the classification engines are representative of the classes. More often than not, this preprocessing is done with great effort, and becomes more prohibitive when facing real world problems. Take HAR as an concrete example [7], [17], where the readings from sensors set on different parts of human body form the input time series data. Some actions' durations are by nature short (E.g., "open a drawer") while others are multiple times longer (E.g., "drink from a cup"), the durations of human activities also depend heavily on the habits of individuals, making it hard to align the lengths of input sequences. Furthermore, the sensors are highly likely generating readings at different sampling rates. Whether to pad the temporal series of those low frequency sensors and how to pad them can significantly affect the performance of the classification engine.

Feature engineering is the key to the success of many data mining tasks and is typically where most of the effort in a learning project goes [18]. How to manufacture good features is the second challenge that lies not only in time series classification but also in other fields of data mining tasks. Compared with other datasets such as images and voices, time series data is relatively sparse especially when one has in hand only a uni-channel temporal dataset. Crafting informative features from such a dataset demands imagination and is more of an art. To reduce the effort in feature engineering, some researchers have successfully deployed deep learning, specifically CNN, to time series classifications and have demonstrated state-of-the-art performance [6], [7]. However most CNNs are originally trained on datasets of natural images, the application of CNN in general time series classification is creative but unnatural, and since nature image data differs significantly from general time series data the benefit of transfer learning (that is, to initialize the weights of a CNN before training from another CNN that shares part of the network architecture and was trained on datasets of similar characteristics) cannot be enjoyed. Moreover, due to its size-fixed FC layers, CNN requires input data to be of the same dimension. Although researchers in computer vision incorporated spatial pyramid pooling [8] to eliminate the FC layers, it is not readily to be generalized to time series classifications and thus the formerly mentioned problem remains unaddressed.

B. Time Series Classification Using S2SwA

We propose **S2SwA** to address the aforementioned problems, the architecture of our model is depicted in Fig. 1.

S2SwA has two LSTMs at the bottom of the network, one encoder and one decoder. The encoder LSTM (shown in light grey) accepts an input sequence $x_1x_2\cdots x_T$ of arbitrary length T and extracts information from it, based on which the decoder LSTM (shown in dark grey) constructs an output sequence $y_1y_2\cdots y_L$ of fixed length L. The generated sequence is then input into the trailing hidden layers in the network. This arrangement effectively gets rid of the restriction that all input sequences should be of a uniform length, because the encoder LSTM can unroll itself along the temporal axis as many times as the input sequence lasts while the decoder LSTM always unrolls a predefined number of times to be compatible with the other layers on the top of it. The same model is applicable to multi-channel time series data with different sampling rates across channels, in which case one can stretch each channel to a common longest length and fill the "holes" with 0's, as the value has a natural interpretation of "no input" at the specified time step. Notice that we are also aware of other possible network architectures that consist of only one LSTM and manage to mitigate the uniform input lengths restriction. For example, network (A) in the left part of Fig. 3 ignores all but the last output of the LSTM which is regarded as scores for class labels. However such a specification has practically restricted the number of hidden units the LSTM can have (E.g., only two hidden units are allowed for a binary classification problem). Although this problem can be easily cured by introducing intermediate layers of bigger sizes in the LSTM, the other downside of this architecture is that it assumes information of the input sequence are lossless with the evolution of time and is completely reflected in the last time step which is usually not true empirically. In the right part of Fig. 3, architecture (B) presents a network where regardless of the input sequence's length, only the last L LSTMs' outputs $y_1y_2\cdots y_L$ are used by the trailing layers. This can actually be interpreted as a sequence-to-sequence model with the differences that *i)* there is only one LSTM that acts as both an encoder and a decoder, and ii) instead of generating the target sequence after seeing the entire input sequence, the network is forced to start constructing the target sequence after being revealed with only part of the input.

S2SwA is superior to such an architecture in that *i)* it is proved that using two LSTMs for encoding and decoding separately has outperformed the model with only one LSTM doing both at a negligible cost of increased parameters [10], and ii) architecture (B) may not have seen the most relevant information regarding the construction of the target sequence by the time it is forced to.

