Pattern Recognition 155 (2024) 110670

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

IRCNN: A novel signal decomposition approach based on iterative residue convolutional neural network

Feng Zhou [a,*], Antonio Cicone [b,c,d], Haomin Zhou [e]

[a] School of Information Sciences, Guangdong University of Finance and Economics, Guangzhou, 510320, China
[b] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, 67100, Italy
[c] Istituto di Astrofisica e Planetologia Spaziali, INAF, Rome, 00133, Italy
[d] Istituto Nazionale di Geofisica e Vulcanologia, Rome, 00143, Italy
[e] School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, United States

ARTICLE INFO

Keywords:
Empirical mode decomposition
Adaptive signal decomposition
Signal local average
Convolutional neural network
Residual network

ABSTRACT

The decomposition of non-stationary signals is an important and challenging task in the field of signal time–frequency analysis. In the recent two decades, many decomposition methods have been proposed, inspired by the empirical mode decomposition method, first published by Huang et al. in 1998. However, they still have some limitations. For example, they are generally prone to boundary and mode mixing effects and are not very robust to noise. Inspired by the successful applications of deep learning, and given the lack in the literature of works in which deep learning techniques are used directly to decompose non-stationary signals into simple oscillatory components, we use the convolutional neural network, residual structure and nonlinear activation function to compute in an innovative way the local average of the signal, and study a new non-stationary signal decomposition method under the framework of deep learning. We discuss the training process of the proposed model and study the convergence analysis of the learning algorithm. In the experiments, we evaluate the performance of the proposed model from two points of view: the calculation of the local average and the signal decomposition. Furthermore, we study the mode mixing, noise interference, and orthogonality properties of the decomposed components produced by the proposed method, and compare it with the state-of-the-art ones. All results show that the proposed model allows for better handling boundary effect, mode mixing effect, robustness, and the orthogonality of the decomposed components than existing methods.

1. Introduction

With the development of technology, many real-life signals that exhibit nonlinearity and non-stationarity, such as human speech, radar systems, and seismic waves, can be accurately captured. It is well known that decomposing and exploring the features of this kind of signal is quite challenging due to their nonlinear and non-stationary characteristics.

In the past two decades, many studies have emerged for processing non-stationary signals. One of the most representative works is the empirical mode decomposition (EMD) algorithm along with the Hilbert spectrum analysis proposed by Huang et al. in 1998 [1]. Because EMD is fully data-driven, and can adaptively decompose a signal into several intrinsic mode functions (IMFs), it has already shown its usefulness in a wide range of applications, including semantic recognition, alcoholism identification [2], and stock trend prediction. Despite its remarkable success, it still lacks mathematical foundations and is sensitive to noise and sampling. This sparked many efforts to improve the EMD. The improvements share the same feature: a signal is decomposed into several simpler components, and then a time–frequency analysis method is applied to each component separately. These signal decomposition methods can be mainly achieved in two ways: by iteration or by optimization.

Methods based on iteration include many techniques, such as moving average, partial differential equation (PDE) and filter. For instance, Smith presented a new iteration method, based on the local average, to decompose the non-stationary signals into a set of functions [3]. Deléchelle et al. proposed a new approach that resolves one major problem in the EMD, that is, the mean envelope detection of a signal, in virtue of a parabolic PDE [4]. Hadji et al. used the differential calculus on envelopes, which makes them prove that iterations of the sifting process are well approximated by the resolution of PDE [5]. Hong et al. introduced a novel sifting method based on the concept of the local integral mean of a signal [6]. And Cicone et al. studied the method based on iterative filtering (IF) to compute the local average, which

* Corresponding author.
E-mail addresses: fengzhou@gdufe.edu.cn (F. Zhou), antonio.cicone@univaq.it (A. Cicone), hmzhou@math.gatech.edu (H. Zhou).

https://doi.org/10.1016/j.patcog.2024.110670
Received 22 October 2022; Received in revised form 29 May 2024; Accepted 9 June 2024
Available online 12 June 2024

is utilized to replace the mean of the upper and lower envelopes in the sifting procedure of the EMD [7]. Tu et al. proposed the iterative nonlinear chirp mode decomposition (INCMD) [8] under the framework of the variational nonlinear chirp mode decomposition. By extending the related concepts such as extreme point and intrinsic mode function to $\mathbb{R}^2$, G. Xu et al. extended the EMD algorithm to decompose image signals [9].

On the other hand, there are methods based on optimization. Peng et al. designed an adaptive local linear operator-based optimization model to decompose a signal into several local narrow band signals [10]. Oberlin et al. proposed an optimization model in computing the mean envelope to replace the original one in EMD [11]. Inspired by the compressed sensing theory, Hou et al. studied a new adaptive data analysis method, which can be seen as a nonlinear version of compressed sensing and provides a mathematical foundation of the EMD method [12]. Flandrin et al. proposed a convex optimization procedure in order to replace the sifting process in the EMD, which follows the idea of texture-geometry decomposition with further specific EMD features such as quasi-orthogonality and extrema-based constraints [13,14]. Dragomiretskiy et al. put forward the variational mode decomposition (VMD), whose goal is to decompose a signal into a discrete number of modes, that have specific sparsity properties while reproducing the input [15]. Rehman et al. generalized the VMD method to multivariate or multichannel data [16]. And Zhou et al. presented a new mathematical framework by finding the local average based on the local variational optimization model [17].

In addition, there are some methods that cannot be classified into the above two categories. For instance, Daubechies et al. proposed the method, called synchrosqueezed wavelet transforms, by combining the wavelet analysis and reallocation method [18]. Gille presented the approach, called empirical wavelet transform (EWT), to build adaptive wavelets [19], whose main idea is to extract the different modes by designing an appropriate wavelet filter bank. Singh et al. studied the adaptive Fourier decomposition method (FDM) based on the Fourier theory, which decomposes any data into a small number of "Fourier intrinsic band functions" [20,21]. And Wang et. extended the adaptive FDM to the multi-channel case [22].

According to the works described above, we find that whether the method is based on iterative or optimization, calculating the local average of a given signal is very critical. For example, in EMD [1], the mean of the upper and lower envelopes are used to measure the local average of the signal; the local variational optimization model is constructed to compute the local average in [17]; and in the iterative filtering method [7], the low-pass filter is employed to find the local average. Although there exist many studies on the characterization of the local average, it is basically impossible to find a method suitable for all signals from a practical point of view. This makes the existing methods prone to several problems including boundary effect and mode mixing. Fig. 1 shows an example of these two issues produced by the EMD method. It is easy to observe that the boundary effect tends to spread the errors from the end point to the inner region to interfere with other components, and the mode mixing affects other components due to the errors in one component. It is clearly important to eliminate, or at least minimize the impact of these two effects. Discussing the local average customized according to the type of signal, it not only provides a new research perspective, but also is likely to become the trend in the near future in signal processing for non-stationary data.

In recent years, thanks to the remarkable results obtained in fields of research like image and natural language processing, the usage and application of deep learning methods have spread widely in an ample variety of research fields, like image processing [23] and natural language processing [24]. These different types of models are studied and proposed mainly because of the differences in the types of data or some particularities of data, such as data scarcity [25]. In signal processing, deep learning models have been used, so far, to achieve various goals, such as: noise removal [26], forecasting [27,28], nonlinear control system [29,30], and detection [31]. However, to the best of our knowledge, not a single method has been proposed so far in the literature, which allows to decompose a given non-stationary signal into simple oscillatory components, like the IMFs, which is solely based on deep learning techniques.

Deep learning allows studying local averages and non-stationary signal decompositions of different types of signals. It is noted that the IF method, which is a typical signal decomposition algorithm, adopts the convolution (or filtering) operation to approximate the local average of the signal. However, the convolutional kernel needs to be given in advance, which makes IF not flexible. Inspired by the personalized expression ability of deep learning and the local average method adopted by IF, this paper applies 1-dimensional (1-D) convolutional neural network (CNN) to improve the local average characterization of IF under the framework of deep learning, and proposes a new deep-learning-based signal decomposition method, named iterative residual convolutional neural network (IRCNN). Compared to IF, deep learning makes it possible for the proposed IRCNN model to depict the local average of non-stationary signals more flexibly, thus avoiding boundary effects and mode mixing issues. Overall, the contributions and limitations of the IRCNN model can be summarized as follows:

(i) To our knowledge, IRCNN is the first deep learning-based approach in the field of non-stationary signal decomposition. The introduction of deep learning will add many unique and effective tools to deal with the non-stationary signal decomposition problem, such as the nonlinear activation function, the residue network structure, and the customized loss function.

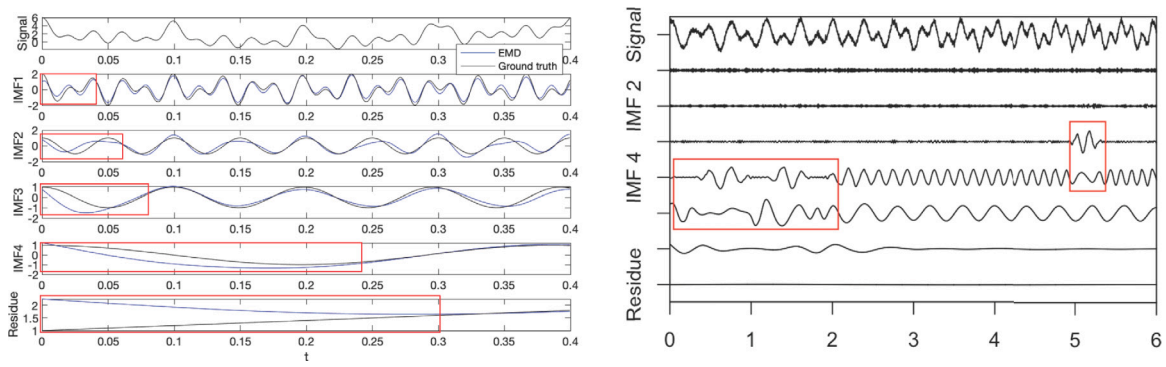
(ii) Unlike the moving average method and the filter operation in the iterative filtering method, the convolutional kernel weights that appear in the proposed IRCNN model, are not fixed in advance, but are learnt adaptively in the training phase according to the inputs. This makes IRCNN more flexible and adaptive in finding the local average and achieving the decomposition for a given signal.

(iii) Several artificial datasets are constructed to verify the performance IRCNN in terms of local average characterization, noise interference, mode mixing, and orthogonality. Furthermore, we compare IRCNN with the state-of-the-art methods. In addition, we also use the solution of the Duffing and Lorenz equations, and the real data including the length of day (LOD) and the mean relative humidity (MRH) to evaluate the approximation ability of IRCNN to the existing models.

(iv) Generally speaking, IRCNN takes a certain amount of time in the training phase, which is a commonality of deep learning-based models. However, once the model training is completed, the computational efficiency in the prediction stage is relatively fast, especially it can use the parallelization mechanism to predict multiple signals at the same time, which is not available in most existing methods.

(v) IRCNN is essentially a supervised deep learning model, so it inherits some of the limitations of supervised models, such as, in the training phase, each input signal needs to have a label associated in advance.

The rest of the paper is organized as follows. We review the IF method and provide its algorithm in Section 2.1. And the concept of β-smooth function and its properties are given in Section 2.2, which are used for proving the convergence of the proposed model. In Section 3, the new local average method and the derived signal decomposition method, collectively called the IRCNN, are proposed. Moreover, the training process and convergence analysis of IRCNN are given in this section. In Section 4, we study a series of examples to evaluate the performance of IRCNN compared with the existing methods. Finally, we give the conclusion in Section 5.



(a) Boundary effect (b) Mode mixing effect

Fig. 1. Graphic illustration of the boundary effect and mode mixing issue in two EMD decompositions.

2. IF and β-smooth function

2.1. IF

The iterative filtering (IF) [7] is a recurrent algorithm that decomposes a nonlinear and non-stationary signal into a number of IMFs. The main idea of IF is the subtraction of local moving averages from the signal iteratively, where the local moving averages are calculated through convolutions with low-pass filters. Alg. 1 shows the detailed steps, where the parameter l_n, called the filter length, is important in the IF method, and is determined by the information contained in the signal itself; $w_n(\cdot)$ represents the low-pass filter function.

Algorithm 1: Iterative filtering (IF)

Data: Given a signal $x(t)$
Result: IMF
while *the number of extrema of* $x \geq 2$ **do**
  $n = 0$;
  $x_1(t) = x(t)$;
  while *the stopping criterion is not satisfied* **do**
    compute the filter length l_n and filter weight function w_n for x_n;
    $x_{n+1}(t) = x_n(t) - \int_{-l_n}^{l_n} x_n(t+y)w_n(y)dy$;
    $n = n + 1$;
  end
  $IMF = IMF \cup \{x_n\}$;
  $x(t) = x(t) - x_n(t)$;
end
$IMF = IMF \cup \{x\}$.

2.2. β-Smooth function and some of its properties

We first introduce the concepts of $\mathcal{L}$-Lipschitz continuous and β-smooth for a function from [32].

