Hybrid Neural Networks for Learning the Trend in Time Series

Tao Lin*, **Tian Guo***, **Karl Aberer**
School of Computer and Communication Sciences
Ecole polytechnique federale de Lausanne
Lausanne, Switzerland
{tao.lin, tian.guo, karl.aberer}@epfl.ch

Abstract

Trend of time series characterizes the intermediate upward and downward behaviour of time series. Learning and forecasting the trend in time series data play an important role in many real applications, ranging from resource allocation in data centers, load schedule in smart grid, and so on. Inspired by the recent successes of neural networks, in this paper we propose TreNet, a novel end-to-end hybrid neural network to learn local and global contextual features for predicting the trend of time series. TreNet leverages convolutional neural networks (CNNs) to extract salient features from local raw data of time series. Meanwhile, considering the long-range dependency existing in the sequence of historical trends, TreNet uses a long-short term memory recurrent neural network (LSTM) to capture such dependency. Then, a feature fusion layer is to learn joint representation for predicting the trend. TreNet demonstrates its effectiveness by outperforming CNN, LSTM, the cascade of CNN and LSTM, Hidden Markov Model based method and various kernel based baselines on real datasets.

1 Introduction

Time series, which is a sequence of data points in time order, is being generated in a wide spectrum of domains, such as medical and biological experimental observations, daily fluctuation of stock markets, power consumption records, performance monitoring of data centres, and so on. In many applications, users are interested in understanding the evolving trend in time series and forecasting the trend, since the conventional prediction on specific data points could deliver very little information about the semantics and dynamics of the underlying process generating the time series. For instance, time series in Figure 1 are from the power consumption dataset [1]. Figure 1(a) shows some raw data points of time series. Though point A and B have approximately the same value, the underlying system is likely to be in two different states when it outputs A and B, because A is in an upward trend while B is in a downward trend [Wang *et al.*, 2011]. On the other hand, even when two points with the similar value are both in the upward trend, e.g, point A and C, the different slopes and durations of the trends where point A and C locate, could also indicate different states of the underlying process.

Particularly, in this paper we are interested in the trend of time series which measures the intermediate behaviour, i.e. upward or downward pattern of time series. It is characterized by the slope and duration [Wang *et al.*, 2011]. For instance, Figure 1(b) and (c) show one time series and the associated trend evolution over the time series. Given such information, we aim to predict the duration and slope of the subsequent trend. Learning and forecasting trends are quite useful in a wide range of applications. For instance, in the smart energy domain, knowing the predictive trend of power consumption time series enables energy providers to schedule power supply and maximize energy utilization [Zhao and Magoulès, 2012]. in the stock market, due to its high volatility and noisy environment. In reality predicting stock price trends is preferred over the prediction of the stock market absolute values [Atsalakis and Valavanis, 2009]. Predicting the trend of stock price time series provides the insight into the intermediate behaviours of economic actors [Atsalakis and Valavanis, 2009].

Meanwhile, in recent years neural networks have shown the dramatical power in a wide spectrum of domains, e.g., natural language processing, computer vision, speech recognition, time series analysis, etc [Wang *et al.*, 2016b; Sutskever *et al.*, 2014; Yang *et al.*, 2015; Lipton *et al.*, 2015; Guo *et al.*, 2016]. For time series data, two mainstream architectures, convolutional neural network (CNN) and recurrent neural network (RNN), have been exploited in different tasks, e.g., RNN in time series classification [Lipton *et al.*, 2015] and CNN in activity recognition and snippet learning [Liu *et al.*, 2015; Yang *et al.*, 2015]. RNN is powerful in discovering the dependency in sequence data [Chung *et al.*, 2014] and particularly the Long Short-Term Memory (LSTM) RNN works well on sequence data with long-term dependencies [Chung *et al.*, 2014; Hochreiter and Schmidhuber, 1997] due to the internal memory mechanism. CNN excels in extracting effective representation of local salience from raw data of time series by enforcing a local connectivity

*These two authors contributed equally.

