arXiv:1806.08946v1 [cs.LG] 23 Jun 2018

Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis

Jingyuan Wang, Ze Wang, Jianfeng Li, Junjie Wu
Beihang University, Beijing, China
{jywang,ze.w,leejianfeng,wujj}@buaa.edu.cn

ABSTRACT

Recent years have witnessed the unprecedented rising of time series from almost all kindes of academic and industrial fields. Various types of deep neural network models have been introduced to time series analysis, but the important frequency information is yet lack of effective modeling. In light of this, in this paper we propose a wavelet-based neural network structure called multilevel Wavelet Decomposition Network (mWDN) for building frequency-aware deep learning models for time series analysis. mWDN preserves the advantage of multilevel discrete wavelet decomposition in frequency learning while enables the fine-tuning of all parameters under a deep neural network framework. Based on mWDN, we further propose two deep learning models called Residual Classification Flow (RCF) and multi-frequecy Long Short-Term Memory (mLSTM) for time series classification and forecasting, respectively. The two models take all or partial mWDN decomposed sub-series in different frequencies as input, and resort to the back propagation algorithm to learn all the parameters globally, which enables seamless embedding of wavelet-based frequency analysis into deep learning frameworks. Extensive experiments on 40 UCR datasets and a real-world user volume dataset demonstrate the excellent performance of our time series models based on mWDN. In particular, we propose an importance analysis method to mWDN based models, which successfully identifies those time-series elements and mWDN layers that are crucially important to time series analysis. This indeed indicates the interpretability advantage of mWDN, and can be viewed as an indepth exploration to interpretable deep learning.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; *Supervised learning by classification*; *Supervised learning by regression*;

KEYWORDS

Time series analysis, Multilevel wavelet decomposition network, Deep learning, Importance analysis

ACM Reference Format:
Jingyuan Wang, Ze Wang, Jianfeng Li, Junjie Wu. 2018. Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. In *KDD 2018: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220060

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD 2018, August 19–23, 2018, London, United Kingdom
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220060

1 INTRODUCTION

A time series is a series of data points indexed in time order. Methods for time series analysis could be classified into two types: time-domain methods and frequency-domain methods.[1] Time-domain methods consider a time series as a sequence of ordered points and analyze correlations among them. Frequency-domain methods use transform algorithms, such as discrete Fourier transform and Z-transform, to transform a time series into a frequency spectrum, which could be used as features to analyze the original series.

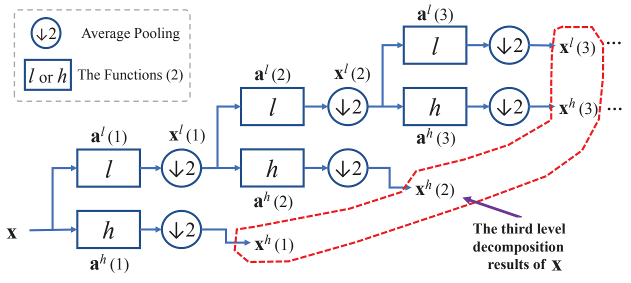
In recent years, with the booming of deep learning concept, various types of deep neural network models have been introduced to time series analysis and achieved state-of-the-art performances in many real-life applications [28, 38]. Some well-known models include Recurrent Neural Networks (RNN) [40] and Long Short-Term Memory (LSTM) [14] that use memory nodes to model correlations of series points, and Convolutional Neural Network (CNN) that uses trainable convolution kernels to model local shape patterns [42]. Most of these models fall into the category of time-domain methods without leveraging frequency information of a time series, although some begin to consider in indirect ways [6, 19].

Wavelet decompositions [7] are well-known methods for capturing features of time series both in time and frequency domains. Intuitively, we can employ them as feature engineering tools for data preprocessing before a deep modeling. While this loose coupling way might improve the performance of raw neural network models [24], they are not globally optimized with independent parameter inference processes. How to integrate wavelet transforms into the framework of deep learning models remains a great challenge.

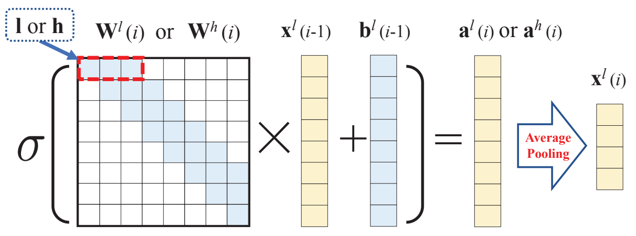
In this paper, we propose a wavelet-based neural network structure, named *multilevel Wavelet Decomposition Network* (mWDN), to build frequency-aware deep learning models for time series analysis. Similar to the standard *Multilevel Discrete Wavelet Decomposition* (MDWD) model [26], mWDN can decompose a time series into a group of sub-series with frequencies ranked from high to low, which is crucial for capturing frequency factors for deep learning. Different from MDWD with fixed parameters, however, all parameters in mWDN can be fine-turned to fit training data of different learning tasks. In other words, mWDN can take advantages of both wavelet based time series decomposition and the learning ability of deep neural networks.

Based on mWDN, two deep learning models, *i.e.*, Residual Classification Flow (RCF) and multi-frequency Long Short-Term Memory

[1]https://en.wikipedia.org/wiki/Time_series



(a) Illustration of the mWDN Framework



l or h
$\mathbf{W}^l(i)$ or $\mathbf{W}^h(i)$
$\mathbf{x}^l(i\text{-}1)$
$\mathbf{b}^l(i\text{-}1)$
$\mathbf{a}^l(i)$ or $\mathbf{a}^h(i)$
$\mathbf{x}^l(i)$
Average Pooling
σ

(b) Approximative Discrete Wavelet Transform

Figure 1: The mWDN framework.

(mLSTM), are designed for time series classification (TSC) and forecasting (TSF), respectively. The key issue in TSC is to extract as many as possible representative features from time series. The RCF model therefore adopts the mWDN decomposed results in different levels as inputs, and employs a pipelined classifier stack to exploit features hidden in sub-series through residual learning methods. For the TSF problem, the key issue turns to inferring future states of a time series according to the hidden trends in different frequencies. Therefore, the mLSTM model feeds all mWDN decomposed sub-series in high frequencies into independent LSTM models, and ensembles all LSTM outputs for final forecasting. Note that all parameters of RCF and mLSTM including the ones in mWDN are trained using the back propagation algorithm in an *end-to-end* manner. In this way, the wavelet-based frequency analysis is seamlessly embedded into deep learning frameworks.

We evaluate RCF on 40 UCR time series datasets for TSC, and mLSTM on a real-world user-volume time series dataset for TSF. The results demonstrate their superiorities to state-of-the-art baselines and the advantages of mWDN with trainable parameters. As a nice try for interpretable deep learning, we further propose an importance analysis method to mWDN based models, which successfully identifies those time-series elements and mWDN layers that are crucially important to the success of time series analysis. This indicates the interpretability advantage of mWDN by integrating wavelet decomposition for frequency factors.

2 MODEL

Throughout the paper, we use lowercase symbols such as a, b to denote scalars, bold lowercase symbols such as $\mathbf{a}, \mathbf{b}$ to denote vectors, bold uppercase symbols such as $\mathbf{A}, \mathbf{B}$ to denote matrices, and uppercase symbols such as A, B, to denote constant.