Definition 2.1. A function f is said to be $\mathcal{L}$-Lipschitz continuous if for all $x, y \in \mathcal{X}$, $\|f(x) - f(y)\| \leq L\|x - y\|$, where $\mathcal{X}$ denotes the convex domain of f, and L is called the Lipschitz constant.

Definition 2.2. A continuously differentiable function f is β-smooth if the gradient ∇f is β-Lipschitz, that is if for all $x, y \in \mathcal{X}$, $\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|$, where $\mathcal{X}$ is the convex domain of f.

Then, for a unconstraint optimization problem, if its objective function is β-smooth, we can prove that the sequence generated by the gradient descent algorithm converges to a stationary point when the learning rate is small enough. The details can be found in Theorem 2.1.

Theorem 2.1. *Let f be a β-smooth function and $f^* = \min f(x) > -\infty$. Then the gradient descent algorithm with a constant learning rate $\lambda < \frac{2}{\beta}$, i.e., $x^{(k+1)} = x^{(k)} - \lambda\nabla f(x^{(k)})$, converges to a stationary point, i.e., the set $\{x : \nabla f(x) = \mathbf{0}\}$.*

Proof. According to the gradient descent algorithm, i,e.,

$$x^{(k+1)} = x^{(k)} - \lambda\nabla f(x^{(k)}), \tag{1}$$

as f is β-smooth, we have

$$\begin{aligned} f(x^{(k+1)}) &\overset{(a)}{\leq} f(x^{(k)}) + \nabla f(x^{(k)})(x^{(k+1)} - x^{(k)}) + \frac{\beta}{2}\|x^{(k+1)} - x^{(k)}\|^2 \\ &\overset{(b)}{=} f(x^{(k)}) - \lambda\|\nabla f(x^{(k)})\|^2 + \frac{\beta\lambda^2}{2}\|\nabla f(x^{(k)})\|^2 \\ &= f(x^{(k)}) - \lambda(1 - \frac{\beta\lambda}{2})\|\nabla f(x^{(k)})\|^2, \end{aligned}$$

where the inequality (a) follows from Lemma 3.4 in [32], and the equality (b) is obtained from Eq. (1). Due to $\lambda < 2/\beta$, it becomes

$$\|\nabla f(x^{(k)})\|^2 \leq \frac{f(x^{(k)}) - f(x^{(k+1)})}{\lambda(1 - \frac{\beta\lambda}{2})}.$$

Next, we have

$$\begin{aligned} \sum_{k=0}^{K}\|\nabla f(x^{(k)})\|^2 &\leq \frac{1}{\lambda(1 - \frac{\beta\lambda}{2})}\sum_{k=0}^{K}(f(x^{(k)}) - f(x^{(k+1)})) \\ &= \frac{f(x^{(0)}) - f(x^{(K+1)})}{\lambda(1 - \frac{\beta\lambda}{2})} \leq \frac{f(x^{(0)}) - f(x^*)}{\lambda(1 - \frac{\beta\lambda}{2})}, \end{aligned}$$

where x^* denotes the global optimization point. Taking the limit as $K \to +\infty$, we have $\sum_{k=0}^{+\infty}\|\nabla f(x^{(k)})\|^2 \leq +\infty$. Hence, $\lim_{k\to+\infty}\nabla f(x^{(k)}) = 0$ is obtained.

3. IRCNN inner loop block and IRCNN

3.1. IRCNN inner loop block

The main operation in IF is the computation of moving average, which is essentially realized by the convolution operation, where the filter length depends on the given signal, and the filter weights are mainly given by some empirical functions selected artificially a priori. Therefore, it is very natural to convert the convolution operation into a 1-D CNN model, where both the kernel length and the kernel weights can be learnt adaptively according to the input signals given in advance. Furthermore, some ingenious mechanisms in deep learning, such as the nonlinear activation function, the residue learning [33], etc., can be adopted to make it more flexible. The structure we design to mimic the inner "while" loop of Alg. 1, is graphically depicted in Fig. 2. Since it mainly contains the iterative mechanism, the convolutional layer and the subtraction operation, we call it the iterative residual convolutional neural network (IRCNN) inner loop block.

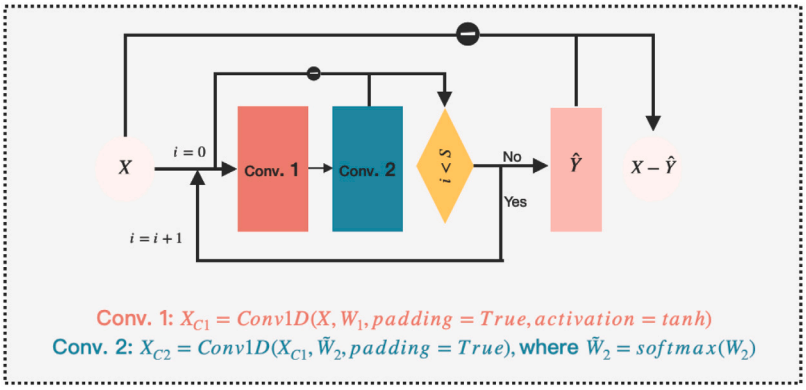


Fig. 2. Graphic illustration of the IRCNN inner loop block.

Algorithm 2: IRCNN inner loop block

Data: $X \in \mathbb{R}^N$
Result: the local average of X
Initialize $i = 0$ and $X^{(0)} = X$;
while $i < S$ **do**
  The input $X^{(i)}$ goes through the first 1-D convolutional layer, i.e., $X_{C1} := Conv1D(X^{(i)}, W_1^{(i)}, padding = True, activation = \tanh)$;
  Transfer X_{C1} to the second convolutional layer, i.e., $X_{C2} := Conv1D\ (X_{C1}, \tilde{W}_2^{(i)}, padding = True)$, where $\tilde{W}_2^{(i)} := softmax(W_2^{(i)})$;
  $X^{(i+1)} = X^{(i)} - X_{C2}$;
  $i = i + 1$;
end
$\hat{Y} = X^{(S)}$ and $X - \hat{Y}$ are the IMF and the local average respectively.

As shown in Fig. 2, the inner loop mainly consists of a judgment-loop structure and a residue operation, and the judgment-loop structure is formed of two convolutional layers and a residual operation. Suppose $X \in \mathbb{R}^N$ (the vectors in this article are column vectors by default unless otherwise specified.) denote the input, the output of the IRCNN inner loop block, called $\hat{Y} \in \mathbb{R}^N$, is computed as the following Alg. 2. And $X - \hat{Y}$ is the local average of the input signal obtained by the IRCNN inner loop block.

Mathematically, the output of the IRCNN inner loop block can be expressed as: $\hat{Y} = F(X, \mathbf{W})$, where $\mathbf{W}$ denotes the undetermined weights in the IRCNN inner loop block, and the function F represents the structure of IRCNN inner loop block, which is composed of the operators including convolution, nonlinear activation and subtraction. The detailed process of F can be formulated as: $F(X, \mathbf{W}) = f(X^{(S-1)}, \mathbf{W}^{(S-1)})$, where S represents the number of iteration in the IRCNN inner loop block, $\mathbf{W}^{(S-1)}$ is the undetermined weights in the $(S-1)$-th iteration, and the function f and $X^{(S-1)}$ are defined as:

$$\begin{cases} f(X^{(i)}, \mathbf{W}^{(i)}) = \tanh(X^{(i)} * W_1^{(i)}) * \tilde{W}_2^{(i)}, \\ X^{(i+1)} = X^{(i)} - f(X^{(i)}, \mathbf{W}^{(i)}), \end{cases} \tag{2}$$

where $X^{(0)} = X$, $\mathbf{W}^{(i)}$ is the undetermined weights in the ith iteration that it includes the weights, denoted as $W_1^{(i)} \in \mathbb{R}^{K_1}$ and $W_2^{(i)} \in \mathbb{R}^{K_2}$, $\tilde{W}_2^{(i)} := softmax(W_2^{(i)}) = \left\{ \frac{exp(W_{2\_l}^{(i)})}{\sum_k exp(W_{2\_k}^{(i)})} \right\}_{l=1}^{K_2}$, $*$ is the 1-D convolution operation, and $i = 0, 1, \ldots, S-2$.

It is worth pointing out that: (1) S, K_1 and K_2 are the hyper-parameters of IRCNN inner loop block. Their values can vary in different iterations, depending on the performance of the validation dataset during training. (2) the roles of the two 1-D convolutional layers in each judgment-loop are different. The role of the first convolutional layer, which is configured with a non-linear activation function (we select tanh in this work), is to enhance the nonlinear expression ability of the method. Whereas, the purpose of the second convolutional layer is to make the result more reasonable to describe the local average of the signal. Therefore, the non-negativity and normalization restrictions of its weights are added; and there is no nonlinear activation function configured with it. The use of padding in the two layers is to ensure that the length of the output is consistent with the input. (3) In the existing methods, it is difficult to overcome the problems like "boundary effect" and "mode mixing effect", one of the main reasons is that they generally lack locality, resulting local errors at any given point that will easily propagate in the sifting process. In IRCNN, the weights of the convolutional kernels are adaptive learnt according to the information of each position of the training data, which makes IRCNN to have a stronger locality than the existing methods.

3.2. IRCNN

After the IRCNN inner loop block for the identification of the local average of a given signal is constructed, we can cascade a finite number of IRCNN inner loop blocks together to derive the signal decomposition, which is called IRCNN, and is shown in Fig. 3. According to it, an input signal $X \in \mathbb{R}^N$ (also denoted as X_0) can be decomposed into M IMFs $\hat{\mathbf{Y}} := \{\hat{Y}_i\}_{m=1}^M$ (each $\hat{Y}_i \in \mathbb{R}^N$) and a residue $X_M \in \mathbb{R}^N$. In the execution process, the value of M needs to be given in advance, its value is generally taken to be the largest number of true IMFs contained by the signal in the training dataset. The detailed steps of IRCNN are listed in Alg. 3.

The output of IRCNN can be formulated as:

$$\begin{cases} \hat{Y}_m = F(X_{m-1}, \mathbf{W}_m), \\ X_m = X_{m-1} - \hat{Y}_m, \end{cases} \tag{3}$$

where $m = 1, 2, \ldots, M$, $X_0 = X$, $F(X_{m-1}, \mathbf{W}_m)$ is the mth IRCNN inner loop block whose purpose is to extract an IMF from X_{m-1}, and $\mathbf{W}_m$ denotes the undetermined weights of the mth IRCNN inner loop block.

All the generated IMFs are concatenated as the outputs of the IRCNN model. The errors between the outputs $\hat{\mathbf{Y}}$, and the labels $\mathbf{Y} \in \mathbb{R}^{N \times M}$ that are composed of the M true IMFs (if some signals do not have M true IMFs, they are filled with zero vectors), are computed by the loss function. For example, the loss function can be expressed as:

$$L(\hat{\mathbf{Y}}, \mathbf{Y}) = \|\hat{\mathbf{Y}} - \mathbf{Y}\|_F^2, \tag{4}$$

where the errors are measured by mean square error (MSE), and $\|\cdot\|_F$ denotes the Frobenius norm. In the IRCNN model equipped with the loss function as in Eq. (4), the computational complexity of the forward process of IRCNN is mainly attributed to the computation of

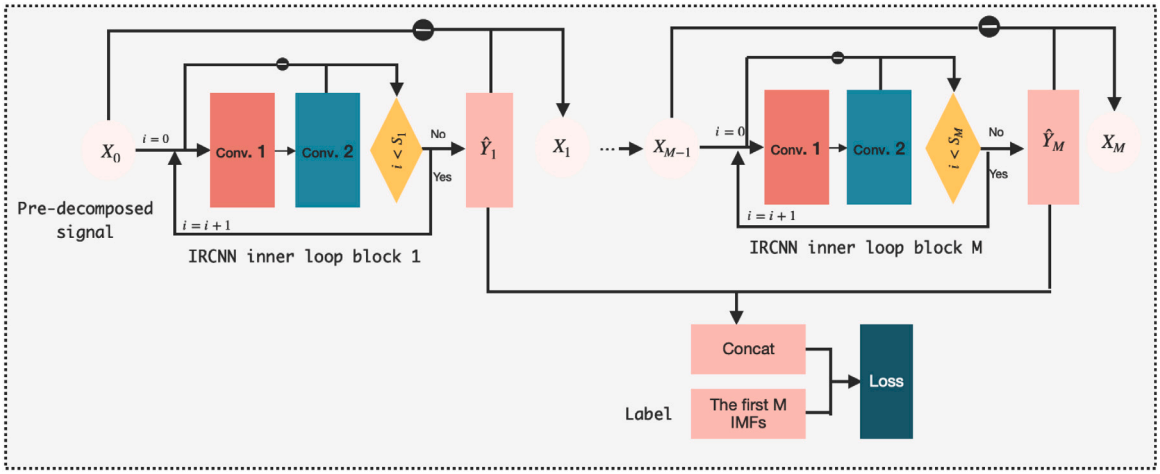


Fig. 3. Graphic illustration of the IRCNN.