[1] https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

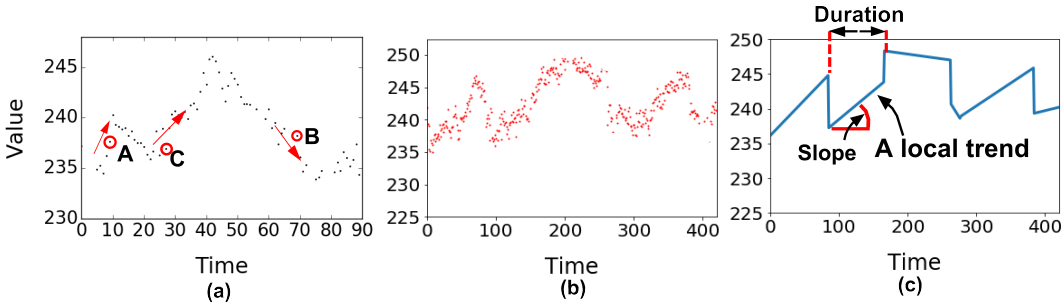


Figure 1: (a) Underlying information in time series. (b) Time series. (c) Sequence of trends associated with the time series.

between neurons [Yang *et al.*, 2015; Hammerla *et al.*, 2016; Wang and Oates, 2015].

In this paper, we focus on learning and forecasting the trends in time series via neural networks. This involves learning different aspects of the data. On the one hand, the trend variation of time series is a sequence of historical trends carrying the long-term contextual information of time series and naturally affects the evolution of the following trend. On the other hand, the recent raw data points of time series [Wang *et al.*, 2011], which represent the local behaviour of time series, affects the evolving of the following trend as well and have particular predictive power for abruptly changing trends. For instance, in Figure 2(b), trend 1, 2 and 3 present a continuous upward pattern corresponding to the time series before the prediction time instant. Then when we aim at predicting the subsequent trend of time series, the previous three successive upward trends outline a probable increasing trend afterwards. However, the local data points around the end of the third trend as is shown in Figure 2(a), e.g., data points in the red circle, indicate that time series could stabilize and even decrease. The true data after the third trend indeed present a decreasing trend indicated by the blue dotted segment. In this case, the subsequent trend has more dependency on the local data points. Therefore, it is highly desired to develop a systematic way to model such hidden and complementary dependencies in time series.

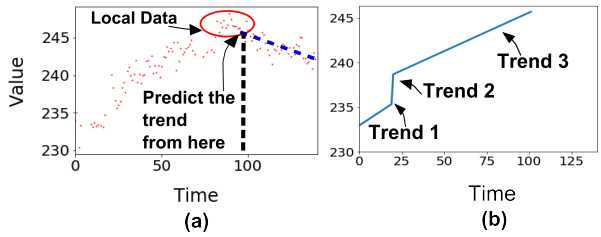


Figure 2: (a) Trend prediction on time series. (b) Associated sequence of trends.

To this end, we propose an end-to-end hybrid neural network, referred to as TreNet. In particular, it consists of an LSTM recurrent neural network to capture the long dependency in historical trends, a convolutional neural network to extract local features from local raw data of time series, and a feature fusion layer to learn joint representation to take advantage of both features drawn from CNN and LSTM. Such joint representation is used for the trend forecasting. The experimental analysis on real datasets demonstrates that TreNet outperforms CNN, RNN, the cascade of CNN and RNN, and a variety of baselines in term of trend prediction accuracy.

The rest of the paper is organized as follows. Section 2 presents related work. In Section 3, we present the proposed TreNet. Section 4 reports experimental results and the paper is concluded in Section 5.

2 Related Work

Traditional learning approaches over trends of time series mainly make use of Hidden Markov Models (HMMs) [Wang *et al.*, 2011; Matsubara *et al.*, 2014]. HMMs maintain short-term state dependences, i.e., the memoryless Markov property and predefined number of states, which requires significant task-specific knowledge. RNNs can capture long-term dependencies in sequence data. Previous time series segmentation approaches [Wang *et al.*, 2011; Yuan, 2015] focus on achieving a meaningful segmentation, rather than modeling the relation in segments and therefore are not suitable for forecasting trends. Multi-step ahead prediction is another way to realize trend prediction by fitting the predicted values to estimate the trend [Chang *et al.*, 2012]. However, multi-step ahead prediction of time series is non-trivial [Venkatraman *et al.*, 2015] and suffers from the accumulative prediction errors [Taieb and Atiya, 2016; Bao *et al.*, 2014]. In this paper, we concentrate on directly learning trends through neural networks.