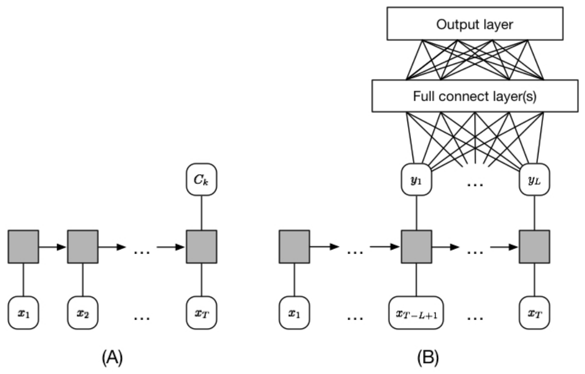

Fig. 3. Other possible network architectures that mitigate the uniform input lengths problem. (A) The output of the last LSTM is regarded as class label C_k; (B) Regardless of the input sequence's length, only the last L LSTMs' outputs are input into the trailing layers.

S2SwA requires minor feature engineering effort because of its deep architecture. It enables end-to-end training and the detail of its network structure is as follows. The model has three hidden layers in both encoder and decoder LSTMs, each hidden layer is followed by a dropout layer [19]. As is discussed in Sec. IV, the number of hidden units in each hidden layer is set as a hyper-parameter in the first experiment and fixed at $N = 300$ in the second. In both training and test phases, the decoder LSTM feeds on its own output from the previous time step ($x'_t \triangleq y_{t-1}$) except for x'_1 which is set to 0. The decoder LSTM is followed by one FC layer that contains 60 and 400 hidden ReLU units with bias terms in the first and the second experiments.

IV. Experiments

A. The UCR Archive

1) Dataset Description: The UCR classification archive [14] is the world's largest collections of time series datasets, at the time of writing this paper, the UCR archive contains more than sixty time series datasets. For the convenience of time series mining researches, all the time series datasets in the archive contain only equal length data. We test S2SwA on 12 of the datasets in the UCR archive both because these are regarded as "hard" datasets and because other deep learning method (GAF-MTF) was also tested by [6] on the very same datasets. The summarized statistics of each of the 12 datasets are given in Table I.

2) Experimental Setup: The purpose of this experiment is *not* to beat every dataset's state-of-the-art record. Rather the experiment is designed to test the effectiveness and generalization of S2SwA on simple yet non-trivial uni-channel time series classification tasks. With this in mind, we setup S2SwA in a way that depends loosely on the datasets. Concretely, as is discussed in Sec. III-B, we fix the number of hidden layers in the encoder and the decoder LSTM at 3. But we leave the input sequence length T (5 values linearly separated between 70% and 90% of the original length), the target sequence length $L \in \{15, 20\}$, number of hidden units in each hidden layer of the 2 LSTMs $N \in \{20, 30\}$, the dropout probability $p_{dropout} \in \{0, 0.2, 0.5\}$ (we use a global probability rather than specifying individual values for each layer) and the L_2 regularization parameter λ (5 values log-linearly separated in $[1.0e - 5, 1.0]$) as hyper-parameters that are tuned in grid search. Setting the input sequence length T as a hyper-parameter may seem strange because S2SwA accepts input data of arbitrary length. The rationale behind this is that to learn a deep network one needs a lot of training data, and we use a sliding window scheme as our data augmentation strategy where the window size is T and the stride is 3. When T is determined, we segment the test split in the same way and the final decision on the original sequence is determined by voting.

For data preprocessing, we normalize the training data by setting data mean to 0 and standard deviation to 1, and as is typical we used the training data mean and standard deviation to normalize the test data. Regardless of the dataset, each trial is trained for 1000 iterations with a mini-batch of 64 samples. Once the state-of-the-art is surpassed, we terminate the grid search on the corresponding dataset. We use softmax loss for all datasets and momentum gradient descent is adopted where momentum is set at 0.9. In all trials, we set the initial learning rate $\eta = 0.005$ and use the strategy to anneal the learning rate based on the learning iterations (with decay factor $\alpha = 0.1$).