2.1 Multilevel Discrete Wavelet Decomposition

Multilevel Discrete Wavelet Decomposition (MDWD) [26] is a wavelet based discrete signal analysis method, which can extract multilevel time-frequency features from a time series by decomposing the series as low and high frequency sub-series level by level.

We denote the input time series as $\mathbf{x} = \{x_1, \ldots, x_t, \ldots, x_T\}$, and the low and high sub-series generated in the i-th level as $\mathbf{x}^l(i)$ and $\mathbf{x}^h(i)$. In the $(i+1)$-th level, MDWD uses a low pass filter $\mathbf{l} = \{l_1, \ldots, l_k, \ldots, l_K\}$ and a high pass filter $\mathbf{h} = \{h_1, \ldots, h_k, \ldots, h_K\}$, $K \ll T$, to convolute low frequency sub-series of the upper level as

$$\begin{aligned} a_n^l(i+1) &= \sum_{k=1}^{K} x_{n+k-1}^l(i) \cdot l_k, \\ a_n^h(i+1) &= \sum_{k=1}^{K} x_{n+k-1}^l(i) \cdot h_k, \end{aligned} \tag{1}$$

where $x_n^l(i)$ is the n-th element of the low frequency sub-series in the i-th level, and $\mathbf{x}^l(0)$ is set as the input series. The low and high frequency sub-series $\mathbf{x}^l(i)$ and $\mathbf{x}^h(i)$ in the level i are generated from the 1/2 down-sampling of the intermediate variable sequences $\mathbf{a}^l(i) = \left\{a_1^l(i), a_2^l(i), \ldots\right\}$ and $\mathbf{a}^h(i) = \left\{a_1^h(i), a_2^h(i), \ldots\right\}$.

The sub-series set $\mathcal{X}(i) = \left\{\mathbf{x}^h(1),\ \mathbf{x}^h(2),\ \ldots,\ \mathbf{x}^h(i),\ \mathbf{x}^l(i)\right\}$ is called as the i-th level decomposed results of $\mathbf{x}$. Specifically, $\mathcal{X}(i)$ satisfies: 1) We can fully reconstruct $\mathbf{x}$ from $\mathcal{X}(i)$; 2) The frequency from $\mathbf{x}^h(1)$ to $\mathbf{x}^l(i)$ is from high to low; 3) For different layers, $\mathcal{X}(i)$ has different time and frequency resolutions. As i increases, the frequency resolution is increasing and the time resolution, especially for low frequency sub-series, is decreasing.

Because the sub-series with different frequencies in $\mathcal{X}$ keep the same order information with the original series $\mathbf{x}$, MDWD is regarded as time-frequency decomposition.

2.2 Multilevel Wavelet Decomposition Network

In this section, we propose a multilevel Wavelet Decomposition Network (mWDN), which approximatively implements a MDWD under a deep neural network framework.

The structure of mWDN is illustrated in Fig. 1. As shown in the figures, the mWDN model hierarchically decomposes a time series using the following two functions

$$\begin{aligned} \mathbf{a}^l(i) &= \sigma\left(\mathbf{W}^l(i)\mathbf{x}^l(i-1) + \mathbf{b}^l(i)\right), \\ \mathbf{a}^h(i) &= \sigma\left(\mathbf{W}^h(i)\mathbf{x}^l(i-1) + \mathbf{b}^h(i)\right), \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is a sigmoid activation function, and $\mathbf{b}^l(i)$ and $b^h(i)$ are trainable bias vectors initialized as close-to-zero random values. We can see the functions in Eq. (2) have similar forms as the functions

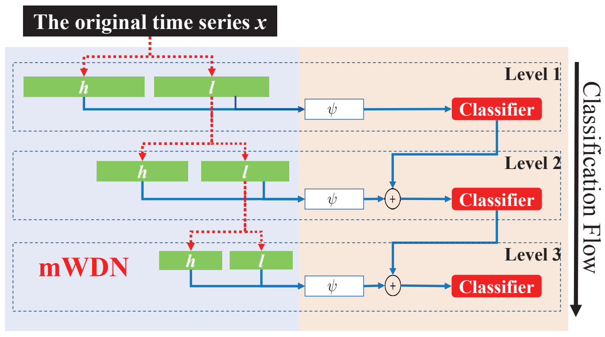


Figure 2: The RCF framework.

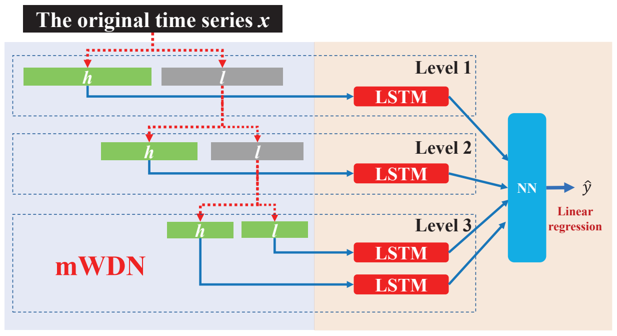


Figure 3: The mLSTM framework.

in Eq. (1) for MDWD. $\mathbf{x}^l(i)$ and $\mathbf{x}^h(i)$ also denote the low and high frequency sub-series of $\mathbf{x}$ generated in the i-th level, which are down-sampled from the intermediate variables $\mathbf{a}^l(i)$ and $\mathbf{a}^h(i)$ using an average pooling layer as $x^l_j(i) = (a^l_{2j}(i) + a^l_{2j-1}(i))/2$.

In order to implement the convolution defined in Eq. (1), we set the initial values of the weight matrices $\mathbf{W}^l$ and $\mathbf{W}^h$ as

$$\mathbf{W}^l(i) = \begin{bmatrix} l_1 & l_2 & l_3 & \cdots & l_K & \epsilon & \cdots & \epsilon \\ \epsilon & l_1 & l_2 & \cdots & l_{K-1} & l_K & \cdots & \epsilon \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & l_1 & \cdots & l_{K-1} & l_K \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & l_1 & l_2 \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & \epsilon & l_1 \end{bmatrix}, \tag{3}$$

$$\mathbf{W}^h(i) = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_K & \epsilon & \cdots & \epsilon \\ \epsilon & h_1 & h_2 & \cdots & h_{K-1} & h_K & \cdots & \epsilon \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & h_1 & \cdots & h_{K-1} & h_K \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & h_1 & h_2 \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & \epsilon & h_1 \end{bmatrix}. \tag{4}$$

Obviously, $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i) \in \mathbb{R}^{P\times P}$, where P is the size of $\mathbf{x}^l(i-1)$. The ϵ in the weight matrices are random values that satisfy $|\epsilon| \ll |l|, \forall l \in \mathbf{l}$ and $|\epsilon| \ll |h|, \forall h \in \mathbf{h}$. We use the Daubechies 4 Wavelet [29] in our practice, where the filter coefficients are set as

$$\begin{aligned} \mathbf{l} =& \{-0.0106, 0.0329, 0.0308, -0.187, \\ & -0.028, 0.6309, 0.7148, 0.2304\}, \\ \mathbf{h} =& \{-0.2304, 0.7148, -0.6309, -0.028, \\ & 0.187, 0.0308, -0.0329, -0.0106\}. \end{aligned}$$

From Eq. (2) to Eq. (3), we use the deep neural network framework to implement an approximate MDWD. It is noteworthy that although the weight matrices $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$ are initialized as the filter coefficients of MDWD, they are still trainable according to real data distributions.