RNNs have recently shown promising results in a variety of applications, especially when there exist sequential dependencies in data [Chung *et al.*, 2014; Sutskever *et al.*, 2014]. Long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997; Chung *et al.*, 2014], a class of recurrent neural networks with sophisticated recurrent hidden and gated units, are particularly successful and popular due to its ability to learn hidden long-term sequential dependencies. [Lipton *et al.*, 2015] uses LSTMs to recognize patterns in multivariate time series, especially for multi-label classification of diagnoses. [Malhotra *et al.*, 2015] evaluate the ability of LSTMs to detect anomalies in ECG time series.

CNNs are often used to learn effective representation of local salience from raw data [Vinyals *et al.*, 2015; Donahue *et al.*, 2015]. [Hammerla *et al.*, 2016; Yang *et al.*, 2015] make use of CNNs to extract features from raw time series data for activity/action recognition. [Liu *et al.*, 2015] focuses on

the prediction of periodical time series values by using CNN and embedding time series with neighbors in the temporal domain. Our proposed TreNet will combine the strengths of both LSTM and CNN and form a novel and unified neural network architecture for trend forecasting.

Hybrid neural networks, which combines the strengths of various neural networks, are receiving increasing interest in the computer vision domain, such as image captioning [Mao *et al.*, 2014; Vinyals *et al.*, 2015; Donahue *et al.*, 2015], image classification [Wang *et al.*, 2016a], action recognition [Ballas *et al.*, 2015; Donahue *et al.*, 2015] and so on. But efficient exploitation of such hybrid architectures has not been well studied for time series data, especially the trend forecasting problem. [Ballas *et al.*, 2015] utilizes CNNs over images in a cascade of RNNs in order to capture the temporal features for classification. [Bashivan *et al.*, 2015] transforms EEG data into a sequence of topology-preserving multi-spectral images and then trains a cascaded convolutional-recurrent network over such images for EEG classification. [Wang *et al.*, 2016a; Mao *et al.*, 2014] propose the CNN-RNN framework to learn a shared representation for image captioning and classification problems.

3 Hybrid neural networks for Learning the Trend

In this section, we provide the formal definition of the trend learning and forecasting problem. Then, we present the proposed TreNet.

3.1 Problem Formulation

We define time series as a sequence of data points $\mathcal{X} = \{x_1, \ldots, x_T\}$, where each data point x_t is real-valued and subscript t represents the time instant. The historical *trend sequence* of $\mathcal{X}$ is a series of historical trends over $\mathcal{X}$, denoted by $\mathcal{T} = \{\langle \ell_k, s_k \rangle\}$. Each element of $\mathcal{T}$, e.g., $\langle \ell_k, s_k \rangle$ describes a linear function over a certain subsequence (or segment) of $\mathcal{X}$ and corresponds to a trend in $\mathcal{X}$. ℓ_k and s_k respectively represent the duration and slope of trend k. ℓ_k is measured in terms of the time range covered by trend k. Both ℓ_k and s_k are continuous values. Trends in $\mathcal{T}$ are time ordered and non-overlapping. The durations of all the trends in $\mathcal{T}$ address $\sum_k \ell_k = T$.

Meanwhile, as we discussed in Section 1, local raw data of time series affects the evolving of trend and thus we define the *local data w.r.t.* each historical trend in $\mathcal{T}$ as a set of data points of size w. Formally, it is denoted by $\mathcal{L} = \{\langle x_{t_k - w}, \ldots, x_{t_k} \rangle\}$, where t_k is the ending time of trend k in $\mathcal{T}$. w is empirically selected, which will be discussed in Section 4.

Then, we aim to propose a neural network based approach to learn a function $f(\mathcal{T}, \mathcal{L})$ to predict the subsequent trend $\langle \hat{l}, \hat{s} \rangle$. In this paper, we focus on univariate time series. The following proposed method can be naturally generalized to multivariate time series as well by augmenting the training data.

3.2 TreNet

In this part, we first present the overview of the proposed hybrid architecture TreNet for the trend forecasting. Then we will detail each component of TreNet.

Overview

The idea of TreNet is to combine CNN and LSTM to utilize their representation abilities on different aspects of data (i.e., $\mathcal{L}$ and $\mathcal{T}$) and to learn a joint feature used for the trend prediction.