3) Results: Classification error rates on the test split of each dataset together with other models are shown in Table I, the best scores for each dataset are highlighted and scores other than S2SwA's are quoted from [6]. Even though S2SwA failed to learn on datasets *Adiac* and *OliveOil* (indicated by "N/A") because of the untuned learning rate or initial weights, it has beaten quite a lot of state-of-the-arts as well. And it has acquired the second places on 4 datasets *50words*, *Beef*, *Coffee* and *OSULeaf*. The problem on *Adiac* and *OliveOil* is probably caused by the inappropriate/narrow specification of parameter search space. Specifically, we did not perform search on the initial learning rates. But since our goal is only to test the effectiveness and generalization of S2SwA and is not to beat every record in the UCR archive, we leave them as-is.

A deeper look at the generated sequences in S2SwA suggests our model indeed learns discriminatory features. Fig. 4 presents a look of the features extracted from a test batch of the dataset *Lighting2*. It is a binary classification problem, the input sequence length is $T = 455$, the number of hidden units $N = 20$ and the length of the target sequence $L = 15$. The clear overlapping of features from the same class in each plot indicates all the hidden units in the network have learned well. And the obvious separation of features from different classes proves our model generates discriminatory features for time series classification tasks.

B. Opportunity Challenge Dataset

1) Dataset Description: The Opportunity Challenge Dataset [15] is a multi-channel time series dataset. It is a subset of the Opportunity Activity Recognition database which contains naturalistic human activities of daily living (ADL) for HAR. In the data collection phase, two types of recording sessions were performed on 4 subjects: Drill sessions where the subjects performs sequentially a pre-defined set of activities and 4 or 5 ADL runs where the subjects executes a high level task with more freedom about the sequence of individual atomic activities. The sampling rate of the sensor signals is 30 per second and each recorded sample comprises of 113 real valued readings (channels) in addition to the time information. Data was manually labelled during the recording and later reviewed by at least two different persons based on the video recording. As Table II shows, there are 18 classes in the ADL recognition task where *Null* refers to the time during which no specific ADL is happening and thus includes ADLs that do not belong to any of the other 17 categories.

2) Experimental Setup: Following [7] and [17], we use only the data of the first 3 subjects. We use Drill and the first two runs of ADLs to form our training and validation splits with a random split of ratio 8 : 2, and we leave the third ADL run as out test split. We also adopt the same evaluation policy in [17], the 3 metrics used are average F-measure (AF), normalized F-measure (NF) and accuracy (AC). NF is F-measure normalized according to data sample ratio, and thus it favors majority classes. In our trials, we keep records of training and validation accuracies and we report the tuple of (AF, NF, AC) from the trial in which the highest validation accuracy is observed.

In initial data analysis, we find the dataset contains missing values caused by sensor malfunctioning or bad communication conditions. Following [7] and [17], we exclude the sensors that contain more than 30% missing values. As a result, we removed 6 sensors' readings from Subject 1's data and 3 from Subject 2's data. No such sensors are found in Subject 3's data though. And contrary to the cubic spline interpolation used in [17], we simply fill with the preceding non-missing value for the sensors with the number of missing values below the 30% threshold. After data interpolation, we normalize the data channel-by-channel by setting the mean to 0 and the standard deviation to 1.

It is necessary to segment each subject's data into subsequences because each subject's data is a long continuous sequence with interleaving class labels on each single record. We could have segmented the training data according to their ground-truth class labels, and even though this strategy will cause the training sequences to be of varied lengths (because different ADLs' durations vary), S2SwA is able to handle them. However the same strategy cannot be applied to the validation and the test splits because for fair model comparisons we are not allowed to access the ground-truth labels. As a consequence, we follow [7] and adopt the sliding window strategy. For each segmented subsequence, its ADL class label is determined by the most frequently occurred labels in the T raw labels. As for result merging, each data record is classified multiple times in different segmented subsequences (except for the beginning and the end of the original data sequence), the final label for each data record is determined by voting. In an analysis of the ADLs' durations, we noted a span of 60 data records (twice of the sampling rate) to be reasonable, thus we set sliding window size in the training phase $T_{train} = 60$ and the stride is set to 3. To illustrate that S2SwA is able to accept input data of arbitrary lengths, in the test phase we segment the original data with $T_{test} \in \{30, 60, 90\}$. Notice that this means we are testing the very same learned S2SwA network with input length different from that in the training phase, which is not possible for traditional methods.