2.3 Residual Classification Flow

The task of TSC is to predict unknown category label of a time series. A key issue of TSC is extracting distinguishing features from time series data. The decomposed results $\mathcal{X}$ of mWDN are natural time-frequency features that could be used in TSC. In this subsection, we propose a Residual Classification Flow (RCF) network to exploit the potentials of mWDN in TSC.

The framework of RCF is illustrated in Fig. 2. As shown in the figure, RCF contains many independent classifiers. The RCF model connects the sub-series generated by the i-th mWDN level, *i.e.*, $\mathbf{x}^h(i)$ and $\mathbf{x}^l(i)$, with a forward neural network as

$$\mathbf{u}(i) = \psi\left(\mathbf{x}^h(i), \mathbf{x}^l(i), \theta^\psi\right), \tag{5}$$

where $\psi(\cdot)$ could be a multilayer perceptron, a convolutional network, or any other types of neural networks, and θ^ψ represents the trainable parameters. Moreover, RCF adopts a residual learning method [13] to join $\mathbf{u}(i)$ of all classifiers as

$$\hat{\mathbf{c}}(i) = S\left(\hat{\mathbf{c}}(i-1) + \mathbf{u}(i)\right), \tag{6}$$

where $S(\cdot)$ is a softmax classifier, $\hat{\mathbf{c}}_i$ is a predicted value of one-hot encoding of the category label of the input series.

In the RCF model, the decomposed results of all mWDN levels, *i.e.* $\mathcal{X}(1), \ldots, \mathcal{X}(N)$, are evolved. Because the decomposed results in different mWDN levels have different time and frequency resolutions [26], the RCF model can fully exploit patterns of the input time series from different time/frequency-resolutions. In other words, RCF employs a multi-view learning methodology to achieve high-performance time series classification.

Moreover, deep residual networks [13] were proposed to solve the problem that using deeper network structures may result in a great training difficulty. The RCF model also inherits this merit. In Eq. (6), the i-th classifier makes decision based on $\mathbf{u}(i)$ and the decision made by the $(i-1)$-th classifier, which can learn from $\mathbf{u}(i)$ the incremental knowledge that the $(i-1)$-th classifier does not have. Therefore, users could append residual classifiers one after another until classification performance does not increase any more.

2.4 Multi-frequency Long Short-Term Memory

In this subsection, we propose a multi-frequency Long-Short Term Memory (mLSTM) model based on mWDN for TSF. The design of mLSTM is based on the insight that the temporal correlations of points hidden in a time series have close relations with frequency. For example, large time scale correlations, such as long-term tendencies, usually lay in low frequency, and the small time scale correlations, such as short-term disturbances and events, usually lay in high frequency. Therefore, we could divide a complicated TSF problem as many sub-problems of forecasting sub-series decomposed by mWDN, which are relatively easier because the frequency components in the sub-series are simpler.

Given a time series with infinite length, on which we open a T size slide window from the past to the time t as

$$\mathbf{x} = \{x_{t-T+1}, \dots, x_{t-1}, x_t\}. \tag{7}$$

Using mWDN to decompose $\mathbf{x}$, we get the low and high frequency component series in the i-th level as

$$\begin{aligned} \mathbf{x}^l(i) &= \{x^l_{t-\frac{T}{2^n}+1}(i), \dots, x^l_{t-1}(i), x^l_t(i)\}, \\ \mathbf{x}^h(i) &= \{x^h_{t-\frac{T}{2^n}+1}(i), \dots, x^h_{t-1}(i), x^h_t(i)\}. \end{aligned} \tag{8}$$

As shown in Fig. 3, the mLSTM model uses the decomposed results of the last level, *i.e.*, the sub-series in $\mathcal{X}(N) = \{\mathbf{x}^h(1), \mathbf{x}^h(2), \dots, \mathbf{x}^h(N), \mathbf{x}^l(N)\}$, as the inputs of $N+1$ independent LSTM sub-networks. Every LSTM sub-network forecasts the future state of one sub-series in $\mathcal{X}(N)$. Finally, a fully connected neural network is employed to fuse the LSTM sub-networks as an ensemble for forecasting.

3 OPTIMIZATION

In TSC applications, we adopt a deep supervision method to train the RCF model [37]. Given a set of time series $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$, we use cross-entropy as loss metric and define the objective function of the i-th classifier as

$$\tilde{\mathcal{J}}^c(i) = -\frac{1}{M}\sum_{m=1}^{M}\left(\mathbf{c}_m^\top \ln \hat{\mathbf{c}}_m(i) + (1-\mathbf{c}_m)^\top \ln(1-\hat{\mathbf{c}}_m(i))\right), \tag{9}$$

where $\mathbf{c}_m$ is the one-hot encoding of $\mathbf{x}_m$'s real category, and $\hat{\mathbf{c}}_m(i)$ is the softmax output of the i-th classifier with the input $\mathbf{x}_m$. For a RCF with N classifiers, the final objective function is a weighted sum of all $\tilde{\mathcal{J}}(i)$ [37]:

$$\mathcal{J}^c = \sum_{i=1}^{N}\frac{i}{N}\tilde{\mathcal{J}}^c(i). \tag{10}$$

The result of the last classifier, $\hat{\mathbf{c}}(N)$, is used as the final classification result of RCF.

In TSF applications, we adopt a pre-training and fine turning method to train the mLSTM model. In the pre-training step, we use MDWD to decompose the real value of the future state to be predicted as N wavelet components, *i.e.* $\mathbf{y}^p = \{y^h(1), y^h(2), \dots, y^h(N), y^l(N)\}$, and then combine the outputs of all LSTM sub-network as $\hat{\mathbf{y}}^p$, then the objective function of the pre-training step is defined as

$$\tilde{\mathcal{J}}^f = -\frac{1}{M}\sum_{m=1}^{M}\|\mathbf{y}_m - \hat{\mathbf{y}}^p_m\|_F^2, \tag{11}$$

where $\|\cdot\|_F$ is the Frobenius Norm. In the fine-turning step, we use the following objective function to train mLSTM based on the parameters learned in the pre-training step:

$$\mathcal{J}^f = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}-y)^2, \tag{12}$$

where $\hat{y}$ is future state predicted by mLSTM and y is the real value.