Technically, TreNet is designed to learn a predictive function $\langle \hat{l}, \hat{s} \rangle = f(R(\mathcal{T}), C(\mathcal{L}))$. $R(\mathcal{T})$ is derived by training the LSTM over the sequence $\mathcal{T}$ to capture the dependency in historical trend evolving, while $C(\mathcal{L})$ corresponds to local features extracted by CNN from sets of local data in $\mathcal{L}$. The long-term and local features respectively captured by LSTM and CNN, i.e., $R(\mathcal{T})$ and $C(\mathcal{L})$ convey complementary information pertaining to the trend varying. Then, the feature fusion layer merges the features for forecasting the subsequent trend. Finally, the trend prediction is realized by the function $f(\_, \_)$, which corresponds to the feature fusion and output layers as is shown in Figure 3.

Learning the dependency in the historical trend sequence

During the training phase the duration ℓ_k and slope s_k of each trend k in sequence $\mathcal{T}$ are fed into the LSTM layer of TreNet. Each j-th neuron in the LSTM layer maintains a memory c_k^j at step k. The output h_k^j or the activation of this neuron is then expressed as: $h_k^j = o_k^j tanh(c_k^j)$, where o_k^j is an output gate of a LSTM neuron [Hochreiter and Schmidhuber, 1997; Chung *et al.*, 2014]. At each step k, the hidden activation $\boldsymbol{h}_k$ is the output to the feature fusion layer.

o_k^j is calculated by

$$o_k^j = \sigma(\boldsymbol{W_o}[\ell_k \; s_k] + \boldsymbol{U_o h_{k-1}} + \boldsymbol{V_o c_k})^j \tag{1}$$

where $[\ell_k \; s_k]$ is the concatenation of the duration and slope of the trend k, $\boldsymbol{h_{k-1}}$ and $\boldsymbol{c_k}$ are the vectorization of the activations of $\{h_{k-1}^j\}$ and $\{c_k^j\}$, and σ is a logistic sigmoid function. Then, the memory cell c_k^j is updated through partially forgetting the existing memory and adding a new memory content $\tilde{c}_k^j$:

$$c_k^j = f_k^j c_{k-1}^j + i_k^j \tilde{c}_k^j \;, \;\; \tilde{c}_k^j = tanh(\boldsymbol{W_c}[\ell_k \; s_k] + \boldsymbol{U_c h_{k-1}})^j \tag{2}$$

The extent to which the existing memory is forgotten is modulated by a forget gate f_k^j, and the degree to which the new memory content is added to the memory cell is modulated by an input gate i_k^j. Then, such gates are computed by

$$f_k^j = \sigma(\boldsymbol{W_f}[\ell_k \; s_k] + \boldsymbol{U_f h_{k-1}} + \boldsymbol{V_f c_{k-1}})^j \tag{3}$$

$$i_k^j = \sigma(\boldsymbol{W_i}[\ell_k \; s_k] + \boldsymbol{U_i h_{k-1}} + \boldsymbol{V_i c_{k-1}})^j \tag{4}$$

V_f and V_i are diagonal matrices.

Learning local features from raw data of time series

When the k-th trend in $\mathcal{T}$ is fed to LSTM, the corresponding local raw time series data points $\langle x_{t_k - w}, \ldots, x_{t_k} \rangle$ in $\mathcal{L}$

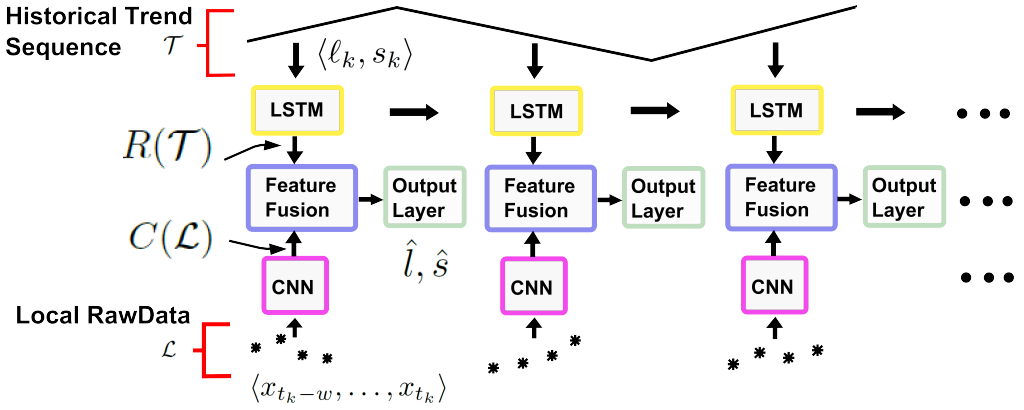


Figure 3: Illustration of the hybrid architecture of TreNet. (best viewed in colour)

is input to the CNN part of TreNet. CNN consists of H stacked layers of 1-d convolutional, activation and pooling operations. Each layer has a specified number of filters of a specified filter size. Each filter on a layer sweeps through the entire set of data points to exact local features. The output of CNN in TreNet is the concatenation of max-pooling outputs on the last layer H [Donahue *et al.*, 2015].