Algorithm 3: IRCNN

Data: $X \in \mathbb{R}^N$, and the number of IMFs M
Result: the IMFs and residue of X
Initialize $m = 1$, $X_0 = X$;
while $m \leq M$ **do**
  Compute the m-th IMF and the local average for the input X_{m-1}, denoted as $\hat{Y}_m$ and X_m respectively, according to the IRCNN inner loop block;
  $m = m + 1$;
end
$\hat{\mathbf{Y}} = \{\hat{Y}_m\}_{m=1}^M$ and X_M are the resulting IMFs and residue of IRCNN.

the convolutional layer, which is $O(N \cdot K)$, where N and K denote the length of the signal and the size of the convolutional filter, respectively.

The loss can be customized according to the characteristic of the decomposition task. For example, if the 3rd IMF are smooth, the quadratic total variation term, expressed as

$$QTV(\hat{\mathbf{Y}}_{\Omega_1}) := \sum_{m\in\Omega_1}\sum_{t=1}^{N-1}(\hat{Y}_{(t+1),m} - \hat{Y}_{t,m})^2,$$

can be added to the loss function, where Ω_1 represents the set of subscripts of those smooth components (here $\Omega_1 = \{3\}$), i.e.,

$$L(\hat{\mathbf{Y}}, \mathbf{Y}) = \|\hat{\mathbf{Y}} - \mathbf{Y}\|_F^2 + \eta QTV(\hat{\mathbf{Y}}_{\Omega_1}), \tag{5}$$

where $\eta \geq 0$ is a penalty parameter, $\hat{\mathbf{Y}}$ is the dependent variable of the function $F(\cdot,\cdot)$, and its independent variables are X and $\mathbf{W}$ respectively.

Moreover, if the 2nd and 3rd IMFs are orthogonal, an orthogonal constraint can be added to the loss function to ensure the orthogonality of the resulting components, i.e.,

$$\begin{aligned} L(\hat{\mathbf{Y}}, \mathbf{Y}) = &\sum_{i\in\hat{\Omega}_2^c}\|\hat{Y}_i - Y_i\|_2^2 + \sum_{(i,j)\in\Omega_2}\|\mathbf{W}^{o_{ij}}\hat{Y}_i - Y_i\|_2^2 + \|\mathbf{W}^{o_{ij}}\hat{Y}_j - Y_j\|_2^2, \\ &\text{s.t. } \mathbf{W}^{o_{ij}}\mathbf{W}^{o_{ij}\top} = \mathbf{I}, \end{aligned} \tag{6}$$

where $\mathbf{W}^{o_{ij}} \in \mathbb{R}^{N\times N}$ stands for the orthogonal matrix to be determined by $\min_{\{\mathbf{W},\mathbf{W}^{o_{ij}}\}} L(\hat{Y}, Y)$, Ω_2 (here $\Omega_2 = \{(2,3)\}$) denotes the subscript pairs of those orthogonal components, $\hat{\Omega}_2$ (here $\hat{\Omega}_2 = \{2,3\}$) represents the set consisting of all subscripts that appear in Ω_2, and $\hat{\Omega}_2^c = \{1,2,\ldots,M\} - \hat{\Omega}_2$. In specific execution process, the orthogonal transformation $\mathbf{W}^{o_{ij}}$ of $\hat{Y}_i$ and $\hat{Y}_j$ can be regarded as adding a fully connected layer after the outputs of $\hat{Y}_i$ and $\hat{Y}_j$. The two fully connected layers share weights, i.e., $\mathbf{W}^{o_{ij}}$, and satisfy orthogonality, i.e., $\mathbf{W}^{o_{ij}}\mathbf{W}^{o_{ij}\top} = \mathbf{I}$. In this case, the result of any IMF whose subscript meeting $i \in \hat{\Omega}_2$ is updated from $\hat{Y}_i$ to $\mathbf{W}^{o_{ij}}\hat{Y}_i$, and the results of other components remain unchanged.

Compared with IF, IRCNN also contains two loops. The outer loop successively acquires M IMFs and one residue. And the purpose of the inner loop, i.e., the IRCNN inner loop block, is mainly to compute the local average by several iterations, and finally get the IMF and local average of the current input signal. On the other hand, these two methods have important differences, which can be summed up as:

(i) In the IF method, the filter length that is a key parameter, is determined only by the signal under decomposition. However, its filter weights lack of adaptability, and they are basically determined by the filter length. For the IRCNN approach, the kernel length in each convolutional layer is adaptive, and can be selected by hyper-parameter optimization. Moreover, its kernel weights are data-driven, which makes IRCNN flexible in the characterization of the local average.
(ii) In addition to more flexibly computing the local average of the signal, the proposed IRCNN method has all the advantages of deep learning, such as: the non-linear expression ability, the customized loss function according to specific decomposition tasks, etc., which are missing in the traditional signal decomposition methods.
(iii) Although the proposed IRCNN method is developed based on the IF method, its ambition is not to replace IF, or even to replace any existing signal decomposition method. IRCNN is essentially a supervised deep learning model, so it has the limitations of the supervised learning model. Such as the ground truth of each training sample needs to be assigned before the training phase. Compared with other methods, the generalization to handling real signals is a challenge for IRCNN in its current form.

3.3. Training process and convergence analysis of IRCNN

In the training process of IRCNN, the gradient-based back-propagation method is used to learn the kernel weights that appear in the convolutional layers by using the training data. Since when the parameter η in Eq. (5) is equal to 0, it degenerates to the case of Eq. (4), we only discuss the training processes and convergences of the models when their loss functions are given in Eqs. (5) and (6), respectively.

We first discuss the situation of the IRCNN model equipped with the loss function (5). For convenience, we consider that the output of the model only produces one IMF, and the number of iteration is also limited to 1, that is, $M = 1$ and $S_1 = 1$ in Fig. 3. Suppose that $(X^j, Y^j)_{j=1}^J \subset \mathbb{R}^N \times \mathbb{R}^N$ is a given set of training samples, where J

$$\nabla_{W_h} L = (\frac{\partial L}{\partial W_{h1}}, \dots, \frac{\partial L}{\partial W_{hK_1}})^\top,$$

$$\frac{\partial L}{\partial W_{2i}} = -\sum_{j=1}^{J}\sum_{t,l=1}^{N} \frac{\partial L}{\partial \hat{Y}_t^j} \frac{\partial X_{C2t}^j}{\partial \tilde{W}_{2l}} \frac{\partial \tilde{W}_{2l}}{\partial W_{2i}},$$

$$\frac{\partial L}{\partial W_{1i}} = -\sum_{j=1}^{J}\sum_{t,l=1}^{N} \frac{\partial L}{\partial \hat{Y}_t^j} \frac{\partial X_{C2t}^j}{\partial X_{C1l}^j} \sigma'(R_l^j) \frac{\partial R_l^j}{\partial W_{1i}},$$

$$\frac{\partial L}{\partial \hat{Y}_t^j} = 2(\hat{Y}_t^j - Y_t^j), \frac{\partial X_{C2t}^j}{\partial \tilde{W}_{2l}} = X^j_{C1(t-\lfloor K_1/2\rfloor+l)},$$

$$R_l^j = (X^j * W_1)_l, \frac{\partial R_l^j}{\partial W_{1i}} = X^j{}_{l-\lfloor K_2/2\rfloor+i},$$

$$\frac{\partial \tilde{W}_{2l}}{\partial W_{2i}} = \begin{cases} \frac{exp(W_{2i}+W_{2l})}{(\sum_k exp(W_{2k}))^2}, & \text{if } l \neq i; \\ \frac{exp(W_{2i})\sum_{k\neq i} exp(W_{2k})}{(\sum_k exp(W_{2k}))^2}, & \text{otherwise,} \end{cases}$$

$$\frac{\partial X_{C2t}^j}{\partial X_{C1l}^j} = \begin{cases} \tilde{W}_{2(l-t+\lfloor K_1/2\rfloor)}, & \text{if } 1 \le l-t+\lfloor K_1/2\rfloor \le K_1; \\ 0, & \text{otherwise.} \end{cases}$$

Box I.

denotes the number of samples. The processes and the loss function of IRCNN are as follows:

$$\begin{cases} X_{C1}^j = \sigma(X^j * W_1), \\ X_{C2}^j = X_{C1}^j * \tilde{W}_2, \\ \hat{Y}^j = X^j - X_{C2}^j, \end{cases} \tag{7}$$

$$L(\hat{Y}, Y) = \sum_{j=1}^{J} \|\hat{Y}^j - Y^j\|_2^2 + \eta QTV(\hat{Y}^j), \tag{8}$$

where $\tilde{W}_2 = softmax(W_2)$, $\sigma = tanh$ in our model, η is a non-negative parameter, $W_1 \in \mathbb{R}^{K_1}$ and $W_2 \in \mathbb{R}^{K_2}$ are the undetermined convolution filters.

According to the gradient descent method and the chain rule of derivation, W_1 and W_2 are learnt following the back propagation method, that is, $W_h^{(n+1)} = W_h^{(n)} - \lambda \nabla_{W_h} L, (h = 1, 2)$, where λ denotes the learning rate, and please refer to Box I for the calculation of $\nabla_{W_h} L$.

Next, we discuss the convergence of the training process of the IRCNN model expressed in Eq. (7).

Theorem 3.1. *For the IRCNN model defined in Eq. (7), the sequences $\{W_1^{(n)}\}$ and $\{W_2^{(n)}\}$ are generated by the gradient descent algorithm. Then, there exists a positive constant $\mathcal{L}$ independent of the input data, so that when the learning rate λ is less than $\mathcal{L}$, $\{W_1^{(n)}\}$ and $\{W_2^{(n)}\}$ converge to their corresponding stationary points, i.e., the sets $\{W_1 : \nabla_{W_1} L = \mathbf{0}\}$ and $\{W_2 : \nabla_{W_2} L = \mathbf{0}\}$, respectively.*

Proof. From the Lagrange's mean value theorem, it is obvious to find that the composition function composed by two functions, the β_1- and β_2-smooth respectively, is still β-smooth ($\beta \le \beta_1\beta_2$) under the condition that it can be composited.

According to Eq. (3), the IRCNN model can be seen as a composition function composed of a series of functions. Then, combining with Theorem 2.1, we have that if the function of the i-layer in the IRCNN model is β_i-smooth, it can be proved that the weights sequences obtained by the gradient descent method, converge under the condition that the learning rate satisfies $\lambda \le 2/\Pi\beta_i$.

For the function $\tilde{W}_2 = softmax(W_2)$ that is included in the IRCNN model, the second partial derivative of each $\tilde{W}_{2l}$ $(l = 1, \dots, K_2)$ with respect to its independent variable vector W_2 exists and satisfies:

$$\left|\frac{\partial^2 \tilde{W}_{2l}}{\partial W_{2i}\partial W_{2j}}\right| = \frac{exp(W_{2i})}{(\sum_k exp(W_{2k}))^3} \begin{cases} \sum_{k\neq i} exp(W_{2k})|\sum_k exp(W_{2k}) - 2exp(W_{2i})|, & \text{if } l = i = j, \\ exp(W_{2l})|2exp(W_{2i}) - \sum_k exp(W_{2k})|, & \text{if } l \neq i = j, \\ exp(W_{2i} + W_{2l} + W_{2j}), & \text{if } l \neq i \neq j, \end{cases} \le 2.$$

Hence, it states that each $\tilde{W}_{2l}$ is β-smooth (and $\beta \le 2$) according to the Lagrange's mean value theorem.

For the rest functions involved in IRCNN include quadratic function, tanh, and 1-D convolution operation, these functions can be easily proved to be β-smooth (β here is a general representation, and the β value of each function may not be the same.) by judging that their second derivative functions exist and bounded. Therefore, the conclusion is proved.

For the case of the model with an orthogonal constraint in the loss function L in Eq. (6), the orthogonal constraint is a new obstacle compared to the previous model. However, in the field of optimization, the study of optimization problems with orthogonal constraints has become very common. And the gradient-based projection method can be used to find $\mathbf{W}^o$ with convergence guarantees [34,35]. Furthermore, under the idea of back propagation used in updating the weights of the neural networks, the solutions of problem (6) can be obtained according to Alg. 4.

Algorithm 4: IRCNN with orthogonal constraint

```
i = 0, given the learning rates λ1, λ2, and initialize the W^o, W;
while i < Max_Iter do
    W^o ← W^o − λ1 ∇_{W^o} L;
    W^o ← P_{S_{p,q}}(W^o), where P_{S_{p,q}}(W^o) denotes the projection of
      W^o ∈ R^{p×q} to the Stiefel manifold S_{p,q}, P_{S_{p,q}}(W^o) = Ũ Ṽ^⊤,
      and Ũ Σ Ṽ^⊤ is the reduced singular value decomposition of
      W^o;
    W ← W − λ2 ∇_W L according to the back propagation
      method;
    i ← i + 1;
end
```

Since the convergence analysis of $\mathbf{W}$ calculated based on the gradient descent method is consistent with Theorem 3.1, and that of $\mathbf{W}^o$ calculated based on the gradient projection method has been discussed in the literature [34,35], so the convergence of Alg. 4 can be obtained intuitively.