We use the error back propagation (BP) algorithm to optimize the objective functions. Denoting θ as the parameters of the RCF or mLSTM model, the BP algorithm iteratively updates θ as

$$\theta \leftarrow \theta - \eta\frac{\partial \mathcal{J}(\theta)}{\partial \theta}, \tag{13}$$

where η is an adjustable learning rate. The weight matrices $\mathbf{W}^h(i)$ and $\mathbf{W}^l(i)$ of mWDN are also trainable in Eq. (13). A problem of training parameters with preset initial values like $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$ is that the model may "forget" the initial values [9] in the training process. To deal with this, we introduce two regularization items to the objective function and therefore have

$$\begin{aligned} \mathcal{J}^* = \mathcal{J}(\theta) &+ \alpha\sum_i \|\mathbf{W}^l(i) - \tilde{\mathbf{W}}^l(i)\|_F^2 \\ &+ \beta\sum_i \|\mathbf{W}^h(i) - \tilde{\mathbf{W}}^h(i)\|_F^2, \end{aligned} \tag{14}$$

where $\tilde{\mathbf{W}}^l(i)$ and $\tilde{\mathbf{W}}^h(i)$ are the same matrices as $\mathbf{W}^h(i)$ and $\mathbf{W}^h(i)$ except that $\epsilon = 0$, and α, β are hyper-parameters which are set as empirical values. Accordingly, the BP algorithm iteratively updates the weight matrices of mWDN as

$$\begin{aligned} \mathbf{W}^l(i) &\leftarrow \mathbf{W}^l(i) - \eta\left(\frac{\partial \mathcal{J}}{\partial \mathbf{W}^l(i)} - 2\alpha\left(\mathbf{W}^l(i) - \tilde{\mathbf{W}}(i)\right)\right), \\ \mathbf{W}^h(i) &\leftarrow \mathbf{W}^h(i) - \eta\left(\frac{\partial \mathcal{J}}{\partial \mathbf{W}^h(i)} - 2\beta\left(\mathbf{W}^l(i) - \tilde{\mathbf{W}}(i)\right)\right). \end{aligned} \tag{15}$$

In this way, the weights in mWDN will converge to a point that is near to the wavelet decomposed prior, unless wavelet decomposition is far inappropriate to the task.

4 EXPERIMENTS

In this section, we evaluate the performance of the mWDN-based models in both the TSC and TSF tasks.

4.1 Task I: Time Series Classification

Experimental Setup. The classification performance was tested on 40 datasets of the UCR time series repository [4], with various competitors as follows:

- *RNN* and *LSTM*. Recurrent Neural Networks [40], and Long Short-Term Memory [14] are two kinds of classical deep neural networks widely used in time series analysis.
- *MLP*, *FCN*, and *ResNet*. These three models were proposed in [38] as strong baselines on the UCR time series datasets. They have the same framework: an input layer, followed by three hidden basic blocks, and finally a softmax output. MLP adopts a fully-connected layer as its basic block, FCN and ResNet adopt a fully convolutional layer and a residual convolutional network, respectively, as their basic blocks.
- *MLP-RCF*, *FCN-RCF*, and *ResNet-RCF*. The three models use the basic blocks of MLP/FCN/ResNet as the ψ model of RCF

Table 1: Comparison of Classification Performance on 40 UCR Time Series Datasets

Err Rate	RNN	LSTM	MLP	FCN	ResNet	**MLP-RCF**	**FCN-RCF**	**ResNet-RCF**	Wavelet-RCF
Adiac	0.233	0.341	0.248	**0.143**	0.174	0.212	0.155	*0.151*	0.162
Beef	0.233	0.333	0.167	0.25	0.233	*0.06*	**0.03**	*0.06*	*0.06*
CBF	0.189	0.118	0.14	**0**	*0.006*	0.056	**0**	**0**	0.016
ChlorineConcentration	0.135	0.16	0.128	0.157	0.172	0.096	**0.068**	*0.07*	0.147
CinCECGtorso	0.333	0.092	0.158	0.187	0.229	0.117	*0.014*	0.084	**0.011**
CricketX	0.449	0.382	0.431	*0.185*	**0.179**	0.321	0.216	0.297	0.211
CricketY	0.415	0.318	0.405	0.208	0.195	0.254	**0.172**	0.301	*0.192*
CricketZ	0.4	0.328	0.408	*0.187*	0.187	0.313	**0.162**	0.275	**0.162**
DiatomSizeReduction	0.056	0.101	0.036	0.07	0.069	**0.013**	*0.023*	0.026	0.028
ECGFiveDays	0.088	0.417	0.03	*0.015*	0.045	0.023	**0.01**	0.035	0.016
FaceAll	0.247	0.192	0.115	**0.071**	0.166	0.094	0.098	0.126	*0.076*
FaceFour	0.102	0.364	0.17	0.068	0.068	0.102	**0.05**	*0.057*	0.058
FacesUCR	0.204	0.091	0.185	*0.052*	**0.042**	0.15	0.087	0.102	0.087
50words	0.316	0.284	0.288	0.321	*0.273*	0.316	0.288	**0.258**	0.3
FISH	0.126	0.103	0.126	0.029	**0.011**	0.086	*0.021*	0.034	0.026
GunPoint	0.1	0.147	0.067	**0**	*0.007*	0.033	**0**	0.02	**0**
Haptics	0.594	0.529	0.539	**0.449**	0.495	0.480	*0.461*	0.473	0.476
InlineSkate	0.667	0.638	0.649	0.589	0.635	**0.543**	*0.566*	0.578	0.572
ItalyPowerDemand	0.055	0.072	0.034	0.03	0.04	0.031	**0.023**	0.034	*0.028*
Lighting2	**0**	**0**	0.279	0.197	0.246	0.213	*0.145*	0.197	0.162
Lighting7	0.288	0.384	0.356	*0.137*	0.164	0.179	**0.091**	0.177	0.144
MALLAT	0.119	0.127	0.064	**0.02**	*0.021*	0.058	0.044	0.046	0.024
MedicalImages	0.299	0.276	0.271	0.208	0.228	0.251	**0.164**	*0.188*	0.206
MoteStrain	0.133	0.167	0.131	*0.05*	0.105	0.105	0.076	**0.032**	0.05
NonInvasiveFatalECGThorax1	0.09	0.08	0.058	0.039	0.052	*0.029*	**0.026**	0.04	0.042
NonInvasiveFatalECGThorax2	0.069	0.071	0.057	0.045	0.049	0.056	**0.028**	*0.033*	0.048
OliveOil	0.233	0.267	0.6	0.167	0.133	0.03	**0**	**0**	*0.012*
OSULeaf	0.463	0.401	0.43	**0.012**	0.021	0.342	*0.018*	0.021	0.021
SonyAIBORobotSurface	0.21	0.309	0.273	*0.032*	**0.015**	0.193	0.042	0.032	0.052
SonyAIBORobotSurfaceII	0.219	0.187	0.161	**0.038**	**0.038**	0.092	*0.064*	0.083	0.072
StarLightCurves	0.027	0.035	0.043	0.033	0.029	*0.021*	**0.018**	0.027	0.03
SwedishLeaf	0.085	0.128	0.107	*0.034*	0.042	0.089	0.057	**0.017**	0.046
Symbols	0.179	0.117	0.147	**0.038**	0.128	0.126	*0.04*	0.107	0.084
TwoPatterns	0.005	*0.001*	0.114	0.103	**0**	0.070	**0**	**0**	0.005
uWaveGestureLibraryX	0.224	0.195	0.232	0.246	0.213	0.213	0.218	*0.194*	**0.162**
uWaveGestureLibraryY	0.335	0.265	0.297	0.275	0.332	0.306	**0.232**	0.296	*0.241*
uWaveGestureLibraryZ	0.297	0.259	0.295	0.271	0.245	0.298	0.265	*0.204*	**0.194**
wafer	**0**	**0**	0.004	*0.003*	*0.003*	*0.003*	**0**	**0**	**0**
WordsSynonyms	0.429	0.343	0.406	0.42	0.368	0.391	*0.338*	0.387	**0.314**
yoga	0.202	0.158	0.145	0.155	0.142	0.138	**0.112**	0.139	*0.128*
Winning times	2	2	0	*9*	6	2	**19**	7	7
AVG arithmetic ranking	7.425	6.825	7.2	4.025	4.55	5.15	**2.175**	3.375	*3.075*
AVG geometric ranking	6.860	6.131	7.043	3.101	3.818	4.675	**1.789**	2.868	*2.688*
MPCE	0.039	0.043	0.041	0.023	0.025	0.028	**0.017**	0.021	*0.019*

in Eq. (5). We compare them with MPL/FCN/ResNet to verify the effectiveness of RCF.