In this experiment, we also include 4 models, SVM and 1NN from [17], CNN from [7] and a simple LSTM model for comparisons. For the SVM and the 1NN models, because the feature extraction process is non-trivial we cite their scores directly. To implement the CNN model, we follow the network architecture and parameter settings in [7][1]. And because of our specific input data extraction strategy in this experiment, we are able to implement an LSTM model that input all its LSTM's output to the trailing FC layer. The LSTM model, though incredibly simple, has been used in video recognitions [12] and proved to have improved performance. We restrict the settings of the LSTM model such as the number of hidden layers $(= 3)$ and units $(= 300)$ in LSTM to be the same as in S2SwA. Furthermore, right after their LSTM layers' output, we append exactly the same architecture from the FC layers and beyond (a hidden layer of 400 ReLU units [20] and the output layer) in the CNN to both the LSTM model and S2SwA. The input sequence lengths are $T = 30$ for the CNN model (as is the case in [7]) and $T = 60$ for the LSTM model (as is the case in S2SwA). The same evaluation policy mentioned previously is equivalent to all the comparative models.

We conducted grid search of CNN, LSTM and S2SwA on each subject's data. The search space of every hyper-parameter is shared among the three models. Concretely, the dropout probability $p_{dropout} \in [0, 0.2, 0.5, 0.9]$, the L_2 regularization parameter λ is a vector of 5 values log-linearly separated in $[1.0e-5, 1.0]$. For S2SwA, we also search for the decoder LSTM's output length $L \in [15, 20]$. In each trial, we train for 5000 iterations with a mini-batch of 128. The similar learning schedules are similar to that in experiment 1.

3) Results: During training, we keep records of training and validation accuracies every 1000 iterations, and we pick the trial with the highest validation accuracy as our final learned model for evaluation. Table III presents the accuracies on the training, validation and test splits of the comparative models from the selected trials. From the training and validation accuracies, it is clear that none of the model is overfitting.

[1]We removed the Tanh and the LRN layers in the 4^{th} section of their architecture because we find the inclusion of these layers deteriorate model performance.

TABLE I
SUMMARY STATISTICS OF DATASETS AND ERRORS OF COMPARATIVE MODELS (SCORES OTHER THAN S2SWA'S ARE QUOTED FROM [6])

Dataset	#Classes	#Train	#Test	Length	1NN-Euclidean	1NN-DTW	Fast Shapelet	BOP	SAX-VSM	GAF-MTF	S2SwA
50words	50	450	455	270	0.369	**0.242**	0.443	0.466	N/A	0.284	0.248
Adiac	37	390	391	176	0.389	0.391	0.514	0.432	0.381	**0.307**	N/A
Beef	5	30	30	470	0.467	0.467	0.447	0.433	**0.033**	0.3	0.3
Coffee	2	28	28	286	0.25	0.18	0.067	0.036	**0**	**0**	0.036
ECG200	2	100	100	96	0.12	0.23	0.227	0.14	0.14	**0.08**	0.17
FaceAll	14	560	1690	131	0.286	0.192	0.402	0.219	0.207	0.223	**0.186**
Lighting2	2	60	61	637	0.246	0.131	0.295	0.164	0.196	0.18	**0.098**
Lignting7	7	70	73	319	0.425	0.274	0.403	0.466	0.301	0.397	**0.247**
OliveOil	4	30	30	570	0.133	0.133	0.213	0.133	**0.1**	0.167	N/A
OSULeaf	6	200	242	427	0.483	0.409	0.359	0.236	**0.107**	0.446	0.207
SwedishLeaf	15	500	625	128	0.213	0.21	0.27	0.198	0.251	0.093	**0.08**
Yoga	2	300	3000	426	0.17	0.164	0.249	0.17	0.164	**0.16**	0.226