Remark 1. Similar to the case with loss function in Eq. (5), we assume that in the IRCNN model, there are only two IRCNN inner loop blocks, i.e., $M = 2$ in Fig. 3. Furthermore, the IMFs obtained by the two blocks satisfy orthogonality, i.e., $\Omega_2 = \{(1, 2)\}$, $\hat{\Omega}_2 = \{1, 2\}$ and $\hat{\Omega}_2^c = \emptyset$ in Eq. (6). In this case, the IRCNN model can be reduced to a simpler formula, which looks close to the expressions of some orthogonal constraint algorithms, which reads:

$$\min_{\mathbf{W},\mathbf{W}^o} \|\mathbf{W}^o\hat{\mathbf{Y}} - \mathbf{Y}\|_F^2, \text{s.t. } \mathbf{W}^o\mathbf{W}^{o\top} = \mathbf{I}, \tag{9}$$

where $\hat{\mathbf{Y}}$ depends on $\mathbf{W}$.

4. Experiments

To evaluate the performance of the proposed IRCNN inner loop block and IRCNN models, we test them against ten aspects, which are: (1) Can IRCNN inner loop block be used to find the local average of the non-stationary signal? (2) Is IRCNN inner loop block still effective on noisy signal? (3) Can IRCNN be used to decompose the non-stationary signals? (4) Can IRCNN be effective on noisy signals? (5) Can IRCNN be effective on the signals composed of orthogonal mono-components? (6) Can IRCNN be effective on solutions of differential equations? (7) Is IRCNN capable of processing real signals? 8) Is IRCNN sensitive to the hyper-parameters? (9) How does IRCNN perform in terms of generalization? (10) How efficiently does IRCNN decompose signals?

In the experiments, we divide each of the constructed input data into the training and validation datasets with ratio 7 : 3, where the training dataset is used to learn the undetermined weights $\mathbf{W}$, the validation dataset is adopted to determine the hyper-parameters. For the hyper-parameters in IRCNN, including the number of iterations S per each IRCNN inner loop block, the sizes of convolutional kernels K_1, K_2 that occur in each iteration of the inner loop block, and the numbers of neurons in the convolutional layers, Hyperopt[1] is adopted to select their values by using the validation data, which is a Python library for serial and parallel optimization over awkward search spaces for hyper-parameters.

Since the existing signal decomposition methods generally take a long time to find a good combination of parameters for each signal, the number of components generated varies, and many methods also need to artificially find the components corresponding to the ground truths from the results, these factors make it difficult to evaluate the performance of the existing methods efficiently on the test set containing multiple signals. We will select some representative signals that are not in both training and validation sets as test signals to evaluate the decomposition performance of IRCNN.

Moreover, our proposed signal average method, i.e., the IRCNN inner loop block, will be compared with the existing signal average methods based on cubic spline envelopes (CSE) [1], optimization model (OP), a segment power-function based envelopes (SPFE) [36] and iterative filtering (IF) [7], respectively. For simplicity, we denote the averages obtained from the CSE, OP, SPFE, and IF as CSA, OPA, SPFA, and IF respectively. And the IRCNN[2] method will be compared with the state-of-the-art methods, including EMD, IF,[3] VMD[4][15], continuous wavelet transform based synchrosqueezing (SYNSQ_CWT) [18], short time Fourier transform based synchrosqueezing (SYNSQ_STFT[5]) [18], INCMD[6] [8], EWT[7] [19], ESMD [37], CEEMD[8][38], MEMD[9] [39], FDM[10] [20], and its variant called DCT_GAS_FDM[11] [21]. In addition, in the experiments for verifying the robustness of noise interference, the proposed IRCNN method is compared with the ensemble EMD (called EEMD[12]) model [40]; and it is compared with the M-LFBF[13][13,14] model to verify the orthogonality of the decomposed components. The code of IRCNN is publicly available on Github https://github.com/zhoudafa08/RRCNN.

Table 1
Evaluation indices.

Metric	MAE	RMSE	$\rho(c_1, c_2)$
Expression	$\frac{1}{N}\sum_{t=1}^{N}\lvert\hat{Y}_t - Y_t\rvert$	$\sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{Y}_t - Y_t)^2}$	$\frac{\lvert\langle c_1, c_2\rangle\rvert}{\lVert c_1\rVert_2 \lVert c_2\rVert_2}$

Table 2
Inputs and labels used in Section 4.1, where $t \in [0, 3]$.

$x_1(t)$	$x_2(t)$	Inputs	Labels	Notes
$0.1kt$	$\cos(3lt)$ $\cos(3klt + t + \cos(t))$	$x_1(t) + x_2(t)$	$x_2(t)$	$k = 2, 3, \ldots, 9$
0	$\cos(3lt)$ $\cos(3klt + t + \cos(t))$			$l = 2, 4, 6, 8$
$0.1k$	$\sin(3klt)$ $\sin(3klt + t^2 + \cos(t))$	$x_1(t) + x_2(t)$	$x_2(t)$	$k = 1, 2, \ldots, 10$
0	$\sin(3klt)$ $\sin(3klt + t^2 + \cos(t))$			$l = 2, 4, \ldots, 28$
$3 + 2\cos(0.5kt)$	$\cos(0.5klt^2)$ $\cos(0.5lt^2 + l\cos(t))$	$x_1(t)x_2(t)$	$x_1(t)x_2(t)$	$k = 2, 3, \ldots, 6$
1.0	$\cos(0.5klt^2)$ $\cos(0.5lt^2 + l\cos(t))$			$l = 4, 5, \ldots, 9$

The results are measured by the metrics listed in Table 1, where Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are used to measure the errors between the predicted results $\hat{Y} \in \mathbb{R}^N$ and the ground truth $Y \in \mathbb{R}^N$, and $\rho(c_1, c_2)$ is used to evaluate the orthogonality between the resulting components $c_1, c_2 \in \mathbb{R}^N$. All the experiments are performed in Python 3.8.12, Matlab R2020b or R version 4.3.3 on a Dell Precision 5820 tower with Intel(R) Xeon(R) W-2102 processor (2.90 GHz), 64G memory, and Ubuntu 18.04.3 operating system.

4.1. Can IRCNN inner loop block be used to find the local average of the non-stationary signal?

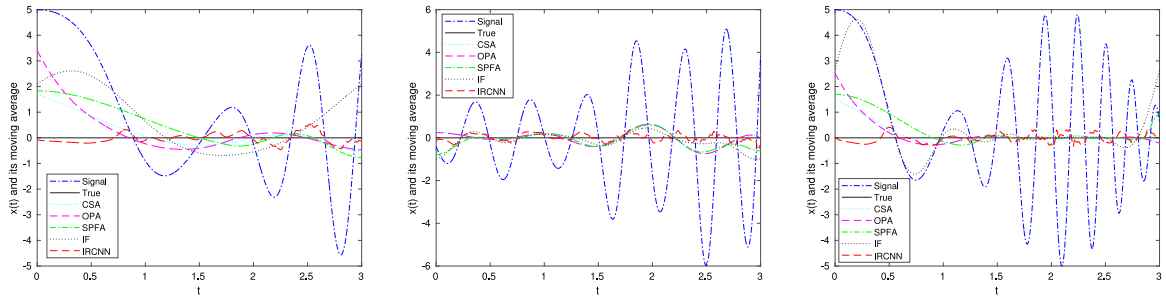
We first evaluate the performance of the proposed IRCNN inner loop block in solving the local average of the non-stationary signal. Several signals composed of the linear function and the mono-component function, or just the mono-component function, are constructed as the inputs, where the mono-component function can generally be expressed as $a(t)\cos\theta(t)$, which meets $a(t), \theta'(t) > 0\ \forall t$, and the changes in time of $a(t)$ and $\theta'(t)$ are much slower than that of $\theta(t)$. Ideally, the average of the mono-component signal is zero. The input signals and the corresponding labels we construct in this part are listed in Table 2. It should be pointed out that the label here represents the first IMF of the corresponding input signal, not the local average. The local average can be computed by subtracting the label from the input signal.

After the IRCNN inner loop block is trained with the inputs in Table 2, we select three mono-component signals with different instantaneous frequencies and instantaneous amplitudes, discussed in Examples 1–2, which are not included in the inputs, to test the performance of the IRCNN inner loop block, respectively.

Example 1. $x(t) = (3 + 2\cos(2t))\cos(2t^2)$, $t \in [0, 3]$.

Example 2. $x(t) = (2t + \cos(2t^2))\cos(12t + t^2 + 2\cos(t))$, $t \in [0, 3]$.

[1] Github website of Hyperopt: https://github.com/hyperopt/hyperopt.
[2] Code of IRCNN is available at https://github.com/zhoudafa08/RRCNN.
[3] Code of IF: http://people.disim.univaq.it/~antonio.cicone/Software.html
[4] Code of VMD: https://www.mathworks.com/help/wavelet/ref/vmd.html.
[5] Codes of SYNSQ_CWT and SYNSQ_STFT: https://github.com/ebrevdo/synchrosqueezing.
[6] Code of INCMD: https://github.com/sheadan/IterativeNCMD.
[7] Code of EWT: https://ww2.mathworks.cn/help/wavelet/ug/empirical-wavelet-transform.html.
[8] Code of CEEMD: https://rdrr.io/cran/hht/man/CEEMD.html.
[9] Code of MEMD: https://it.mathworks.com/matlabcentral/fileexchange/70566\protect\discretionary{\char\hyphenchar\font}{}{}multivariate\protect\discretionary{\char\hyphenchar\font}{}{}signal\protect\discretionary{\char\hyphenchar\font}{}{}denoisingtoolbox?s_tid=srchtitle.
[10] Code of FDM: https://www.researchgate.net/publication/274570245_Matlab_Code_Of_The_Fourier_Decomposition_Method_FDM.
[11] Code of DCT_GAS_FDM: https://www.researchgate.net/publication/326294577_MATLABCodeOfFDM_DCT_DFT_FIR_FSASJuly2018.
[12] Code of EEMD: http://perso.ens-lyon.fr/patrick.flandrin/emd.html.
[13] Code of M-LFBF: http://perso.ens-lyon.fr/nelly.pustelnik/.



(a) Example 1 (b) Example 2 (c) Example 3

Fig. 4. Moving averages by different methods of Examples 1–3.

Table 3
Metrics of the moving averages of the input in Example 1.

Metric	CSA	OPA	SPFA	IF	IRCNN
MAE	0.4082	0.5296	0.6558	1.0977	**0.1656**
RMSE	0.6001	0.8763	0.8884	1.3829	**0.2040**

Table 4
Metrics of the moving averages of the input in Example 2.

Metric	CSA	OPA	SPFA	IF	IRCNN
MAE	0.2544	0.2514	0.2943	0.2791	**0.1442**
RMSE	0.3267	0.3276	0.3662	0.3672	**0.1718**

Table 5
Metrics of the moving averages of the input in Example 3.

Metric	CSA	OPA	SPFA	IF	IRCNN
MAE	0.2067	0.2448	0.3646	0.8624	**0.1418**
RMSE	0.4185	0.5427	0.6388	1.5833	**0.1712**

Example 3. $x(t) = (3 + 2\cos(3t))\cos(5t^2)$, $t \in [0, 3]$.

The moving averages of the signals in Examples 1–3 obtained from different methods are shown in Fig. 4(a)–(c), respectively, and the errors between the obtained moving averages and the true average are listed in Tables 3–5, respectively. According to the results, we can observe the following phenomena:

(i) The existing methods are prone to boundary effects, which can be seen from the left boundaries of Fig. 4(a)–(c). However, the IRCNN inner loop block method can avoid this problem to a certain extent.

(ii) When the signal is in a situation where the amplitude changes quickly and the frequency changes slowly, the IRCNN inner loop block performs best among all models according to the left parts of Fig. 4(a)–(c). When the amplitude change is reduced and the frequency change is accelerated, its performance may even be inferior to other models, which can be seen from the right half of Fig. 4(c).

(iii) Even though the IRCNN inner loop block has some dazzling and bleak performance compared with other local average methods, it can be seen from Tables 3–5 that the MAE and RMSE of IRCNN are significantly reduced compared to other models.

The reason for the phenomenon above can be attributed to:

(i) The averages obtained by the comparison methods are basically determined by the local information of the signal, which makes the results reasonable when the information is sufficient (e.g., the part of the amplitude change is reduced and the frequency change is accelerated); and the results differ greatly when the information is insufficient (e.g., the part of the amplitude changes quickly and the frequency changes slowly).

Table 6
Inputs disturbed by the Gaussian noise with the SNR set to $15dB$, and the labels used in Section 4.2, where $t \in [0, 3]$.

$x_1(t)$	$x_2(t)$	Inputs	Labels	Notes
$0.1kt$	$\cos(3lt)$ $\cos(3klt + t + \cos(t))$	$x_1(t) + x_2(t) + \varepsilon(t)$	$x_2(t)$	$k = 2, 3, \ldots, 9$
0	$\cos(3lt)$ $\cos(3klt + t + \cos(t))$			$l = 2, 4, 6, 8$
$0.1k$	$\sin(3klt)$ $\sin(3klt + t^2 + \cos(t))$	$x_1(t) + x_2(t) + \varepsilon(t)$	$x_2(t)$	$k = 1, 2, \ldots, 10$
0	$\sin(3klt)$ $\sin(3klt + t^2 + \cos(t))$			$l = 2, 4, \ldots, 28$
$3 + 2\cos(0.5kt)$	$\cos(0.5klt^2)$ $\cos(0.5lt^2 + l\cos(t))$	$x_1(t)x_2(t) + \varepsilon(t)$	$x_1(t)x_2(t)$	$k = 2, 3, \ldots, 6$
1.0	$\cos(0.5klt^2)$ $\cos(0.5lt^2 + l\cos(t))$			$l = 4, 5, \ldots, 9$

Table 7
Metrics of the moving averages of the input in Example 4 obtained from different methods.