Feature fusion and output layers

The feature fusion layer combines the output representations from LSTM and CNN, i.e., $R(\mathcal{T})$ and $C(\mathcal{L})$, to form a joint feature. Then, such a joint feature is fed to the output layer to provide the trend prediction. Particularly, we first map $R(\mathcal{T})$ and $C(\mathcal{L})$ to the same feature space and then add them together to obtain the activation of the feature fusion layer [Mao *et al.*, 2014]. The output layer is a fully-connect layer following the feature fusion layer. Mathematically, the prediction of TreNet is expressed as:

$$\begin{aligned}\langle \hat{l}, \hat{s} \rangle &= f(R(\mathcal{T}),\ C(\mathcal{L})) \\ &= \boldsymbol{W^o} \cdot \underbrace{\boldsymbol{\phi}(\boldsymbol{W^r} \cdot R(\mathcal{T}) + \boldsymbol{W^c} \cdot C(\mathcal{L}))}_{feature\ fusion} + \boldsymbol{b^o} \end{aligned} \quad (5)$$

where $\boldsymbol{\phi}(\cdot)$ is element-wise leaky ReLU activation function and $+$ here denotes the element-wise addition. $\boldsymbol{W^o}$ and $\boldsymbol{b^o}$ are the weights and bias of the output layer.

To train TreNet, we adopt the squared error function plus a regularization term as:

$$J(\boldsymbol{W}, \boldsymbol{b}; \mathcal{T}, \mathcal{X}) = \frac{1}{|\mathcal{T}|} \sum_{k=1}^{|\mathcal{T}|} \left[(\hat{l}_k - l_k)^2 + (\hat{s}_k - s_k)^2 \right] + \lambda \|\boldsymbol{W}\|_2 \quad (6)$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ represent all the weight and bias parameters in TreNet, λ is a hyperparameter for the regularization term. The cost function is differentiable and the architecture of TreNet allows the gradient of loss function (6) to be backpropagated to both LSTM and CNN parts.

4 Experimental Analysis

In this section, we report experimental results to demonstrate the advantage of TreNet by comparing to a variety of baselines.

4.1 Experiment Setup

Dataset:

- **Power Consumption** (PC). This dataset[2] contains measurements of electric power consumption with a one-minute sampling rate over a period of almost 4 years. We use the voltage time series throughout the experiments.
- **Gas Sensor** (GasSensor). This dataset[3] contains the recordings of chemical sensors exposed to dynamic gas mixtures at varying concentrations. The measurement was constructed by the continuous acquisition of the sensor array signals for a duration of about 12 hours without interruption. We mainly use the gas mixture time series regarding Ethylene and Methane in air.
- **Stock Transaction** (Stock): This dataset is extracted from Yahoo Finance and contains the daily stock transaction information in New York Stock Exchange from 1950-10 to 2016-4.

For the ease of result interpretation, the slope of the trends is represented by the angle in a bounded value range $[-90, 90]$. The duration of trends is measured by the number of data points covered by the trend. Data instances are built by combining historical trend sequence, local raw data and the target trend *w.r.t.* each time series subsequence. We then do random shuffling over such data instances, where 10% of the data instances is held out as the testing dataset and the rest is used for cross-validation. Totally, there exist 42279 instances in the power consumption dataset, 4418 instances for the gas sensor dataset and 10014 instances for the stock dataset.

Baselines: We compare TreNet with the following six baselines:

- **CNN**. This baseline method predicts the trend by using CNN over the raw data of time series to learn features used for the trend forecasting.
- **LSTM**. This method uses LSTM to learn the trend sequence $\mathcal{T}$ and then predicts the trend.