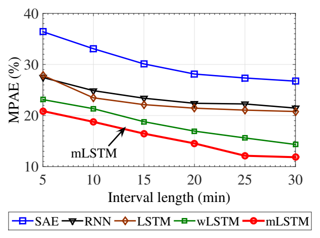
- *Wavelet-RCF.* This model has the same structure as ResNet-RCF but replaces the mWDN part with a standard MDWD with fixed parameters. We compare it with ResNet-RCF to verify the effectiveness of trainable parameters in mWDM.

For each dataset, we ran a model 10 times and returned the average *classification error rate* as the evaluation. To compare the overall performances on all the 40 data sets, we further introduced *Mean Per-Class Error* (MPCE) as the performance indicator for each competitor [38]. Let C_k denote the amount of categories in the kth dataset, and e_k the error rate of a model on that dataset, MPCE of a model is then defined as

$$\text{MPCE} = \frac{1}{K}\sum_{l=1}^{K}\frac{e_k}{C_k}. \tag{16}$$

Note that the factor of category amount is wiped out in MPCE. A smaller MPCE value indicates a better overall performance.

Results & Analysis. Table 1 shows the experimental results, with the summarized information listed in the bottom two lines. Note that the best performance for each dataset is highlighted in bold, and the second best is in italic. From the table, we have various



(a) Comparison by MAPE (b) Comparison by RMSE

Figure 4: Comparison of prediction performance with varying period lengths (Scenario I).

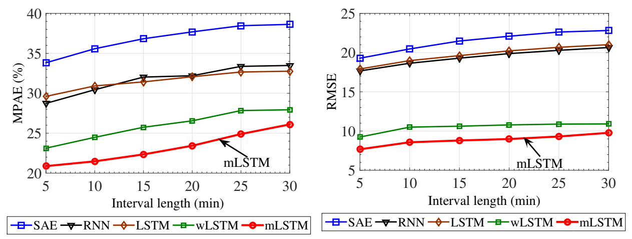
interesting observations. Firstly, it is clear that among all the competitors, FCN-RCF achieves the best performance in terms of both the largest number of wins (the best in 19 out of 40 datasets) and the smallest MPCE value. While the baseline FCN itself also achieves a satisfactory performance — the second largest number of wins at 9 and a rather small MPCE value at 0.023, the gap to FCM-RCF is still rather big, implying the significant benefit from adopting our RCF framework. This is actually not an individual case; from Table 1, MLP-RCF performs much better than MLP on 37 datasets, and the number for ResNet-RCF against ResNet is 27. This indicates RCF is indeed a general framework compatible with different types of deep learning classifiers and can improve TSF performance sharply.

Another observation is from the comparison between Wavelet-RCF and ResNet-RCF. Table 1 shows that Wavelet-RCF achieved the second overall performance on MPCE and AVG rankings, which indicates that the frequency information introduced by wavelet tools is very helpful for time series problems. It is clear from the table that ResNet-RCF outperforms Wavelet-RCF on most of the datasets. This strongly demonstrates the advantage of our RCF framework in adopting parameter-trainable mWDN under the deep learning architecture, rather than using directly the wavelet decomposition as a feature engineering tool. More technically speaking, compared with Wavelet-RCF, mWND-based ResNet-RCF can achieve a good tradeoff between the prior of frequency-domain and the likelihoods of training data. This well illustrates why RCF based models can achieve much better results in the previous observation.

Summary. The above experiments demonstrate the superiority of RCF based models to some state-of-the-art baselines in the TSC tasks. The experiments also imply that the trainable parameters in a deep learning architecture and the strong priors from wavelet decomposition are two key factors for the success of RCF.

4.2 Task II: Time Series Forecasting

Experimental Setup. We tested the predictive power of mLSTM on a visitor volume prediction scenario [35]. The experiment adopts a real-life dataset named *WuxiCellPhone*, which contains user-volume time series of 20 cell-phone base stations located in the downtown of Wuxi city during two weeks. Detail informantion of cell-phone data refers [30, 31, 34]. The time granularity of a user-volume series is 5 minutes. In the experiments, we compared mLSTM with the following baselines:



(a) Comparison by MAPE (b) Comparison by RMSE

Figure 5: Comparison of prediction performance with varying interval lengths (Scenario II).

- *SAE* (Stacked Auto-Encoders), which has been used in various TSF tasks [25].
- *RNN* (Recurrent Neural Networks) and *LSTM* (Long Short-Term Memory), which are specifically designed for time series analysis.
- *wLSTM* , which has the same structure with mLSTM but replaces the mWDN part with a standard MDWD.

We use three metrics to evaluate the performance of the models, including *Mean Absolute Percentage Error* (MAPE) and *Root Mean Square Error* (RMSE), which are defined as

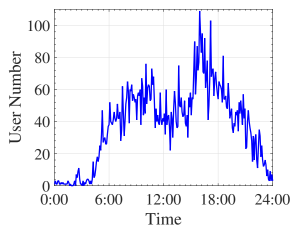
$$\begin{aligned} \text{MAPE} &= \frac{1}{T}\sum_{t=1}^{T}\frac{|\hat{x}_t - x_t|}{x_t} \times 100\%, \\ \text{RMSE} &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{x}_t - x_t)^2}, \end{aligned} \tag{17}$$

where x_t is the real value of the t-th sample in a time series, and $\hat{x}_t$ is the predicted one. The less value of the three metrics means the better performance.

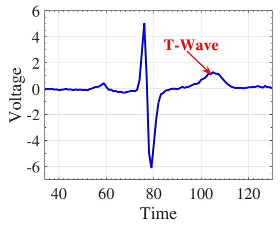
Results & Analysis. We compared the performance of the competitors in two TSF scenarios suggested in [33]. In the first scenario, we predicted the average user volumes of a base station in subsequent periods. The length of the periods was varied from 5 to 30 minutes. Fig. 4 is a comparison of the performance averaged on the 20 base stations in one week. As can be seen, while all the models experience a gradual decrease in prediction error as the period length increases, that mLSTM achieves the best performance compared with the baselines. Particularly, the performance of mLSTM is consistently better than wLSTM, which again approves the introduction of mWDN for time series forecasting.

In the second scenario, we predicted the average user volumes in 5 minutes after a given time interval varying from 0 to 30 minutes. Fig. 5 is a performance comparison between mLSTM and the baselines. Different from the tend we observed in Scenario I, the prediction errors in Fig. 5 generally increase along the x-axis for the increasing uncertainty. From Fig. 5 we can see that mLSTM again outperforms wLSTM and other baselines, which confirms the observations from Scenario I.

Summary. The above experiments demonstrate the superiority of mLSTM to the baselines. The mWDN structure adopted by mLSTM again becomes an important factor for the success.



(a) Cell-phone User Number



(b) ECG

Figure 6: Samples of time series.