Metric	CSA	OPA	SPFA	IF	IRCNN
MAE	0.2023	0.1998	0.1936	0.2183	**0.1190**
RMSE	0.2529	0.2501	0.2420	0.2762	**0.1500**

(ii) The filter weights of each convolutional layer in the IRCNN are shared, they are determined by all the information contained in the whole signal. Therefore, the average obtained by the IRCNN is relatively stable, and it is not easy to cause interference due to the large difference in the changing of the signal amplitude and frequency.

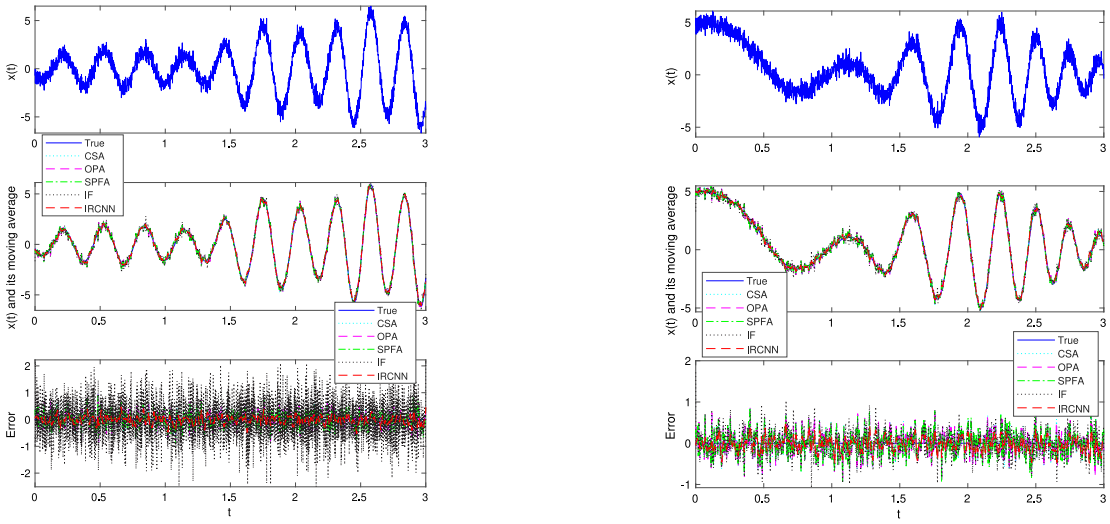
4.2. Is IRCNN inner loop block still effective on noisy signal?

In this part, we consider the robustness of the IRCNN inner loop block model to noise based on the constructed inputs and labels in Section 4.1. Specifically, each input signal is perturbed with additive Gaussian noise with the signal-to-noise ratio (SNR) set to 15 dB, and the corresponding label remains unchanged, as detailed in Table 6. Similar to the section above, we select a noisy signal, which is essentially the signal in Examples 2–3 with additive Gaussian noise, and is described in Examples 4–5, to test the performance.

Example 4. $x(t) = (2t+\cos(2t^2))\cos(20t+t^2+2\cos(t))+\varepsilon(t)$, where $t \in [0, 3]$ and $SNR = 15$ dB.

Example 5. $x(t) = (3 + 2\cos(3t))\cos(5t^2) + \varepsilon(t)$, where $t \in [0, 3]$ and $SNR = 15$ dB.

The results of Examples 4–5 are shown in Fig. 5(a)–(b) and Tables 7–8. From the results, we can find that the IRCNN inner loop block performs the most robust among all models for the signal with



(a) Example 4 (b) Example 5

Fig. 5. Moving averages by different methods of Examples 4–5.

Table 8
Metrics of the moving averages of the input in Example 5.

Metric	CSA	OPA	SPFA	IF	IRCNN
MAE	0.2141	0.2126	0.2004	0.2339	**0.1290**
RMSE	0.2676	0.2658	0.2520	0.2967	**0.1623**

additive Gaussian noise. Specifically, in Example 4, the IRCNN inner loop block reduces the MAE and RMSE from $0.1936, 0.2420$, for the second the best method (i.e., SPFA), to $0.1190, 0.1500$, respectively; and in Example 5, the IRCNN inner loop block reduces the MAE and RMSE from $0.2004, 0.2520$, for the second the best method (i.e., SPFA), to $0.1290, 0.1623$, respectively.

4.3. Can IRCNN be used to decompose the non-stationary signals?

To demonstrate the decomposition performance of IRCNN for non-stationary signals, we only consider decomposing the signals consisting of two components. The input signals in the part can be divided into two categories: one is composed of a mono-component signal and a zero signal, and the other is of two mono-components with close frequencies. The former is to train IRCNN describe the zero local average of the mono-component signal; and the latter is to enable IRCNN to decompose signals with close frequencies, which is the main factor causing the mode mixing effect. The inputs and labels are constructed as Dataset_1 and shown in Table 9. To challenge the proposed model, we hereby choose the signal composed of two cosine signals with frequencies of 2.5 Hz and 3.4 Hz respectively, described in Example 6, that is more prone to introduce the mode mixing effect in the existing methods.

Example 6. $x(t) = \cos(5\pi t) + \cos(6.8\pi t)$, $t \in [0, 6]$.

In Example 6, the components predicted by the trained IRCNN model are compared with those obtained from the state-of-the-art methods in signal decomposition. The metrics of the errors between the obtained components and the labels, measured by MAE and RMSE, are shown in Table 10. In addition, to compare IRCNN and the existing methods more intuitively, we select the top three methods with comprehensive performance from Table 10, i.e., IRCNN, EMD, and INCMD, and plot their obtained components and the corresponding time–frequency–energy (TFE) representations in Fig. 6(a), (b), respectively. It should be noted that the identification of an optimal TFE representation is a research topic on its own, and it is out of the scope of this work. Here, we set as TFE representation the Fourier quadrature transforms that was proposed in [21].

According to the results, we have the following conclusions:

(i) The mode mixing problem is indeed a big challenge for some of the existing methods. For example, the maximum value of $x(t)$ is 2, but the MAEs of the components obtained by the CEEMD and SYNSQ_STFT method are greater than 0.55, which basically do not separate the cosine signals with frequencies of 2.5 Hz and 3.4 Hz from $x(t)$.

(ii) Many methods achieve satisfactory decomposition for $x(t)$. For example, it can be seen from the left plots in the 2nd and 3rd rows of Fig. 6(a) that the components obtained by the EMD, INCMD, and IRCNN methods have relatively consistent oscillation modes with the ground truths. This viewpoint can also be drawn from Fig. 6(b), although there are some obvious fluctuations, the TFE representations of the two components, obtained by EMD, INCMD, and IRCNN methods, are basically separated just like the those of the real components.

(iii) Nonetheless, a closer look at the right plots in the 2nd and 3rd rows of Fig. 6(a) reveals the subtle advantage of the IRCNN model at the boundaries. Due to the incompleteness of the waveform at the boundary, many existing methods are deeply affected by it, as are EMD and INCMD. However, the weights of the convolutional kernels in the IRCNN model rely on the entire waveform of the training data, containing the information at the endpoints and at the internal points, which helps to reduce the boundary effect.

4.4. Can IRCNN be effective on noisy signals?

Similar to the IRCNN inner loop block, we verify the robustness of IRCNN against additive Gaussian noise in this part. The constructed inputs and labels form Dataset_2, which is listed in Table 11, where the inputs are generated by introducing additive Gaussian noise with the SNR set to 25 dB to the signals in Table 9. After the IRCNN model is trained, we choose the signal consisting of two mono-components and additive Gaussian noise with a SNR of 15 dB as the test data, which is given in Example 7. Since the smaller SNR value, the greater the noise, the noise of $x(t)$ is larger than that in the training data.

Example 7. $x(t) = \cos(5\pi t) + \sin(8\pi t + 2t^2 + \cos(t)) + \varepsilon(t)$, $t \in [0, 6]$, SNR=15 dB.

Table 9
Dataset_1: Inputs and labels used in Section 4.3, where $t \in [0, 6]$.

$x_1(t)$	$x_2(t)$	Inputs	Labels	M (Number of IMFs)	Notes
$\cos(k\pi t)$	$\cos((k+1.5)\pi t)$ $\cos((k+1.5)\pi t + t^2 + \cos(t))$	$x_1(t) + x_2(t)$	$[x_2(t), x_1(t)]$	2	$k = 5, 6, \ldots, 14$
0	$\cos((k+1.5)\pi t)$ $\cos((k+1.5)\pi t + t^2 + \cos(t))$				
$\cos(k\pi t)$	$\cos(kl\pi t)$ $\cos(kl\pi t + t^2 + \cos(t))$	$x_1(t) + x_2(t)$	$[x_2(t), x_1(t)]$	2	$k = 5, 6, \ldots, 14$ $l = 2, 3, \ldots, 19$
0	$\cos(kl\pi t)$ $\cos(kl\pi t + t^2 + \cos(t))$				

Table 10
Metrics of the errors between the obtained components by different methods and the ground truth of Example 6.

Method	c_1		c_2		Method	c_1		c_2	
	MAE	RMSE	MAE	RMSE		MAE	RMSE	MAE	RMSE
EMD [1]	0.1049	0.2110	0.1132	0.2058	INCMD [8]	0.1144	0.1667	0.1160	0.1509
VMD [15]	0.2193	0.2479	0.2193	0.2479	SYNSQ_CWT [18]	0.2227	0.2496	0.2589	0.2965
EWT [19]	0.2145	0.2605	0.2145	0.2605	SYNSQ_STFT [18]	0.6620	0.7402	0.6778	0.7608
FDM [20]	0.2438	0.2947	0.2429	0.2893	DCT_GAS_FDM [21]	0.1608	0.2005	0.1605	0.2003
IF [7]	0.2253	0.2756	0.2060	0.2489	ESMD [37]	0.1317	0.1809	0.2075	0.2670
MEMD [39]	0.1503	0.2587	0.1707	0.2614	**IRCNN**	**0.1013**	**0.1242**	**0.0331**	**0.0422**
CEEMD [38]	0.5637	0.6410	0.5584	0.6265					

Table 11
Dataset_2: Inputs disturbed by the Gaussian noise with the SNR set to $25dB$, and the labels used in Section 4.4, where $t \in [0, 6]$.

$x_1(t)$	$x_2(t)$	Inputs	Labels	M (Number of IMFs)	Notes
$\cos(k\pi t)$	$\cos((k+1.5)\pi t)$ $\cos((k+1.5)\pi t + t^2 + \cos(t))$	$x_1(t) + x_2(t) + \varepsilon(t)$	$[x_2(t), x_1(t)]$	2	$k = 5, 6, \ldots, 14$
0	$\cos((k+1.5)\pi t)$ $\cos((k+1.5)\pi t + t^2 + \cos(t))$				
$\cos(k\pi t)$	$\cos(kl\pi t)$ $\cos(kl\pi t + t^2 + \cos(t))$	$x_1(t) + x_2(t) + \varepsilon(t)$	$[x_2(t), x_1(t)]$	2	$k = 5, 6, \ldots, 14$ $l = 2, 3, \ldots, 19$
0	$\cos(kl\pi t)$ $\cos(kl\pi t + t^2 + \cos(t))$				

The errors between the ground truths and the components obtained by different methods, measured by MAE and RMSE, are reported in Table 12. Furthermore, the components, errors, and TFE representations of the three best performing methods, i.e., FDM, DCT_GAS_FDM, and IRCNN, are shown in Fig. 6(d), (e), respectively. According to the results, we can find that IRCNN works for the signals with additive Gaussian noise, although there is no overwhelming advantage, especially over FDM. Specifically, from the left plots in the 2nd-3rd rows of Fig. 6(d), IRCNN basically separates the two mono-components, and the resulting components are consistent with the ground truths in the oscillation mode. Moreover, as shown in the right plots in the 2nd-3rd rows of Fig. 6(d), the errors of IRCNN are relatively evenly dispersed in the entire time period, while those of the FDM and DCT_GAS_FDM methods are both small in the middle and large at the boundaries, which is consistent with the observations in Section 4.3.

Since IRCNN is a method designed with the help of CNN in the time domain, it can obtain an effect comparable to the existing methods in the time domain. Due to the lack of a prior information of IRCNN in the frequency domain, the effect of this method might be slightly reduced from the time–frequency domain. According to the results in Fig. 6(e), the TFE distributions of the two mono-components obtained by the FDM, DCT_GAS_FDM, and IRCNN, are obviously spaced apart, but the TFE distribution of the component c_1 obtained by IRCNN has a much more severe jitter than the true component $\sin(8\pi t + 2t^2 + \cos(t))$ in the interval $t \in [2, 4]$. However, from the TFE representations, we can also observe that the IRCNN method is able to reduce boundary effect compared to other methods.