[2]https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

[3]https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures

- **ConvNet+LSTM(CLSTM)**. It is based on the cascade structure of ConvNet and LSTM proposed in [Bashivan *et al.*, 2015] which feeds the features learnt by ConvNet over time series raw data to an LSTM and obtains the prediction from the LSTM.
- **Support Vector Regression (SVR)**. A family of support vector regression based approaches with different kernel methods are used for the trend forecasting. We consider three commonly used kernels [Liu *et al.*, 2015], i.e., Radial Basis kernel (**SVRBF**), Polynomial kernel (**SVPOLY**), Sigmoid kernel (**SVSIG**). The trend sequence and the corresponding set of local time series data are concatenated as the input features to such SVR approaches.
- **Pattern-based Hidden Markov Model (pHMM)**. [Wang *et al.*, 2011] proposed a pattern-based hidden Markov model (HMM), which segments the time series and models the dependency in segments via HMM. The derived HMM model is used to predict the state of time series and then to estimate the trend based on the state.
- **Naive**. This is the naive approach which takes the duration and slope of the last trend as the prediction for the next one.

Evaluation metric: We evaluate the predictive performance of TreNet and baselines in terms of Root Mean Square Error (RMSE). The lower the RMSE, the more accurate the predictions.

Training: In TreNet, CNN has two stacked convolutional layers, which have 32 filters of size 2 and 4. The number of memory cells in LSTM is 600. In addition to the learning rate, the number of neurons in the feature fusion layer is chosen from the range $\{300, 600, 900, 1200\}$ to achieve the best performance.

We use dropout and L2 regularization to control the capacity of neural networks to prevent overfitting, and set the values to 0.5 and 5×10^{-4} respectively for all datasets [Mao *et al.*, 2014]. Regarding the SVR based approaches, we carefully tune the parameters c (error penalty), d (degree of kernel function), and γ (kernel coefficient) for kernels. Each parameter is selected from the sets $c \in \{10^{-5}, 10^{-4}, \ldots, 1, \ldots, 10^4, 10^5\}$, $d \in \{1, 2, 3\}$, $\gamma \in \{10^{-5}, 10^{-4}, \ldots, 1, \ldots, 10^5\}$ respectively.

4.2 Experiment Results

Table 1 studies the prediction performances of TreNet and baselines on different datasets. For the approaches requiring local raw data in the training data (i.e., SVRBF, SVPOLY, SVSIG, and TreNet), the size of local data is chosen by cross validation.

In Table 1, we observe that TreNet consistently outperforms baselines on the duration and slope prediction by achieving around 30% less errors at the maximum. Compared with CNN and LSTM baselines, the results verify that the hybrid architecture of TreNet can improve the performance by utilizing the information captured by both CNN and LSTM. Specifically, pHMM method performs worse due to the limited representation capability of HMM. On the slope prediction, SVR based approaches can get comparable results as TreNet. Meanwhile, the results show that the trend in PC dataset is less predictive compared with that of Stock and GasSensor datasets.

In the following group of experiments, we investigate the effect of local data size (i.e., w) on the prediction performance. In particular, we tune the value of local data size for the approaches whose input data contains local time series data and observe the prediction errors. Such approaches include SVRBF, SVPOLY, SVSIG and TreNet. LSTM only consumes the trend sequence and thus is not included. Baseline Naive has no original time series data as input, while CLSTM works on the whole time series. Thus they are excluded from this set of experiments as well. We report the results on each dataset in Table 2 to Table 7.

