SpanNet: Pre-trained CNN and Recurrent CNN for time series classification

Gaurav Anand
University of Massachusetts, Amherst
Amherst, MA 01003
ganand@umass.edu

Abstract

In this paper, we present SpanNet – a pre-trained neural network whose representations can be utilized for the task of time-series classification. As opposed to previous recurrent encoder-decoder based methods, SpanNet builds on convolutional neural networks (CNN) and recurrent convolutional neural networks (RCNN) architectures considering the fact that convolutional operations are quick when performed on modern GPU based hardware. We have conducted our experiments on a plethora of time-series datasets arising from 50 different domains. Results suggest that the explored method surpasses the performance of existing state-of-the-art DTW-based-classifier and RNN models in 11 out of 30 test datasets while the existing state-of-the-art RNN achieved 8 out of 30. Finally, t-SNE visualization of SpanNet embeddings form well-separated clusters, which makes it evident that proposed method captures the semantics of encoded time-series from different classes of time-series data.

1. Introduction

With the digitization of manufacturing and advent of "Internet of things (IoT)" unlabeled time series data from sensors has become abundant. And also the need for real-time analysis of these time series. Still in many domains, getting labeled data is costly and sometimes implausible although unlabeled data may be readily available.

Deep Learning models have shown remarkable results in Speech Recognition, Computer Vision, and NLP. The dominant approach for sequential tasks like time series analysis involves multilayered recurrent neural networks which have produced state-of-the-art results for various time series problems (Hermans and Schrauwen, 2013 [**?**]; Malhotra et al., 2015 [**?**]). Malhotra et al., 2017 [**?**] have shown that an RNN autoencoder trained on some time series can be used to get representations which can be easily classified for other time series that are very different from the ones on which it was trained.

In the present work, we explore convolutional models for sequence task such as time series classification. We propose two seq2seq autoencoder models, Recurrent CNN autoencoder, and CNN autoencoder which will be trained in an unsupervised manner to extract embeddings from time series of variable-length from diverse domains and show that they can be useful representation for various tasks. We evaluate our models by comparing the classification error with the results of Malhotra et al, 2017 [**?**]. We visualize the encoder embeddings using t-SNE for various time series from different domains which weren't part of the training. The visualization confirms that our model captures important characteristics of time series data.

Compared to recurrent models, convolutional models can be parallelized to utilize GPU hardware and computational graph optimizations are possible on it because the graph is fixed and independent of input length. Convolutions create representations for fixed size contexts and hence allows one to precisely model and control the maximum length of dependencies. "Hierarchical structure provides a shorter path to capture long-range dependencies compared to the chain structure modeled by recurrent networks, e.g. we can obtain a feature representation capturing relationships within a window of n words by applying only $\mathcal{O}\left(\frac{n}{k}\right)$ convolutional operations for kernels of width k, compared to a linear number $\mathcal{O}(n)$ for recurrent neural networks. Inputs to a convolutional network are fed through a constant number of kernels and non-linearities, whereas recurrent networks apply up to n operations and non-linearities to the first word and only a single set of operations to the last word. Fixing the number of non-linearities applied to the inputs also eases learning", Gehring et al. 2017 [**?**].

2. Related Work

Malhotra et al., 2017 [**?**] have shown that an RNN autoencoder can be trained on diverse sets of time series, it can then be used as a generic off-the-shelf feature extractor for time series classification, to obtain fixed-dimensional representations or embeddings.

Fixed-dimensional vector representations or embeddings of variable-length sentences have been shown to be useful for a variety of document classification tasks. Deep recurrent neural networks (RNNs) has been state-of-the-art technique to perform hierarchical processing of time series and other sequential data.

Convolutional Neural Networks have been successfully shown to work for various tasks like sentence classification (Kim, 2014) [**?**] and sequence to sequence learning (Gehring et al 2017 [**?**]). As pointed out above the benefits of convolutional models and their recent adaption for various seq2seq tasks, it is inspiring to try them out for time series classification. We train a seq2seq autoencoder in an unsupervised way on various time series as opposed to domain-specific autoencoders, which provide useful representations for time series classification but may be prone to overfitting given few training instances.

3. Approach

Previous works involving convolutional network for sequtial tasks serve as a motivation to explore multilayered convolutional autoencoders that process and transforms a univariate time series to a fixed-dimensional vector representation, and once trained on diverse enough time series, serve as a generic feature extractor for time series.