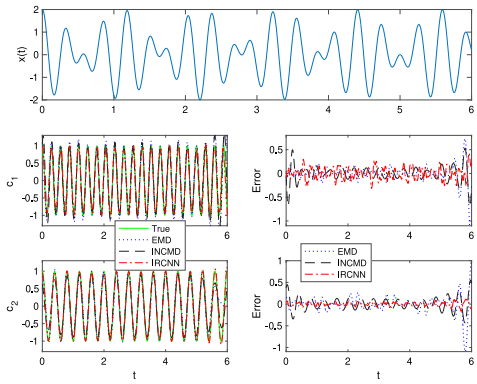
We further investigate the decomposition robustness of the trained IRCNN model in dealing with the signal in Example 7 for different values of SNR, and also compare it with the decompositions produced by other methods. All the results are listed in Table 13, the RMSEs are also graphically shown in Fig. 7. From them, we can derive the following conclusions:

(i) Although the two components obtained by IRCNN do not perform best in all cases, the errors measured by both MAE and RMSE of the second components are the smallest for the cases with different SNRs, and the performance of the first component under RMSE is basically in the top 4 among all models. Therefore, IRCNN model has good robustness in general.

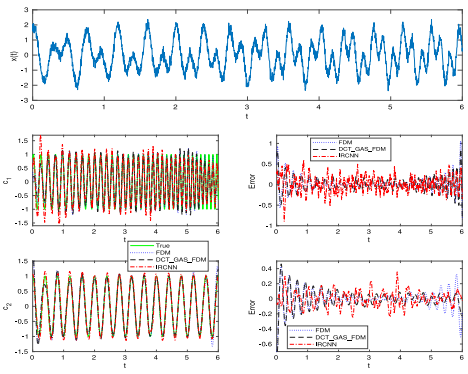
(ii) The IRCNN model is trained on the signals with additive Gaussian noise with SNR of 25. When dealing with other noisy signals with different SNRs, the errors measured by MAE and RMSE basically increase as the SNR decreases (it means that the proportion of noise increases). The other methods also mostly follow this rule. For a few abnormal cases, we speculate that because these methods rely heavily on some parameters, the values selected in some experiments happen to be in their favor, while others they are not.

4.5. Can IRCNN be effective on the signals composed of orthogonal mono-components?

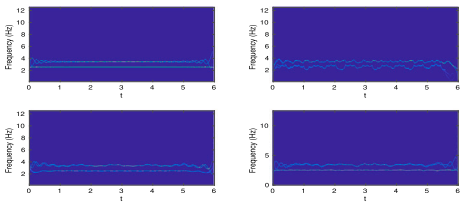
In this part, we test the proposed IRCNN model on the signal composed of the orthogonal components. As discussed in Section 3.2, the IRCNN model in this part should have been equipped with the loss function with an orthogonal constraint, we directly add an inner product term to the loss function to control the orthogonality, i.e., $\gamma \frac{|\langle c_1, c_2 \rangle|}{\|c_1\|_2 \|c_2\|_2}$, where γ denotes the positive penalty parameter, and c_1 and c_2 are two components that are orthogonal. Then we train the model by the back propagation method based on the loss function. Although there is no guarantee of convergence in this case, it is simple, computationally



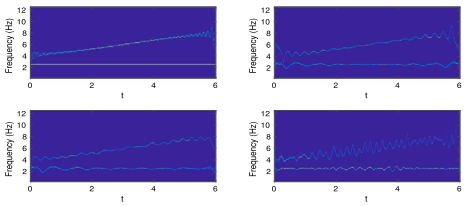
(a) True and obtained components of the signal (top panel) in Example 6. Left of the 2nd-3rd rows: true (green solid curve) and components by EMD (blue dotted curve), INCMD (black dashed curve), and IRCNN (red dot dash curve). Right of the 2nd-3rd rows: the errors between the ground truth and the obtained components by EMD (blue dotted curve), INCMD (black dashed curve), and IRCNN (red dot dash curve), respectively.



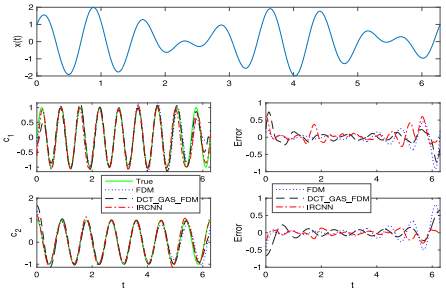
(d) True and obtained components of the signal (top panel) in Example 7. Left of the 2nd-3rd rows: true (green solid curve) and component by FDM (blue dotted curve), DCT_GAS_FDM (black dashed curve), and IRCNN (red dot dash curve). Right of the 2nd-3rd rows: the errors between the true and obtained components by FDM (blue dotted curve), DCT_GAS_FDM (black dashed curve), and IRCNN (red dot dash curve), respectively.



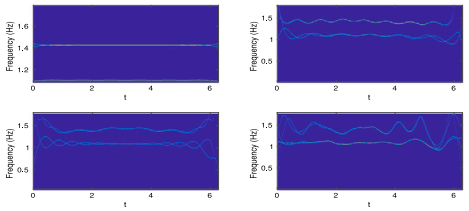
(b) TFE representations of the true and obtained components of the signal in Example 6. Left of the top row: TFE of the ground truths. Right of the top row: TFE of the results by EMD. Left of the bottom row: TFE of the results by INCMD. Right of the bottom row: TFE of the results by IRCNN.



(e) TFE representations of the true and obtained components of the signal in Example 7. Left of the top row: TFE of the ground truths. Right of the top row: TFE of the results by FDM. Left of the bottom row: TFE of the results by DCT_GAS_FDM. Right of the bottom row: TFE of the results by IRCNN.



(c) Same as (d) for Example 8.



(f) Same as (e) for Example 8.

Fig. 6. Results of Examples 6–8.

Table 12
Metrics of the errors between the results obtained by different methods and the ground truths of Example 7.

Method	c_1		c_2		Method	c_1		c_2	
	MAE	RMSE	MAE	RMSE		MAE	RMSE	MAE	RMSE
EMD [1]	0.1693	0.2961	0.2029	0.3485	INCMD [8]	0.2502	0.3724	0.2549	0.3704
VMD [15]	0.1440	0.2371	0.1363	0.2286	SYNSQ_CWT [18]	0.5817	0.6477	0.0915	0.1522
EWT [19]	0.1840	0.2945	0.1754	0.2884	SYNSQ_STFT [18]	0.4353	0.4992	0.2009	0.2463
FDM [20]	**0.1039**	**0.1765**	0.0857	0.1327	DCT_GAS_FDM [21]	0.1288	0.1801	0.0826	0.1252
IF [7]	0.1482	0.2267	0.1322	0.2143	EEMD [40]	0.1870	0.2308	0.1243	0.1925
ESMD [37]	0.2153	0.2989	0.2037	0.2786	MEMD [39]	0.3754	0.5680	0.3238	0.4869
CEEMD [38]	0.2024	0.2867	0.1757	0.2883	**IRCNN**	0.1556	0.1977	**0.0805**	**0.1014**

efficient, and basically meets the expectation of orthogonality from the experimental results.

The Fourier basis is adopted for the construction of the inputs of IRCNN because they are orthogonal. The constructed inputs and labels constitutes Dataset_3, are given in Table 14. After the IRCNN is trained, we use it to decompose the signal given in Example 8, which is composed of two mono-components that are orthogonal and close in frequency.

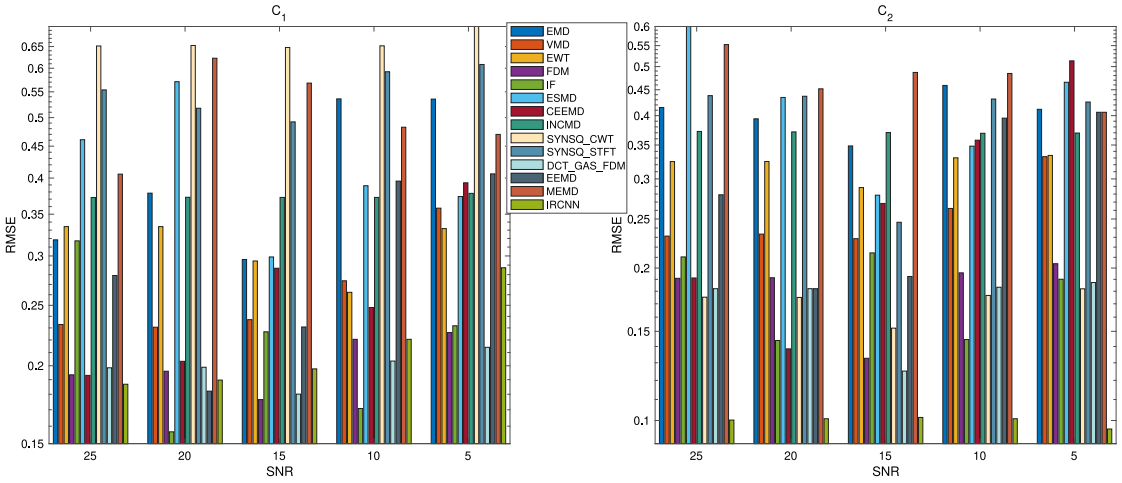


Fig. 7. Graphic illustration of RMSE between the results by different methods and the true components for dealing with the signal in Example 7 by changing the value of SNR.

Table 13
Metrics of the errors (MAE, RMSE) of Example 7 by changing the value of SNR.

IMF	Model \ SNR	25	20	15	10	5
c_1	EMD [1]	(0.1544, 0.3184)	(0.2111, 0.3783)	(0.1693, 0.2961)	(0.4273, 0.5368)	(0.3683, 0.5354)
	VMD [15]	(0.1359, 0.2329)	(0.1404, 0.2306)	(0.1440, 0.2371)	(0.1816, 0.2737)	(0.2585, 0.3579)
	EWT [19]	(0.1888, 0.3343)	(0.1888, 0.3343)	(0.1840, 0.2945)	(0.1690, 0.2623)	(0.2353, 0.3319)
	FDM [20]	(0.1352, 0.1934)	(0.1367, 0.1960)	(**0.1039**, **0.1765**)	(0.1588, 0.2205)	(0.1582, 0.2261)
	IF [7]	(0.2055, 0.3172)	(**0.0865**, **0.1567**)	(0.1482, 0.2267)	(**0.1091**, **0.1708**)	(0.1630, 0.2318)
	ESMD [37]	(0.3211, 0.4608)	(0.3358, 0.5709)	(0.2153, 0.2989)	(0.2312, 0.3888)	(0.2550, 0.3736)
	CEEMD [38]	(0.1368, 0.1930)	(0.1476, 0.2033)	(0.2024, 0.2867)	(0.1745, 0.2480)	(0.2858, 0.3930)
	INCMD [8]	(0.2502, 0.3723)	(0.2501, 0.3726)	(0.2502, 0.3724)	(0.2520, 0.3724)	(0.2617, 0.3780)
	SYNSQ_CWT [18]	(0.5852, 0.6514)	(0.5865, 0.6526)	(0.5817, 0.6477)	(0.5856, 0.6518)	(0.6461, 0.7276)
	SYNSQ_STFT [18]	(0.4442, 0.5539)	(0.4067, 0.5176)	(0.4353, 0.4922)	(0.4839, 0.5923)	(0.4955, 0.6084)
	DCT_GAS_FDM [21]	(**0.1343**, 0.1985)	(0.1347, 0.1989)	(0.1288, 0.1801)	(0.1397, 0.2035)	(**0.1522**, **0.2141**)
	EEMD [40]	(0.1866, 0.2791)	(0.1274, 0.1821)	(0.1870, 0.2308)	(0.2841, 0.3955)	(0.2996, 0.4062)
	MEMD [39]	(0.2379, 0.4058)	(0.3666, 0.6226)	(0.3754, 0.5680)	(0.3564, 0.4826)	(0.3299, 0.4699)
	IRCNN	(0.1462, **0.1868**)	(0.1487, 0.1897)	(0.1556, 0.1977)	(0.1749, 0.2205)	(0.2296, 0.2872)
c_2	EMD [1]	(0.1862, 0.4150)	(0.1995, 0.3943)	(0.2029, 0.3485)	(0.3276, 0.4589)	(0.3009, 0.4117)
	VMD [15]	(0.1341, 0.2313)	(0.1374, 0.2334)	(0.1363, 0.2286)	(0.1690, 0.2623)	(0.2353, 0.3319)
	EWT [19]	(0.1801, 0.3246)	(0.1818, 0.3247)	(0.1754, 0.2884)	(0.1955, 0.3302)	(0.2111, 0.3338)
	FDM [20]	(0.1353, 0.1909)	(0.1356, 0.1914)	(0.0857, 0.1327)	(0.1388, 0.1957)	(0.1478, 0.2040)
	IF [7]	(0.1881, 0.2104)	(0.0836, 0.1439)	(0.1322, 0.2143)	(0.0908, 0.1445)	(0.1289, 0.1901)
	ESMD [37]	(0.4478, 0.6566)	(0.2354, 0.4345)	(0.2037, 0.2786)	(0.2062, 0.3482)	(0.3172, 0.4656)
	CEEMD [38]	(0.1085, 0.1912)	(0.1041, 0.1385)	(0.1757, 0.2683)	(0.2144, 0.3578)	(0.3499, 0.5132)
	INCMD [8]	(0.2546, 0.3723)	(0.2545, 0.3715)	(0.2549, 0.3704)	(0.2567, 0.3692)	(0.2630, 0.3696)
	SYNSQ_CWT [18]	(0.1157, 0.1752)	(0.1155, 0.1750)	(0.0915, 0.1522)	(0.1171, 0.1766)	(0.1218, 0.1820)
	SYNSQ_STFT [18]	(0.3651, 0.4381)	(0.3641, 0.4369)	(0.2009, 0.2463)	(0.3597, 0.4314)	(0.3549, 0.4256)
	DCT_GAS_FDM [21]	(0.1210, 0.1821)	(0.1219, 0.1821)	(0.0826, 0.1252)	(0.1276, 0.1833)	(0.1367, 0.1873)
	EEMD [40]	(0.1125, 0.2791)	(0.1274, 0.1821)	(0.1243, 0.1925)	(0.2841, 0.3955)	(0.2996, 0.4062)
	MEMD [39]	(0.2981, 0.5526)	(0.3017, 0.4519)	(0.3238, 0.4869)	(0.3473, 0.4846)	(0.2749, 0.4062)
	IRCNN	(**0.0799**, **0.1002**)	(**0.0802**, **0.1008**)	(**0.0805**, **0.1014**)	(**0.0795**, **0.1008**)	(**0.0731**, **0.0962**)

Table 14
Dataset_3: Inputs and labels used in Section 4.5, where $t \in [0, 2\pi]$.