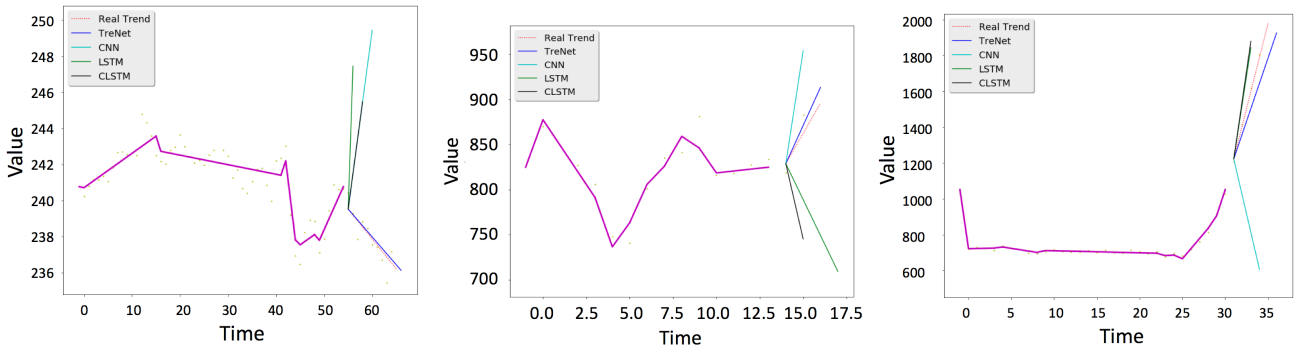
Dataset	Model	Duration	Slope
PC	CNN	27.51	13.56
	LSTM	27.27	13.27
	CLSTM	25.97	13.77
	SVRBF	31.81	12.94
	SVPOLY	31.81	12.93
	SVSIG	31.80	12.93
	pHMM	34.06	26.00
	Naive	39.68	21.17
	TreNet	**25.62**	**12.89**
Stock	CNN	18.87	12.78
	LSTM	11.07	8.40
	CLSTM	9.26	7.31
	SVRBF	11.39	7.47
	SVPOLY	11.44	7.41
	SVSIG	11.36	7.42
	pHMM	36.37	8.70
	Naive	11.36	8.58
	TreNet	**8.51**	**6.58**
GasSensor	CNN	53.99	11.51
	LSTM	55.77	11.22
	CLSTM	54.20	14.86
	SVRBF	61.45	9.61
	SVPOLY	63.83	10.37
	SVSIG	68.09	11.54
	pHMM	111.62	13.07
	Naive	53.76	10.57
	TreNet	**51.25**	**9.46**

Table 1: RMSE of the prediction of trend duration and slope on each dataset.

Window Size	SVRBF	SVPOLY	SVSIG	TreNet
300	31.17	31.61	31.66	25.94
500	31.81	31.81	31.80	25.89
700	31.10	31.09	31.11	25.72
900	31.28	31.27	31.27	**25.62**

Table 2: RMSE of the duration predictions *w.r.t.* different sizes of local data in PC dataset.

In Table 2, we observe that compared to baselines TreNet has the lowest errors on the duration prediction across different window sizes. As the window size increases and more



(a) PC (b) Stock (c) GasSensor

Figure 4: Visualization of the trend prediction by TreNet on examples from PC, Stock and GasSensor datasets. The yellow points represent time series and the purple line in each figure represents the associated historical trend sequence.

Window Size	SVRBF	SVPOLY	SVSIG	TreNet
300	12.93	12.9346	12.9345	13.15
500	12.94	12.9342	12.9346	12.89
700	12.93	12.9345	12.9345	**12.86**
900	12.94	12.9350	12.9346	12.96

Table 3: RMSE of the slope predictions *w.r.t.* different sizes of local data in PC dataset.

local data points are fed to the training process, the prediction errors of TreNet decrease and then stabilize. This is because feature fusion and output layer of TreNet selectively focus on the features having strong predictive power and giving overwhelming local data only gives rise to marginal improvements. Such similar phenomenon is observed in the rest of tables. The observation in above experiments w.r.t. the varying window size provides inspiration for choosing the size of local data instead of expensive cross validation. Given the training dataset, we can find out the maximum duration of local trends and takes it as the local data size. This is because doing so can ensure that the range of local data in each training instance can cover the most recent local trend, whose raw data is believed to have strong predictive power for the subsequent trend.

Window Size	SVRBF	SVPOLY	SVSIG	TreNet
300	11.41	11.44	11.42	8.85
500	11.39	11.44	11.36	**8.51**
700	11.45	11.59	11.58	8.58
900	11.32	11.47	11.59	8.78

Table 4: RMSE of the duration predictions on different sizes of local data in Stock dataset.

Window Size	SVRBF	SVPOLY	SVSIG	TreNet
300	7.42	7.51	7.38	**6.53**
500	7.47	7.41	7.42	6.58
700	7.53	7.58	7.51	6.75
900	7.61	7.45	7.59	6.73

Table 5: RMSE of the slope predictions on different sizes of local data in Stock dataset.

In the next group of experiments in Figure 4, we visualize the trend prediction using sample testing data instances from each dataset. The fusion layer of TreNet selectively uses complementary information from CNN and LSTM and therefore we can observe that in PC TreNet successfully predicts the changed trend, though there are successive upward trends before. However, for the cascade structure of CLSTM it fails to capture the effect of local and abrupt changing data on the trend evolution.