3.1. Model 1: Convolutional Autoencoder

A Convolutional autoencoder which takes time series of length 512 (enforcing the same limit enforced by Malhotra et al. [**?**]). Time series shorter than that are duplicated as many time as needed to achieve that get length ≥ 512 and the extra length is clipped. Thus, width of time series is fixed to length = 512. The best model parameters that we found have 32 filters in first layer i.e. $F_1 = 32$, and so on $F_2 = 16$, $F_3 = 4$, kernel sizes are 20, 11, 8 for L1, L2, and L3, and strides of 2, 4, 2 respectively and zero padding in all layers. See figure 1. Batch Normalization and Dropout were tried for autoencoder but the best accuracy was achieved without them.

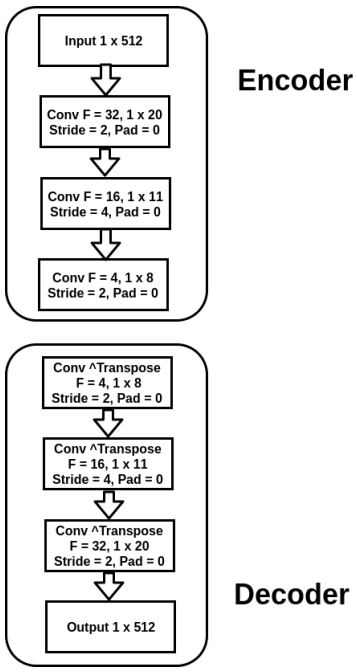


Figure 1: Convolutional Autoencoder.

3.2. Model 2: Convolutional LSTM Autoencoder

Although fully connected LSTM models are powerful for handling temporal correlation, there is too much redundancy in their structure for spatial information. Convolutional LSTMs have been shown to capture spatiotemporal sequences efficiently with far fewer parameters. Hence, they require far fewer data and don't overfit.

The key equations of ConvLSTM (Shiand et al. [**?**]) are shown in (1) below, where '$*$' denotes the convolution operator and '$\circ$', as before, denotes the Hadamard product:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
\tag{1}
$$

The best model parameters for Recurrent Convolutional Neural Network have convolutional input length = 32, time series is chunked into length of size 32. 36 filters in first layer i.e. $F_1 = 36$, and so on $F_2 = 12$, $F_3 = 4$, kernel sizes are 21, 11, 7 for L1, L2, and L3, and strides is 1 and padding is done such that input length is same as output length. See figure 2. Time series is chunked into input size length to match the 1-D convolution length of hidden state.

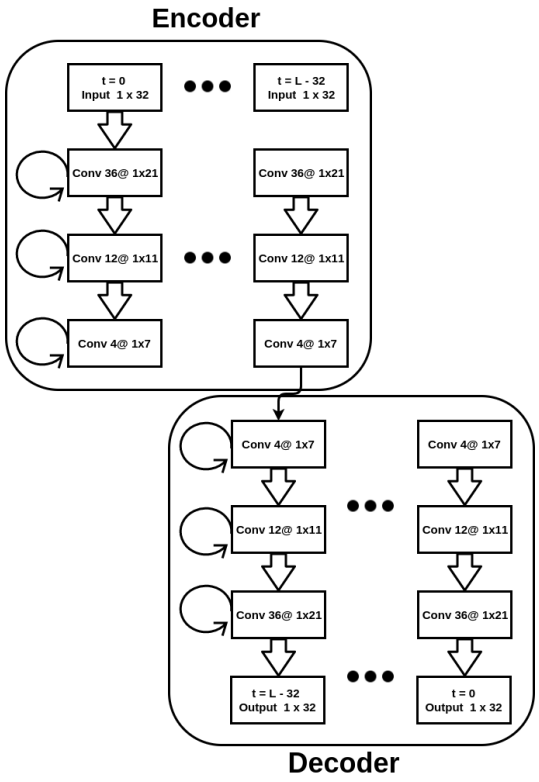


Figure 2: ConvLSTM RNN Autoencoder

3.3. SVM RBF Kernel

After training the weights of autoencoders are fixed and we learn a classifier from the embeddings of the encoder for each test set. We learn two Support Vector Machine (SVM) with radial basis function (RBF) kernel for both the models separately on training data of new set of time series data. We report accuracy on the test set.