$x_1(t)$	$x_2(t)$	Inputs	Labels	M (Number of IMFs)	Notes
$\cos kt$	$\sin(k+l)t$	$x_1(t)+x_2(t)$	$[x_2(t), x_1(t)]$	2	$k = 6, 7, 8, 9$ $l = 3, 5, \ldots, 33$

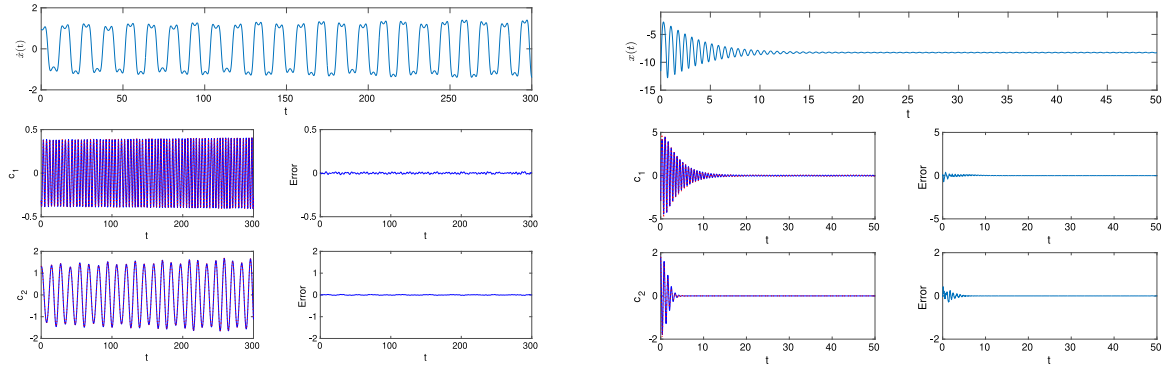
Example 8. $x(t) = \cos(7t) + \sin(9t)$, $t \in [0, 2\pi]$.

The orthogonality and the errors between the resulting components and the corresponding ground truths are reported in Table 15. According to the results, the EWT and DCT_GAS_FDM methods perform best in terms of orthogonality, which is attributed to the fact that the former is based on the wavelet transform and the latter is based on the discrete cosine transform, both of which have strong orthogonal constraint on the decomposition results. For IRCNN, its orthogonality is promoted by minimizing the loss function. Therefore, on one hand, the results of IRCNN tend to find a balance in each item of the loss function. On the other hand, the limited iterative process cannot ensure that the results are completely converges to the true solution. Combined with orthogonality and error metrics, the overall performance of the IRCNN model is still satisfactory. Specifically, it is not the best in terms of orthogonality, but it is also very close to orthogonality, outperforming the other models except EWT and DCT_GAS_FDM; in terms of error, its two components are also the closest to the true components.

Moreover, we select the top three methods in terms of the error metrics from Table 15, that is, FDM, DCT_GAS_FDM, and IRCNN, and plot their obtained components, errors, and TFE representations in Fig. 6(c), (f), respectively. From the plots, we can draw a conclusion

Table 15
Metrics of the orthogonality, and errors of the obtained components by different methods in Example 8.

Method	$\rho(c_1, c_2)$	c_1		c_2		Method	$\rho(c_1, c_2)$	c_1		c_2	
		MAE	RMSE	MAE	RMSE			MAE	RMSE	MAE	RMSE
EMD [1]	0.1141	0.1796	0.2776	0.2615	0.3581	INCMD [8]	0.1785	0.1846	0.2485	0.6205	0.7004
VMD [15]	0.4351	0.5682	0.6303	0.5682	0.6303	SYNSQ_CWT [18]	0.6268	0.7697	0.9299	0.4094	0.4823
EWT [19]	**0.0000**	0.6364	0.7070	0.6364	0.7070	SYNSQ_STFT [18]	0.1561	0.3225	0.3880	0.3202	0.3835
IF [7]	0.6896	0.2262	0.2776	0.2519	0.3092	M-LFBF [14]	0.9543	0.3858	0.4721	0.3857	0.4720
FDM [20]	0.0939	0.1229	0.2047	0.1244	0.2035	DCT_GAS_FDM [21]	**0.0000**	0.1222	0.1898	0.1211	0.1824
ESMD [37]	0.0127	0.6058	0.6768	0.6083	0.6811	MEMD [39]	0.2715	0.2781	0.3448	0.2932	0.3452
CEEMD [38]	0.0518	0.5923	0.6677	0.5901	0.6631	**IRCNN**	0.0595	**0.1195**	**0.1759**	**0.0639**	**0.0889**



(a) Example 9 (b) Example 10

Fig. 8. (a) Components of $\dot{x}(t)$ of Duffing equation with $\alpha = -0.85, \beta = 1.05, \gamma = 0.045, \omega = 1.3$ by EMD and IRCNN. First row: the original signal. The 2nd-3rd rows, left: component obtained by EMD (blue solid curve) and IRCNN (red dotted curve); right: the error between the obtained components by EMD and IRCNN. (b) Components of $x(t)$ of Lorenz equation with $\sigma = 10.5, r = 20.5, b = 3.5$ by EMD and IRCNN. First row: the original signal. The 2nd-3rd rows, left: component obtained by EMD (blue solid curve) and IRCNN (red dotted curve); right: the error between the obtained components by EMD and IRCNN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

similar to Example 6, that is, the IRCNN model can obtain the performance that is comparable to the state-of-the-art methods in the time domain, especially at the boundary, which can weaken the impact of incomplete waveforms at the boundary to a certain extent. However, due to the lack of a priori information in the frequency domain in the design of IRCNN, its advantage in terms of TFE distribution might be reduced when compared with other methods especially in the middle of the signal. However its performance are overall better than existing methods at the boundaries, also in the time–frequency domain.

4.6. Can IRCNN be effective on solutions of differential equations?

To address this question, and following what has been done in the seminal work by Huang et al. [1], we test IRCNN against the solutions of a Duffing and Lorenz equations. The labels are necessary in the training process of the IRCNN model, however, they are not known in this instance. Since the EMD method works well for these two types of signals, we decided to use the results of EMD as the labels to verify the learning and generalization capabilities of the IRCNN model.

Example 9. We consider the following Duffing equation: $\ddot{x} + \alpha x + \beta x^3 = \gamma \cos(\omega t)$. We focus our attention on the decomposition of $\dot{x}(t)$. We first construct the inputs by solving the equation using the Euler method with the parameters set to $\alpha \in [-1, 0)$, $\beta \in [0.8, 1.2]$, $\omega \in [1.0, 1.4]$ with step size 0.1, and $\gamma \in [0.04, 0.06]$ with step size 0.005, respectively, where $t \in [0, 300]$ and the initial conditions: $\{x(0) = 1, \dot{x}(0) = 1\}$. And then, the first two IMFs of each input obtained by the EMD are collected as the labels. We refer to the inputs and their corresponding labels as Dataset_4

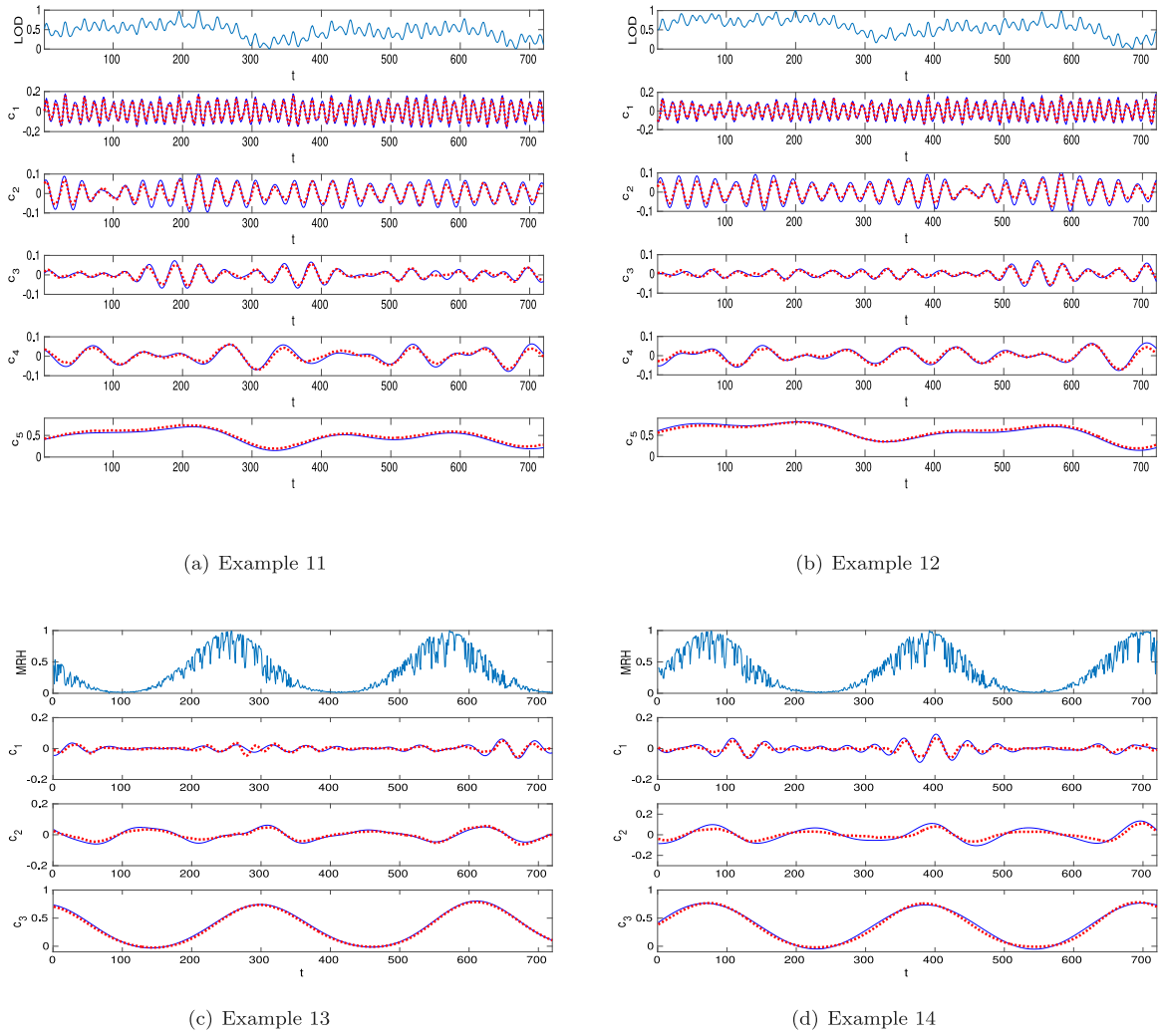
After the IRCNN is trained, we apply it to decompose the signal $\dot{x}(t)$ with $\alpha = -0.85, \beta = 1.05, \gamma = 0.045, \omega = 1.3$, which was not included in the training data. The results are depicted in Fig. 8 (a) . Both the resulting components produced by IRCNN are very close to those of EMD.