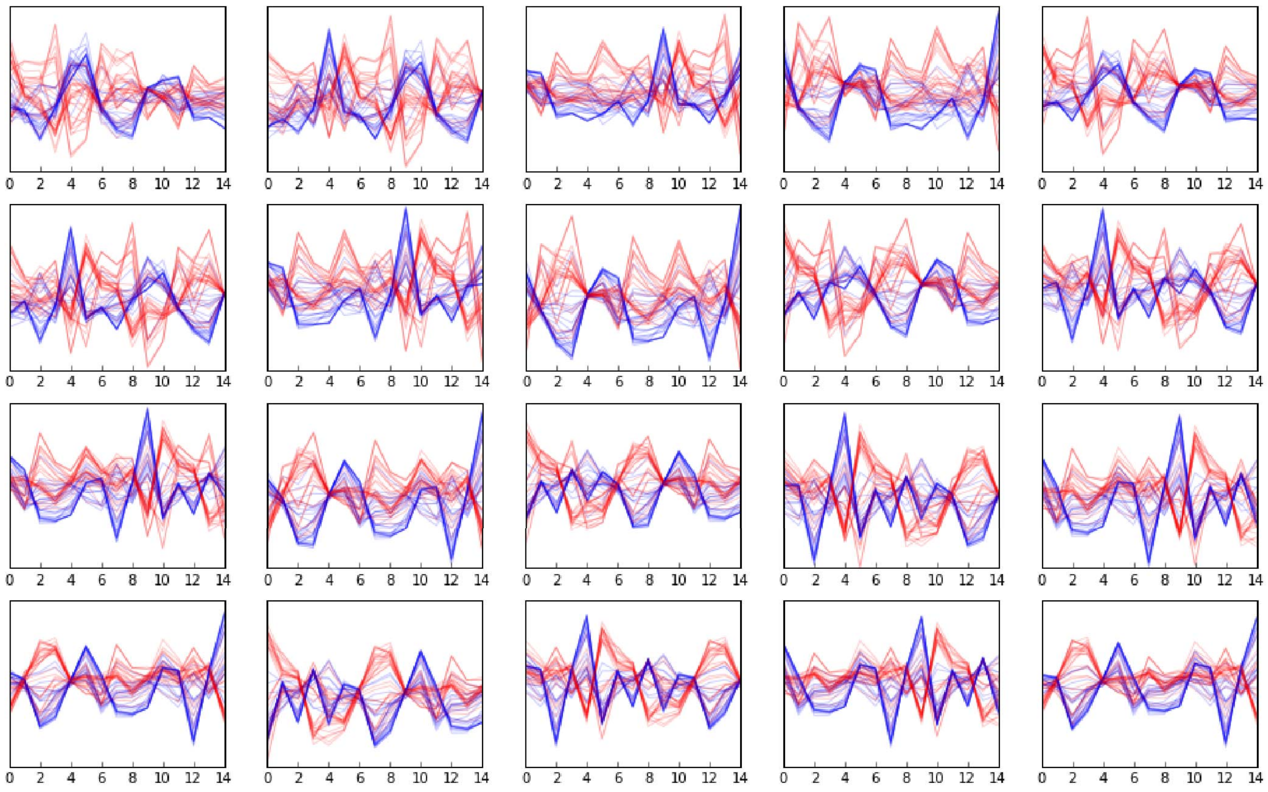

Fig. 4. Decoder LSTM's outputs from a test batch of the dataset *Lighting2* ($N = 20$, $L = 15$; label$=0$ are shown in red, label$=1$ are shown in blue)

However, the gaps between validation accuracies and test accuracies are large. This is possible because the training and the validation splits are random splits from the same set of ADL runs (Drill and the first 2 ADL runs), while the test split is from a completely different ADL run (the third run). The specific details of each run is unknown, but it is possible that different ADL runs contain actions on different objects or in a differently specified way even if the action is in the same class. We also notice that the modelling power of the LSTM model is not as strong as the other two models as its relatively low training accuracies indicate.

We summarize the scores of all the comparative models in Table IV, the best scores are highlighted in boldface. The scores for SVM and 1NN are cited from [17], and as is discussed in the description of experimental setup, we performed 3 separate tests on the same trained S2SwA model with different test input sequence lengths. The first thing to notice is that S2SwA's performance leaves those of non-deep methods far behind and we virtually did nothing on feature engineering. This result is consistent with many of the researches that advocate better representation learning ability of deep learning over shallow models. Another thing to focus on is that S2SwA constantly outperforms the LSTM model regardless of its input sequence length. Especially in the tests

TABLE II

SUMMARY OF ADL SHARES IN THE OPPORTUNITY CHALLENGE DATASET (NUMBERS ARE CITED FROM [17])