4. Experimental Setup

4.1. Dataset

We use UCR Time Series Classification Archive [**?**], which has a large number of univariate time series datasets from many domains.

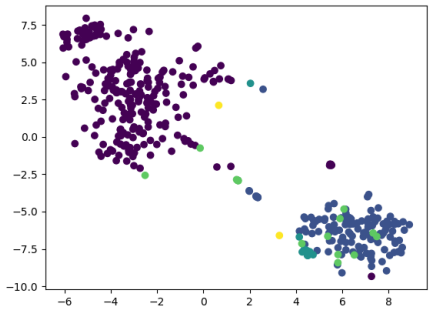
4.2. Training

We implemented and trained our model using pytorch framework. It takes about an hour for ConvLSTM autoencoder and few minutes for pure convolutional autoencoder. We use mean squared error for autoencoder reconstruction loss.
All time series were normalized to have unit variance and zero mean individually. The output of ConvLSTM decoder was produced in reverse order because it is easier to train and learn it that way. In case of Conv LSTM RNN Autoencoder, only one type of time-series was given in a batch and loss was normalized w.r.t. the length of time series so that we can decipher the validation loss i.e. we expect it to decrease consistently after the loss is normalized. We use held out validation set for training of both autoencoders to check overfitting, under-fitting issues. After autoencoder, we train SVMs (RBF Kernel) on embeddings from encoder for each time series. And we also show the t-SNE visualizations with suitable perplexity for each time series.

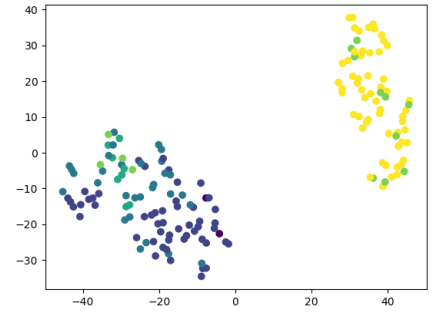
4.3. Qualitative Results

We visualize embedding from the autoencoders on different time series. The t-SNE visualizations confirm the ability of the autoencoders to yield meaningful embeddings. See t-SNE visualizations below.

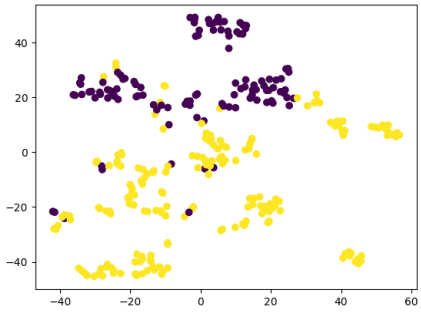
4.4. Visualization of embeddings using t-SNE



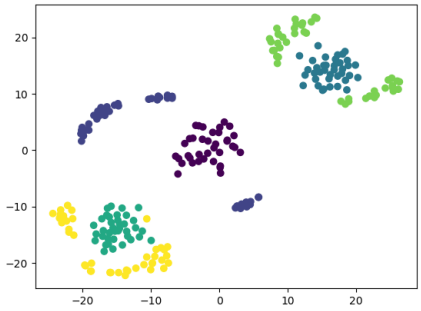
(a) ECG 5000 (C=4)



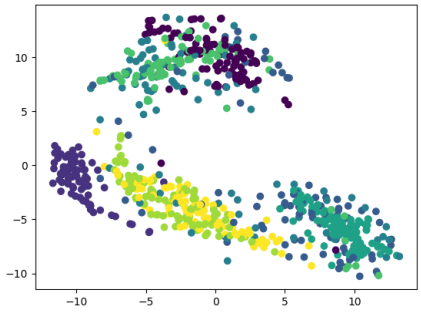
(b) ProximalPhalanxTW (C=7)



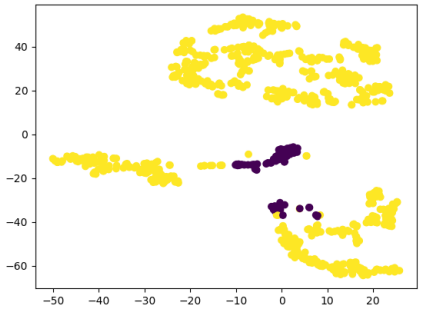
(c) Strawberry (C=2)



(d) Synthetic Control (C=6)



(e) uWaveGestureLibrary Z (C=8)



(f) Wafer (C=2)

4.5. Quantitative Results

We compare the classification error of our model with the results from Timenet model [**?**] and Dynamic Time Warping methods. Our convolutional model gives the best results.