Window Size	SVRBF	SVPOLY	SVSIG	TreNet
300	62.81	70.91	85.69	52.28
500	61.86	64.33	91.51	51.77
700	61.20	63.89	78.20	**51.15**
900	61.45	63.83	68.09	51.25

Table 6: RMSE of the duration predictions on different sizes of local data in GasSensor dataset.

Window Size	SVRBF	SVPOLY	SVSIG	TreNet
300	10.21	10.95	11.92	9.57
500	10.08	10.65	11.64	9.60
700	9.54	10.44	11.72	9.55
900	9.61	10.37	11.54	**9.46**

Table 7: RMSE of the slope predictions on different sizes of local data in GasSensor dataset.

5 Conclusion

In this paper we propose TreNet, a novel hybrid neural network over time series and associated trend sequence to predict the trend evolution of time series. The experimental results demonstrate that such a hybrid network can indeed capture complementary information to enhance the prediction performance. Moreover, such architecture is generic and extendible in that additional exogenous time series can be fed to TreNet.

Acknowledgements

This work was partly supported by Nano-Tera.ch through the OpenSense2 project.

References

[Atsalakis and Valavanis, 2009] George S Atsalakis and Kimon P Valavanis. Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications*, 36(7):10696–10707, July 2009.

[Ballas *et al.*, 2015] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[Bao *et al.*, 2014] Yukun Bao, Tao Xiong, and Zhongyi Hu. Multi-step-ahead time series prediction using multiple-output support vector regression. *Neurocomputing*, 129:482–493, 2014.

[Bashivan *et al.*, 2015] Pouya Bashivan, Irina Rish, Mohammed Yeasin, and Noel Codella. Learning representations from eeg with deep recurrent-convolutional neural networks. *arXiv preprint arXiv:1511.06448*, 2015.

[Chang *et al.*, 2012] Li-Chiu Chang, Pin-An Chen, and Fi-John Chang. Reinforced two-step-ahead weight adjustment technique for online training of recurrent neural networks. *IEEE transactions on neural networks and learning systems*, 23(8):1269–1278, 2012.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Donahue *et al.*, 2015] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE CVPR*, pages 2625–2634, 2015.

[Guo *et al.*, 2016] Tian Guo, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya. Robust online time series prediction with recurrent neural networks. In *2016 IEEE DSAA*, pages 816–825. IEEE, 2016.

[Hammerla *et al.*, 2016] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Lipton *et al.*, 2015] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[Liu *et al.*, 2015] Jiajun Liu, Kun Zhao, Brano Kusy, Ji-rong Wen, and Raja Jurdak. Temporal embedding in convolutional neural networks for robust learning of abstract snippets. *arXiv preprint arXiv:1502.05113*, 2015.

[Malhotra *et al.*, 2015] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *European Symposium on Artificial Neural Networks*, volume 23, 2015.

[Mao *et al.*, 2014] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[Matsubara *et al.*, 2014] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. Autoplait: Automatic mining of co-evolving time sequences. In *ACM SIGMOD*, pages 193–204. ACM, 2014.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[Taieb and Atiya, 2016] Souhaib Ben Taieb and Amir F Atiya. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE transactions on neural networks and learning systems*, 27(1):62–76, 2016.

[Venkatraman *et al.*, 2015] Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030, 2015.

[Vinyals *et al.*, 2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE CVPR*, pages 3156–3164, 2015.

[Wang and Oates, 2015] Zhiguang Wang and Tim Oates. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at AAAI*, 2015.

[Wang *et al.*, 2011] Peng Wang, Haixun Wang, and Wei Wang. Finding semantics in time series. In *ACM KDD*, pages 385–396. ACM, 2011.

[Wang *et al.*, 2016a] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. *arXiv preprint arXiv:1604.04573*, 2016.

[Wang *et al.*, 2016b] Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. Morphological segmentation with window lstm neural networks. In *AAAI*, 2016.

[Yang *et al.*, 2015] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 25–31, 2015.

[Yuan, 2015] Chao Yuan. Unsupervised machine condition monitoring using segmental hidden markov models. In *AAAI*, pages 4009–4016. AAAI Press, 2015.

[Zhao and Magoulès, 2012] Hai-xiang Zhao and Frédéric Magoulès. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6):3586–3592, 2012.