Shares of ADL averaged over the three subjects					
Null (62%)	*open door 1* (2%)	*open door 2* (2%)	*close door 1* (2%)	*close door 2* (2%)	*open fridge* (3%)
close fridge (2%)	*open dishwasher* (2%)	*close dishwasher* (2%)	*open drawer 1* (1%)	*close drawer 1* (1%)	*open drawer 2* (1%)
close drawer 2 (1%)	*open drawer 3* (2%)	*close drawer 3* (2%)	*clean table* (2%)	*drink cup* (9%)	*toggle switch* (2%)

TABLE III

MODEL ACCURACIES ON SEGMENTED SEQUENCES

Models	**Subject 1**			**Subject 2**			**Subject 3**		
	Training	**Validation**	**Test**	**Training**	**Validation**	**Test**	**Training**	**Validation**	**Test**
CNN ($T_{train} = 30$) [7]	99.37	98.66	86.10	98.79	98.17	81.67	98.93	98.08	85.03
LSTM ($T_{train} = 60$)	95.20	95.28	85.91	94.38	93.94	80.57	94.22	94.15	81.68
S2SwA ($T_{train} = 60$)	99.61	98.54	87.20	99.68	98.68	82.04	99.20	98.19	82.39

TABLE IV

SUMMARY STATISTICS OF COMPARATIVE MODELS

Models	**Subject 1**			**Subject 2**			**Subject 3**		
	AF	**NF**	**AC**	**AF**	**NF**	**AC**	**AF**	**NF**	**AC**
SVM (scores quoted from [17])	45.6	83.4	83.8	44.4	75.6	79.4	32.1	76.8	78.1
1NN (scores quoted from [17])	42.7	80.3	79.3	41.1	73.5	73.9	28.5	67.5	63.8
CNN ($T_{test} = 30$) [7]	52.9	85.2	86.2	57.0	78.0	81.8	**60.8**	**83.7**	**85.8**
LSTM ($T_{test} = 60$)	50.2	84.5	85.5	42.1	77.4	80.5	43.0	78.3	81.9
S2SwA ($T_{test} = 30$)	56.0	85.4	85.9	50.2	78.1	81.4	48.8	80.3	81.9
S2SwA ($T_{test} = 60$)	**63.9**	**86.9**	**87.4**	**52.1**	**79.6**	**82.4**	54.7	81.4	83.0
S2SwA ($T_{test} = 90$)	60.2	86.4	87.2	49.2	78.1	81.6	43.1	79.7	81.8

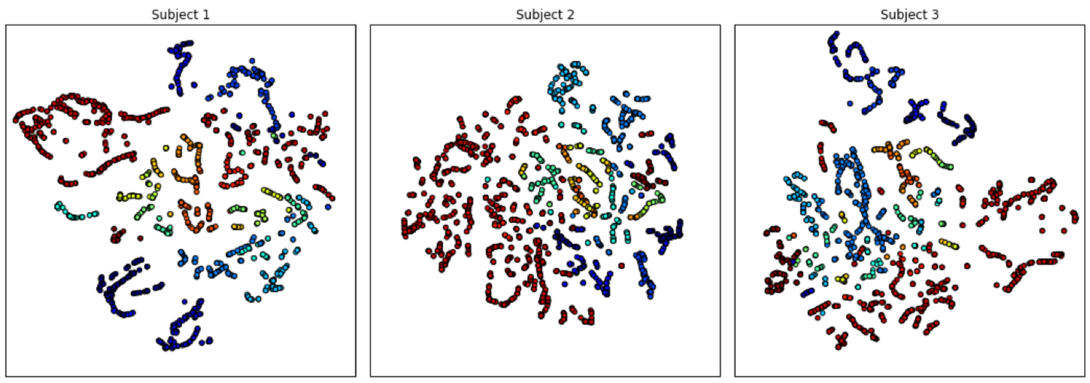

Fig. 5. Decoder LSTM's outputs projected onto 2D space using t-SNE. These features are from the test split of subject's data with $T_{test} = 60$, features of different ADLs are coloured differently. Features of the *Null* actions are not shown as this ADL's size is overwhelming and the features do not form a cluster because any actions do not belong to the rest 17 ADLs are defined as *Null*.