The classification error rates for TN-C, SAE-C, and DTW-C are reported in Table 1. The error rates for DTW-C are taken from .

Dataset	T	Conv-AE	Conv-LSTM-RNN	DTW-C	SAE-C	TN-C
Adiac	176	0.4878	0.9772	0.391	0.435	**0.322**
ChlorineConcentration	166	0.3786	0.4698	0.35	0.277	**0.269**
Cricket_X	300	0.4487	0.9230	**0.236**	0.341	0.300
Cricket_Y*	300	0.5421	0.8289	**0.197**	0.397	0.338
Cricket_Z*	300	0.5342	0.8611	**0.180**	0.305	0.308
DistalPhalanxOutlineAgeGroup*	80	0.25	0.2812	0.228	**0.160**	0.223
DistalPhalanxOutlineCorrect*	80	0.2115	0.2391	0.232	**0.187**	0.188
DistalPhalanxTW*	80	0.4230	0.3333	0.272	0.243	**0.208**
ECG5000	140	**0.0490**	0.0937	0.075	0.066	0.069
ECGFiveDays*	136	0.2857	**0.0**	0.203	0.063	0.074
ElectricDevices	96	**0.2571**	0.4716	0.376	0.335	0.267
FordA	500	0.3571	0.4179	0.341	0.284	**0.219**
FordB	500	0.4011	0.4808	0.414	0.405	**0.263**
MedicalImages	99	0.4487	0.4342	0.253	**0.247**	0.250
MiddlePhalanxOutlineAgeGroup*	80	0.4062	0.3928	0.253	0.348	**0.210**
MiddlePhalanxOutlineCorrect*	80	**0.1489**	0.2	0.318	0.307	0.270
MiddlePhalanxTW*	80	0.4705	0.5294	0.419	0.381	**0.363**
PhalangesOutlinesCorrect	80	**0.1656**	0.3145	0.239	0.228	0.207
ProximalPhalanxOutlineAgeGroup*	80	**0.1182**	0.2560	0.215	0.137	0.146
ProximalPhalanxOutlineCorrect*	80	**0.1538**	0.3008	0.210	0.179	0.175
ProximalPhalanxTW*	80	**0.0952**	0.3142	0.263	0.188	0.195
Strawberry	235	**0.0597**	0.1973	0.062	0.070	0.062
Swedish Leaf	128	0.2110	0.7589	0.157	**0.099**	0.102
Two Patterns	128	0.1675	0.5929	0.002	0.001	**0.000**
Wafer	152	**0.0**	0.0773	0.005	0.006	0.005
Yoga	426	0.3103	0.5	**0.155**	0.174	0.160
Synthetic Control	60	**0.0**	0.7460	0.017	0.017	0.013
uWaveGestureLibrary_X	315	0.2554	0.3371	0.227	**0.211**	0.214
uWaveGestureLibrary_Y*	315	**0.2470**	0.4	0.301	0.291	0.311
uWaveGestureLibrary_Z*	315	0.3103	0.4186	0.322	**0.280**	0.281
Win or Tie compared to DTW-C		**11/30**	1/30	4/30	6/30	8/30

Table 1: Classification error rates. Conv-AE is our purely convolutional autoencoder. Conv-LSTM-RNN is also our model. TN-C is timenet, SAE-C is Specific to dataset autoencoder, DTW is Dynamic Time Warping

5. Conclusion and Future Work

This paper explored the use of CNNs and RCNNs based model to learn representations from time series data as opposed to previous RNN based models. Experimental results empirically suggest that the explored approach provides an alternative in addition to previous methods. As the explored approach have convolutional operations it yields predictions relatively faster than recurrent models. Though in our experiments, the explored approach outperforms in 11 out of 30 test datasets, we believe that this performance can be further improved by exploring more complex CNN based models. For instance, it will be interesting to use convolutional filters of different sizes in a convolutional layer as they can more easily encapsulate time-series patterns of

different scale and swatch. We also tried ConvLSTM network instead of 'vanilla' LSTM to capture spatiotemporal sequence patterns in time series data. The result of ConvLSTM RNN model can definitely be improved in future. Like, having different sizes in a convolutional layer, and overlapping input windows. And since it is time series one can try autoregressive model rather than an autoencoder.