5 INTERPRETATION

In this section, we highlight the unique advantage of our mWDN model: the interpretability. Since mWDN is embedded with a discrete wavelet decomposition, the outputs of the middle layers in mWDN, *i.e.*, $\mathbf{x}^l(i)$ and $\mathbf{x}^h(i)$, inherit the physical meanings of wavelet decompositions. We here take two data sets for illustration: *WuxiCellPhone* used in Sect. 4.2 and *ECGFiveDays* used in Sect. 4.1. Fig. 6(a) shows a sample of the user number series of a cell-phone base station in one day, and Fig. 6(b) exhibits an electrocardiogram (ECG) sample.

5.1 The Motivation

Fig. 7 shows the outputs of mWDN layers in the mLSTM and RCF models fed with the two samples given in Fig. 6, respectively. In Fig. 7(a), we plot the outputs of the first three layers in the mLSTM model as different sub-figures. As can be seen, from $\mathbf{x}^h(1)$ to $\mathbf{x}^l(3)$, the outputs of the middle layers correspond to the frequency components of the input series running from high to low. A similar phenomenon could be observed in Fig. 7(b), where the outputs of the first three layers in the RCF model are presented. This phenomenon again indicates that the middle layers of mWDN inherit the frequency decomposition function of wavelet. Then here comes the problem: can we evaluate quantitatively what layer or which frequency of a time series is more important to the final output of the mWDN based models? If possible, this can provide valuable interpretability to our mWDN model.

5.2 Importance Analysis

We here introduce an importance analysis method for the proposed mWDN model, which aims to quantify the importance of each middle layer to the final output of the mWDN based models.

We denote the problem of time series classification/forecasting using a neural network model as

$$p = M(\mathbf{x}), \tag{18}$$

where M denotes the neural network, $\mathbf{x}$ denotes the input series, and p is the prediction. Given a well-trained model M, if a small disturbance ε to the i-th element $x_i \in \mathbf{x}$ can cause a large change to the output p, we say M is sensitive to x_i. Therefore, the sensibility of the network M to the i-th element x_i of the input series is defined as the partial derivatives of $M(\mathbf{x})$ to x_i as follows:

$$S(x_i) = \left|\frac{\partial M(x_i)}{\partial x_i}\right| = \left|\lim_{\varepsilon \to 0} \frac{M(x_i) - M(x_i - \varepsilon)}{\varepsilon}\right|. \tag{19}$$

Obviously, $S(x_i)$ is also a function of x_i for a given model M. Given a training data set $\mathcal{X} = \{\tilde{\mathbf{x}}^1, \cdots, \tilde{\mathbf{x}}^j, \cdots, \tilde{\mathbf{x}}^J\}$ with J training samples, the importance of the i-th element of the input series $\mathbf{x}$ to the model M is defined as

$$I(x_i) = \frac{1}{J}\sum_{j=1}^{J} S(\tilde{x}_i^j), \tag{20}$$

where $\tilde{x}_i^j$ is the value of the i-th element in the j-th training sample.

The importance definition in Eq. (20) can be extended to the middle layers in the mWDN model. Denoting a as an output of a middle layer in mWDN, the neural network M can be rewritten as

$$p = M(a(\mathbf{x})), \tag{21}$$

and the sensibility of M to a is then defined as

$$S_a(\mathbf{x}) = \left|\frac{\partial M(a(\mathbf{x}))}{\partial a(\mathbf{x})}\right| = \left|\lim_{\varepsilon \to 0} \frac{M(a(\mathbf{x})) - M(a(\mathbf{x}) - \varepsilon)}{\varepsilon}\right|. \tag{22}$$

Given a training data set $\mathcal{X} = \{\tilde{\mathbf{x}}^1, \cdots, \tilde{\mathbf{x}}^j, \cdots, \tilde{\mathbf{x}}^J\}$, the importance of a *w.r.t.* M is calculated as

$$I(a) = \frac{1}{J}\sum_{j=1}^{J} S_a(\tilde{\mathbf{x}}^j). \tag{23}$$

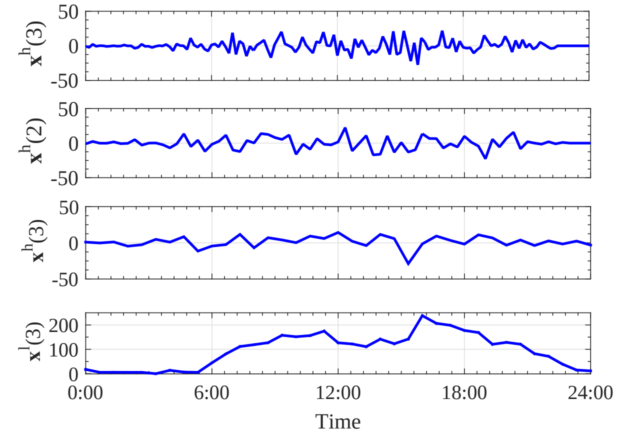
The calculation of $\frac{\partial M}{\partial \mathbf{x}_i}$ and $\frac{\partial M}{\partial a}$ in Eq. (19) and Eq. (22) are given in the Appendix for concision. Eq. (20) and Eq. (23) respectively define the importance of a time-series element and an mWDN layer to an mWDN based model.

5.3 Experimental Results

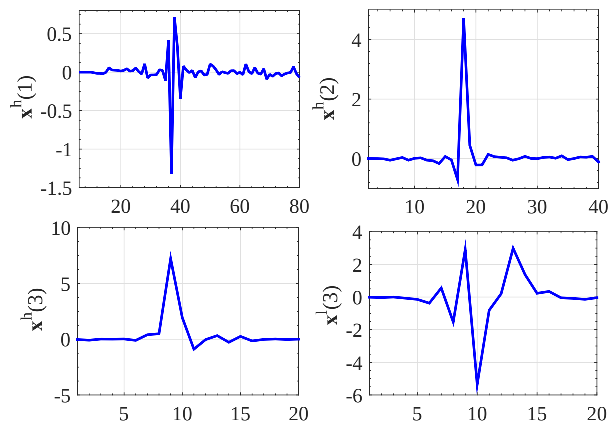
Fig. 8 and Fig. 9 shows the results of importance analysis. In Fig. 8, the mLSTM model trained on *WuxiCellPhone* in Sect. 4.2 is used. Fig. 8(b) exhibits the importance spectrum of all the elements, where the x-axis denotes the increasing timestamps and the colors in spectrum denote the varying importance of the features: the redder, the more important. From the spectrum, we can see that the latest elements are more important than the older ones, which is quite reasonable in the scenario of time series forecasting and justifies the time value of information.

Fig. 8(a) exhibits the importance spectra of the middle layers listed from top to bottom in the increasing order of frequency. Note that for the sake of comparison, we resize the lengths of the outputs to the same. From the figure, we can observe that *i*) the lower frequency layers in the top are with higher importance, and *ii*) only the layers with higher importance exhibit the time value of the elements as in Fig. 8(b). These imply that the low frequency layers in mWDN are crucially important to the success of time series forecasting. This is not difficult to understand since the information captured by low frequency layers often characterizes the essential tendency of human activities and therefore is of great use to revealing the future.