where $T_{test} = 60$, the advantage of S2SwA over LSTM is quite impressive. This is interesting because even with the same amount of input information and with a larger FC layer (LSTM outputs a sequence of $L = T_{train} = 60$ while S2SwA's output is of length $L = 15$ or $L = 20$, and both have the same number of hidden units $N = 300$), LSTM is still unable to catch up with S2SwA. We also notice that the performance gap between S2SwA and the CNN model is relatively small. But we are advantageous in terms of S2SwA's tolerance of variable input sequence length, which is a more realistic setting in real world time series classification tasks. Moreover, for this task the CNN model is designed with 1D filters in convolution layer that convolves along each data channel, and the estimation of relations between channels is deferred until the FC layer. This scheme has both up and down sides. On the bright side, deferring the inference of channel correlations can prevent other channels' noises from interfering the detection of local patterns. But the downside is that it also becomes a bottleneck for the model to extract more informative features that could only be discovered by considering channel correlations in an early stage. Simply resorting to 2D filters does not solve the problem, because unlike in an image pixels forming a meaningful pattern always resides in nearby area, the group of channels that contain correlations may be arranged in a way that is only partially covered by a filter or completely missed. We shall discuss this aspect of CNN again in Sec. V. Finally, as an illustration that S2SwA is indeed able to accept input time series of arbitrary length, we conducted tests in which the test sequences' length is different from that used in the training phase (the rows where $T_{test} = 30$ and 90). The scores dropped when T_{test} is shrunken as is expected, because the long term correlations detected in the training phase may not present in shorter input sequences. However, the scores dropped when T_{test} is extended as well. This is a little surprising, but since the extent of drops vary significantly on each subject, we suspect this is due to data "purity". To be concrete, when segmenting the original input data, we determine the class label of each subsequence by majority voting. The longer a subsequence becomes, the more likely it becomes that the subsequence contains multiple raw labels, hence, lowering its "purity". When we assign a single class label to this subsequence, we are generating more conflicting results against the ground-truths. Furthermore, since subsequences are becoming longer, we have less segmented subsequence, in the result merging phase, we have less voting to correct these conflicted decisions.

We use t-SNE to project the decoder LSTM's output onto a 2D space in Fig. 5 ($T_{test} = 60$). Features of different ADL class labels are coloured differently and we have excluded the *Null* class. The shape and the separation status of each subject's feature plot is consistent with the scores in Table IV. For example, subject 1's scores are the highest and the corresponding feature plot shows apparent clusters that are separated clearly. On the other hand, though subject 2's features form clusters as well, the clusters are in a cluttered form and is harder to discern.

V. Conclusion and Future Works

In this paper, we present a novel model S2SwA for general time series classification. It contains a sequence-to-sequence model that accepts input series of arbitrary length. We incorporate the attention mechanism into S2SwA for better utilization of its input data while generating features. To prove the effectiveness of our proposed model, we conducted experiments on both uni-channel and multi-channel time series datasets, and demonstrated our model is competitive with the state-of-the-art in applications such as HAR.

Sequence-to-sequence models in language tranlations have inputs into its decoder LSTM in the training phase. In the case of time series classification, we wonder whether it is possible to utilize this interface such that existing hand-crafted features can be integrated and if this is a viable method to boost the model's performance. As another piece of our future works, we shall further explore the restrictions of CNNs on the tasks of time series classification.

References

[1] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 947–956.

[2] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 289–297.

[3] B. Hu, Y. Chen, and E. Keogh, "Time series classification under more realistic assumptions," 2013.

[4] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, Oct. 2003.

[5] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle, "Indexing large human-motion databases," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 780–791.

[6] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAi, 2015.

[7] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 3995–4001.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112.

[11] A. Karpathy and F. Li, "Deep visual-semantic alignments for generating image descriptions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 3128–3137.

[12] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *CoRR*, vol. abs/1411.4389, 2014.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[14] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[15] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. del R. Millán, "Collecting complex activity datasets in highly rich networked sensor environments," in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, Jun. 2010, pp. 233 –240.

[16] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[17] H. Cao, M. N. Nguyen, C. Phua, S. Krishnaswamy, and X.-L. Li, "An integrated framework for human activity recognition," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 621–622.

[18] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012.

[19] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.