Fig. 9 depicts the importance spectra of the RCF model trained on the *ECGFiveDay* data set in Sect. 4.1. As shown in Fig. 9(b), the most important elements are located in the range from roughly 100 to 110 of the time axis, which is quite different from that in Fig. 8(b). To understand this, recall Fig. 6(b) that this range corresponds to the T-Wave of electrocardiography, covering the period of the heart relaxing and preparing for the next contraction. It is generally

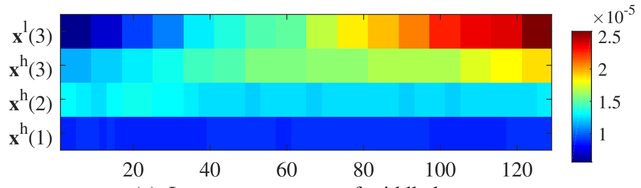


(a) Cell-phone User Numbers in Different Layers

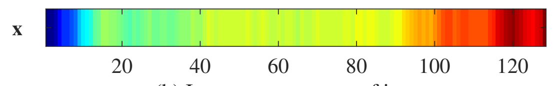


(b) ECG Waves in Different Layers

Figure 7: Sub-series generated by the mWDN model.

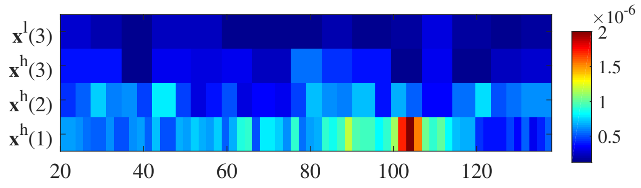


(a) Importance spectra of middle layers

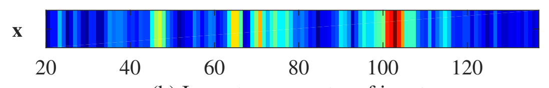


(b) Importance spectra of inputs

Figure 8: Importance spectra of mLSTM on *WuxiCellPhone*.



(a) Importance spectra of middle layers



(b) Importance spectra of inputs

Figure 9: Importance spectra of RCF on *ECGFiveDays*.

believed that abnormalities in the T-Wave can indicate seriously impaired physiological functioning [2]. As a result, the elements describing T-Wave are more important to the classification task.

Fig. 9(a) shows the importance spectra of middle layers, also listed from top to bottom in the increasing order of frequency. It is interesting that the phenomenon is opposite to the one in Fig. 8(a); that is, the layers in high frequency are more important to the classification task on *ECGFiveDays*. To understand this, we should know that the general trends of ECG curves captured by low frequency layers are very similar for everyone, whereas the abnormal fluctuations captured by high frequency layers are the real distinguishable information for heart diseases identification. This also indicates the difference between a time-series classification task and the a time-series forecasting task.

Summary. The experiments in this section demonstrate the interpretability advantage of the mWDN model stemming from the integration of wavelet decomposition and our proposed importance analysis method. It can also be regarded as an indepth exploration to solve the black box problem of deep learning.

6 RELATED WORKS

Time Series Classification (TSC). The target of TSC is to assign a time series pattern to a specific category, *e.g.*, to identify a word based on series of voice signals. Traditional TSC methods could be classified into three major categories: distance based, feature based, and ensemble methods [6]. Distance based methods predict the category of a time series by comparing the distances or similarities to other labeled series. The widely used TSC distances includes the Euclidean distance and dynamic time warping (DTW) [2], and DTW with KNN classifier has been the state-of-the-art TSC method for a long time [18]. A defect of distance based TSC methods is the relatively high computational complexity. Feature based methods overcome this defect by training classifiers on deterministic features and category labels of time series. Traditional methods, however, usually depend on handcraft features as inputs, such as symbolic aggregate approximation and interval mean/deviation/slop [8, 22]. In recent years, automatic feature engineering was introduced to TSC, such as time series shapelets mining [11], attention [27] and deep learning based representative learning [20]. Our study also falls in this area but with frequency awareness. The well-known ensemble methods for TSC include PROP [23], COTE [1], *etc.*, which

[2]https://en.m.wikipedia.org/wiki/T_wave

aim to improve classification performance via knowledge integration. As reported by some latest works [6, 38], however, existing ensemble methods are yet inferior to some distance based deep learning methods.

Time Series Forecasting (TSF). TSF refers to predicting future values of a time series using past and present data, which is widely adopted in nearly all application domains [32, 36]. A classic model is autoregressive integrated moving average (ARIMA) [3], with a great many variants, *e.g.*, ARIMA with explanatory variables (ARIMAX) [21] and seasonal ARIMA (SARIMA) [39], to meet the requirements of various applications. In recent years, a tendency of TSF research is to introduce supervised learning methods, such as support vector regression [16] and deep neural networks [41], for modeling complicated non-linear correlations between past and future states of time series. Two well-known deep neural network structures for TSF are recurrent neural networks (RNN) [5] and long short-term memory (LSTM) [10]. These indicate that an elaborate model design is crucially important for achieving excellent forecasting performance.

Frequency Analysis of Time Series. Frequency analysis of time series data has been deeply studied by the signal processing community. Many classical methods, such as Discrete Wavelet Transform [26], Discrete Fourier [12], and Z-Transform [17], have been proposed to analysis the frequency pattern of time series signals. In existing TSC/TSF applications, however, transforms are usually used as an independent step in data preprocessing [6, 24], which have no interactions with model training and therefore might not be optimized for TSC/TSF tasks from a global view. In recent years, some research works, such as Clockwork RNN [19] and SFM [15], begins to introduce the frequency analysis methodology into the deep learning framework. To our best knowledge, our study is among the very few works that embed wavelet time series transforms as a part of neural networks so as to achieve an end-to-end learning.

7 CONCLUSIONS

In this paper, we aim at building frequency-aware deep learning models for time series analysis. To this end, we first designed a novel wavelet-based network structure called mWDN for frequency learning of time series, which can then be seamlessly embedded into deep learning frameworks by making all parameters trainable. We further designed two deep learning models based on mWDN for time series classification and forecasting, respectively, and the extensive experiments on abundant real-world datasets demonstrated their superiority to state-of-the-art competitors. As a nice try for interpretable deep learning, we further propose an importance analysis method for identifying important factors for time series analysis, which in turn verifies the interpretability merit of mWDN.

REFERENCES

[1] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. 2015. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE TKDE* 27, 9 (2015), 2522–2535.

[2] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *KDD '94*, Vol. 10. Seattle, WA, 359–370.

[3] George EP Box and David A Pierce. 1970. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association* 65, 332 (1970), 1509–1526.

[4] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/.

[5] Jerome T Connor, R Douglas Martin, and Les E Atlas. 1994. Recurrent neural networks and robust time series prediction. *IEEE T NN* 5, 2 (1994), 240–254.

[6] Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* (2016).

[7] Ingrid Daubechies. 1992. *Ten lectures on wavelets.* SIAM.

[8] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. 2013. A time series forest for classification and feature extraction. *Information Sciences* 239 (2013), 142–153.

[9] Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* 3, 4 (1999), 128–135.

[10] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2002. Applying LSTM to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01.* Springer, 193–200.

[11] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2014. Learning time-series shapelets. In *KDD '14.* ACM, 392–401.

[12] Fredric J Harris. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proc. IEEE* 66, 1 (1978), 51–83.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR '16.* 770–778.

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

[15] Hao Hu and Guo-Jun Qi. 2017. State-Frequency Memory Recurrent Neural Networks. In *International Conference on Machine Learning.* 1568–1577.

[16] Young-Seon Jeong, Young-Ji Byon, Manoel Mendonca Castro-Neto, and Said M Easa. 2013. Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE T ITS* 14, 4 (2013), 1700–1707.

[17] Eliahu Ibraham Jury. 1964. Theory and Application of the z-Transform Method. (1964).

[18] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems* 7, 3 (2005), 358–386.

[19] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork rnn. In *International Conference on Machine Learning.* 1863–1871.

[20] Martin Längkvist, Lars Karlsson, and Amy Loutfi. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 42 (2014), 11–24.

[21] Sangsoo Lee and Daniel Fambro. 1999. Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board* 1678 (1999), 179–188.

[22] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *SIGMOD'03 workshop on Research issues in DMKD.* ACM, 2–11.

[23] Jason Lines and Anthony Bagnall. 2015. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery* 29, 3 (2015), 565–592.

[24] Hui Liu, Hong-qi Tian, Di-fu Pan, and Yan-fei Li. 2013. Forecasting models for wind speed using wavelet, wavelet packet, time series and Artificial Neural Networks. *Applied Energy* 107 (2013), 191–208.

[25] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE T ITS* 16, 2 (2015), 865–873.

[26] Stephane G Mallat. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE T PAMI* 11, 7 (1989), 674–693.

[27] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971* (2017).

[28] Pranav Rajpurkar, Awni Y Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y Ng. 2017. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *arXiv preprint arXiv:1707.01836* (2017).

[29] Alistair CH Rowe and Paul C Abbott. 1995. Daubechies wavelets and mathematica. *Computers in Physics* 9, 6 (1995), 635–648.

[30] Xin Song, Yuanxin Ouyang, Bowen Du, Jingyuan Wang, and Zhang Xiong. 2017. Recovering Individualąŕs Commute Routes Based on Mobile Phone Data. *Mobile Information Systems,2017,(2017-02-9)* 2017, 18 (2017), 1–11.

[31] Jingyuan Wang, Chao Chen, Junjie Wu, and Zhang Xiong. 2017. No Longer Sleeping with a Bomb: A Duet System for Protecting Urban Safety from Dangerous Goods. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1673–1681.

[32] Jingyuan Wang, Fei Gao, Peng Cui, Chao Li, and Zhang Xiong. 2014. Discovering urban spatio-temporal structure from time-evolving traffic networks. In *Proceedings of the 16th Asia-Pacific Web Conference.* Springer International Publishing, 93–104.

[33] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic speed prediction and congestion source exploration: A deep learning method. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on.* IEEE, 499–508.

[34] Jingyuan Wang, Xu He, Ze Wang, Junjie Wu Wu, Nicholas Jing Yuan, Xing Xie, and Zhang Xiong. 2018. CD-CNN: A Partially Supervised Cross-Domain Deep Learning Model for Urban Resident Recognition. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence.*

[35] Jingyuan Wang, Yating Lin, Junjie Wu, Zhong Wang, and Zhang Xiong. 2017. Coupling Implicit and Explicit Knowledge for Customer Volume Prediction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence.* 1569–1575.

[36] Jingyuan Wang, Yu Mao, Jing Li, Zhang Xiong, and Wen-Xu Wang. 2014. Predictability of road traffic and congestion in urban areas. *Plos One* 10, 4 (2014), e0121825.

[37] Liwei Wang, Chen-Yu Lee, Zhuowen Tu, and Svetlana Lazebnik. 2015. Training deeper convolutional networks with deep supervision. *arXiv preprint arXiv:1505.02496* (2015).

[38] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *IJCNN '17.* IEEE, 1578–1585.

[39] Billy M Williams and Lester A Hoel. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering* 129, 6 (2003), 664–672.

[40] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1, 2 (1989), 270–280.

[41] G Peter Zhang. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50 (2003), 159–175.

[42] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. 2016. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Frontiers of Computer Science* 10, 1 (2016), 96–112.

APPENDIX

In a neural network model, the outputs of the layer l are connected as the inputs of the layer $l+1$. According to the chain rule, the partial derivative of the model M to middle layer outputs could be calculated layer-by-layer as

$$\frac{\partial M}{\partial a_i^{(l)}} = \sum_j \frac{\partial M}{\partial a_j^{(l+1)}} \frac{\partial a_j^{(l+1)}}{\partial a_i^{(l)}}, \tag{24}$$

where $a_i^{(l)}$ is the i-th output of the layer l. The proposed models contain types of layers: the convolutional, LSTM and fully connected layers, which are discussed below.

For convolutional layers, only 1D convolutional operation is used in our cases. The output of the layer l is a matrix with the size of $L \times 1 \times C$, which is connected to neural matrix of the $l+1$-th with a convolutional kernel in the size of $k \times 1 \times C$. The partial derivative of M to the i^{th} output of the layer l is calculated as

$$\begin{aligned}\frac{\partial M}{\partial a_i^{(l)}} &= \sum_{n=0}^{k-1} \frac{\partial M}{\partial a_{i-n}^{(l+1)}} \frac{\partial a_{i-n}^{(l+1)}}{\partial a_i^{(l)}} \\ &= \sum_{n=0}^{k-1} \delta_{i-n}^{(l+1)} w_n^{(l+1)} f'\left(a_i^{(l)}\right),\end{aligned}$$

where w_n denotes the n-th element of the convolutional kernel, $\delta_i^{(l)} = \frac{\partial M}{\partial a_i^{(l)}}$, and $f'\left(a_i^{(l)}\right)$ is the derivative of activation function.

For LSTM laysers, we denote the output of a LSTM unit in layer $l+1$ at time t as

$$a_i^{t,(l+1)} = f\left(b^{t,(l)}\right),$$

where $b^t(l)$ is calculated as

$$b^{t,(l)} = \sum_i w_i^a a_i^{t,(l)} + \sum_i w_i^b b_i^{t-1,(l)} + \sum_i w_i^s s_i^{t-1,(l)}.$$

$s_i^{t-1,(l)}$ is the history state that is saved in the memory cell. Therefore, the partial derivative of M to the $a_i^{t,(l)}$ is calculated as

$$\frac{\partial M}{\partial a_i^{(l)}} = \sum_t \frac{\partial M}{\partial b^{t,(l)}} \frac{\partial b^{t,(l)}}{\partial a_i^{t,(l)}} = \sum_t \delta_i^{t,(l+1)} f'(b^{t,(l)}) \theta_i^{t,(l)},$$

where $\theta_i^{t,(l)}$ is an equation as

$$\theta_i^{t,(l)} = \left(w_i^a + w_i^b \frac{\partial b^{t+1,(l)}}{\partial a_i^{t+1,(l)}} + w_i^s \frac{\partial s^{t+1,(l)}}{\partial a_i^{t+1,(l)}} \right).$$

The derivative $\frac{\partial s^{t,(l)}}{\partial a_i^{t,(l)}}$ in the above equation is calculated as

$$\frac{\partial s^{t,(l)}}{\partial a_i^{t,(l)}} = s^{t-1,(l)} \frac{\partial b^{t,(l)}}{\partial a_i^{t,(l)}} + \frac{\partial b^{t,(l)}}{\partial a_i^{t,(l)}} f(a_i^{t,(l)}) + b^{t,(l)} f'(a_i^{t,(l)}).$$

For fully connect layers, the output $a_i^{(l)} = f(w_i a_i^{(l-1)} + b)$. Then the partial derivative is equal to

$$w_i f'(w_i a_i^{(l-1)} + b).$$