Example 10. The Lorenz equation is mathematically expressed as: $\dot{x} = -\sigma(x - y), \dot{y} = rx - y - xz, \dot{z} = -bz + xy$. We take into account the decomposition of $x(t)$. The inputs are the $x(t)$ achieved by the ode45 (code: https://www.mathwo rks.com/help/matlab/ref/ode45.html) method by changing the parameters $\sigma \in [9, 11]$ with step size 0.2, $r \in [19, 21]$ with step size 0.2, and $b \in [2, 5]$ with step size 1, where $t \in [0, 50]$ and initial conditions: $\{x(0) = -10, y(0) = 0, z(0) = 0\}$. Similarly to Example 9, we treat the first two IMFs of each input produced by the EMD as the labels. We call the inputs and labels Dataset_5. The results for signal $x(t)$ with parameters set to $\sigma = 10.5, r = 20.5, b = 3.5$, predicted by the trained IRCNN model are shown in Fig. 8 (b). The results show that the IRCNN has good learning and generalization capabilities for the solution to Lorenz equation, and can basically achieve the performance of EMD.

4.7. Is IRCNN capable of processing real signals?

We hereby employ the IRCNN model to process the real data, i.e., the length of day (LOD, data source: http://hpiers.obspm.fr/eoppc/eop/eopc04/eopc04.62-now, start date: Jan. 1,1962, end date: May 24, 2022), and the daily mean relative humidity (MRH, data source: https://data.gov.hk/en-data/dataset/hk-hko-rss-daily-mean-relative-humidity , start date: June 1, 1997, end date: Feb. 29, 2024) monitored at Hong Kong International Airport.

To generate the signals from LOD data for use as the inputs to train the IRCNN model, we first split the LOD into a series of length 720 (about two years) segments with a stride of 180 (about half year). And then, similar to the situation in which we were dealing with the solutions of differential equations, the main challenge in training the IRCNN model on real signals is the ground truth calibration. We use the EWT method to produce the labels for each segment, because it can produce an artificially pre-set number of components. For another IRCNN model built for the MRH data, its inputs and labels are constructed the



(a) Example 11

(b) Example 12

(c) Example 13

(d) Example 14

Fig. 9. Comparisons of the components obtained by EWT and IRCNN for LOD and MRH signals. (a-b) Top panel: the original LOD signals. The 2nd-6st panels: the obtained components by EWT (blue solid curve) and IRCNN (red dotted curve). (c-d) Top panel: the original MRH signals. The 2nd-4st panels: three components with seasonal oscillations by EWT (blue solid curve) and IRCNN (red dotted curve). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

same way as that for the LOD data. Compared with the LOD signal, for each MRH signal, the decomposition results by EWT contain some high frequency oscillations that can be interpreted as random perturbations, so we only retain the last three seasonal oscillation components as the labels here. The inputs and labels of LOD and MRH are referred to Dataset_6 and Dataset_7, respectively.

Example 11. After the IRCNN model for LOD is trained on Dataset_6, we apply it to the signal, which is the LOD data ranging from Dec. 12, 2018 to Nov. 30, 2020, and is excluded in the training dataset.

Example 12. We also apply the trained IRCNN model on Dataset_6 to another signal, which is the LOD data ranging from Dec. 7, 2019 to Nov. 25, 2021, and is also excluded in the training dataset.

Example 13. We first apply the IRCNN model trained on Dataset_7 to decompose the MRH signal ranging from April 12, 2018 to April 1, 2020, which is excluded in the training dataset.

Example 14. We further employ the trained IRCNN model to decompose another MRH signal ranging from Oct. 4, 2019 to Sep. 23, 2021, which is also excluded in the training dataset.

The components obtained from EWT and IRCNN for the signals in Examples 11–12, and three components with seasonal oscillations by EWT and IRCNN for the signals in Examples 13–14 are depicted in Fig. 9. In general, the plots prove that IRCNN can approximate EWT well.

4.8. Is IRCNN sensitive to the hyper-parameters?

This work selects the values of the hyper-parameters in IRCNN based on the performance on the validation set. To further explore the effect of the hyper-parameters in IRCNN on signal decomposition performance, we build different IRCNN models by changing the values of S (the number of recursion in the IRCNN inner loop block), K_1 (the kernel size of the first convolutional layer in the IRCNN inner loop block) and K_2 (the kernel size of the second convolutional layer in the IRCNN inner loop block), then train them on Dataset_1. To reduce the complexity of analysis, we assume that all IRCNN inner loop blocks have the same convolution kernel size.

The results measured by MAE and RMSE on the training and validation sets of Dataset_1 by setting S as 4, 8, 12, 16, 20 and K_1, K_2 as 32, 48, 64, 80 obtained from IRCNN are given in Table 16 and Fig. 10. According to them, we have the following findings:

Table 16
Performance of IRCNN, measured by (MAE, RMSE), on the training and validation sets of Dataset_1 by setting different values of the hyper-parameters S and K_1, K_2.

	S \ K_1,K_2	32	48	64	80	***Mean***
Training	4	(0.0969, 0.1535)	(0.0921, 0.1468)	(0.0918, 0.1438)	(0.0932, 0.1484)	*(0.0935, 0.1481)*
	8	(0.0740, 0.1256)	(0.0730, 0.1259)	(0.0682, 0.1190)	(0.0737, 0.1246)	*(0.0722, 0.1238)*
	12	(0.0733, 0.1231)	(0.0740, 0.1220)	(0.0704, 0.1212)	(0.0709, 0.1226)	*(0.0722, 0.1222)*
	16	(0.0673, 0.1203)	(0.0675, 0.1199)	(0.0686, 0.1218)	(0.0696, 0.1208)	(***0.0683***, ***0.1207***)
	20	(0.0706, 0.1245)	(0.0681, 0.1212)	(0.0679, 0.1201)	(0.0881, 0.1427)	*(0.0737, 0.1271)*
	Mean	*(0.0764,0.1294)*	*(0.0749, 0.1272)*	(***0.0734***, ***0.1252***)	*(0.0791, 0.1318)*	
Validation	4	(0.0960, 0.1527)	(0.0917, 0.1460)	(0.0939, 0.1458)	(0.0922, 0.1476)	*(0.0935, 0.1480)*
	8	(0.0782, 0.1335)	(0.0795, 0.1364)	(0.0775, 0.1329)	(0.0828, 0.1380)	*(0.0795, 0.1352)*
	12	(0.0790, 0.1334)	(0.0807, 0.1330)	(0.0783, 0.1340)	(0.0781, 0.1331)	*(0.0790, 0.1334)*
	16	(0.0747, 0.1319)	(0.0750, 0.1314)	(0.0763, 0.1339)	(0.0780, 0.1346)	(***0.0760***, ***0.1330***)
	20	(0.0758, 0.1337)	(0.0751, 0.1325)	(0.0772, 0.1335)	(0.0884, 0.1440)	*(0.0791, 0.1359)*
	Mean	*(0.0807, 0.1370)*	(***0.0804***, ***0.1359***)	*(0.0806, 0.1360)*	(0.0839, ***0.1359***)	

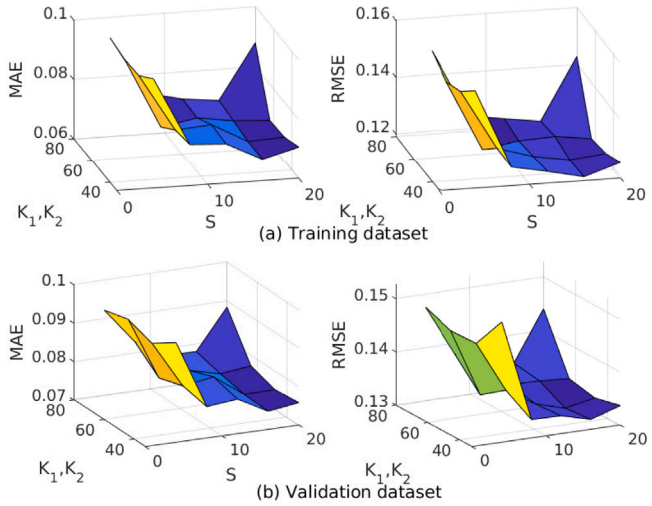


Fig. 10. Graphically show the effect of changing the values of S and K_1, K_2 on the decomposition performance of IRCNN, where both the training set and the validation set are derived from Dataset_1.

(i) The value of S has a relatively large influence on the decomposition performance, it states that IRCNN is sensitive to the number of recursion in IRCNN inner loop block. Specifically, if the value of S is too small, each IRCNN inner loop block cannot fully extract the average of its processed signal, if S is too large, it will also lead to poor decomposition due to excessive extraction. For Dataset_1, 16 is the best value for S.

(ii) Compared with S, the performance of the proposed method is not much influenced by the convolutional kernel sizes, i.e., K_1 and K_2 values. In particular, when the values of K_1, K_2 are large, it makes IRCNN adopt many parameters to extract the average of the processed signal flexibly. When they take small values, although the flexibility of IRCNN decreases with the reduction of parameters, since each inner loop block contains multiple convolutional layers, the receptive field can be increased to make up for its shortcoming. For Dataset_1, IRCNN performs well with K_1, K_2 taking as 48 or 64.

4.9. How does IRCNN perform in terms of generalization?

Since the performance of traditional signal decomposition methods generally depends on the parameters, the number of components obtained by them cannot be determined in advance, and many methods end up producing a bigger number of components compared to the ground truth ones. This requires often a post processing of the decomposition. In particular, some IMF components may contain purely noise and some others may correspond to a single frequency ground truth component that has been split into several IMFs. The comparison of these decompositions with the ideal ground truth one often requires manual handling. This makes the comparison of deep learn-based methods and classical signal processing decomposition methods performance challenging, especially when we try to verify the generalization ability of the trained model.

We plan to evaluate whether IRCNN is "overfitting" by comparing the gap in decomposition performance between the training set and the validation set. Generally speaking, if IRCNN is "overfitting", it will seriously affect its generalization ability. For the IRCNN models trained according to Dataset_1, Dataset_2, ..., Dataset_7 in Sections 4.3–4.7, respectively, their performance, measured by MAE and RMSE, on both training and validation sets are given in Table 17, and the percentage relative errors of MAE and RMSE on the training and test sets are shown in the last column of Table 17. It can be seen from the results that, except for Dataset_ 3 and Dataset_5, there is no obvious sign of overfitting in the IRCNN models trained by the other datasets, and the degree of overfitting in Dataset_5 is relatively slight.

4.10. How efficiently does IRCNN decompose signals?

Like other deep learning models, the IRCNN model is time-consuming in the training phase. But once it is trained, it is efficient in the test phase. Especially when processing multiple signals at the same time, deep learning models do not process each signal individually, but make predictions in parallel based on the size of the batch. This is, to the best of our knowledge, a unique feature compared to any other decomposition method available in the literature.

To compare the computational time of the proposed method with other methods, we apply them to decompose the signal in Example 6, and record the corresponding runtime for comparison. Moreover, we compare the running time of these methods when decomposing multiple signals. In this case, we no longer construct new signals to be decomposed, but just repeat the processing of the signal in Example 6, which does not affect the comparison of running time.

We depict the running time of different methods when decomposing different numbers of signals in Fig. 11. From it, we have the following findings: When decomposing only one signal, the computational efficiency of IRCNN ranks 6th out of the 14 methods. However, as the number of signals increases to 10, the computational efficiency of IRCNN begins to improve, and its ranking rises to 4th, surpassing the IF and VMD methods; when the number increases to 100, its computational efficiency ranks 2nd, beyond the EMD and DCT_GAS_FDM methods. Although there is still a gap between IRCNN and the EWT method that ranks 1st in computational efficiency, it can be seen that as the number of signals increases, the gap between them shrinks significantly.

Table 17
Performance of IRCNN on the training and validation sets of different datasets.

Dataset	Training (MAE, RMSE)	Validation (MAE, RMSE)	Percentage relative error
Dataset_1	(0.0522, 0.0919)	(0.0403, 0.0705)	(−22.8%, −23.3%)
Dataset_2	(0.1509, 0.2545)	(0.1565, 0.2630)	(3.7%, 3.3%)
Dataset_3	(0.0106, 0.0178)	(0.0142, 0.0239)	(35.8%, 34.3%)
Dataset_4	(0.0087, 0.0112)	(0.0086, 0.0111)	(−1.1%, −0.9%)
Dataset_5	(0.0234, 0.0627)	(0.0265, 0.0729)	(13.2%, 16.3%)
Dataset_6	(0.0944, 0.0681)	(0.0900, 0.0675)	(−4.7%, −0.9%)
Dataset_7	(0.0291, 0.0304)	(0.0308, 0.0321)	(5.8%, 5.6%)

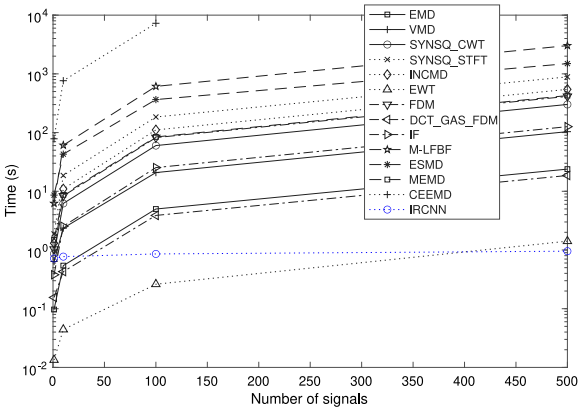


Fig. 11. Comparison of computational time of different signal decomposition methods when decomposing different numbers of signals.

5. Conclusion

In the paper, we use deep learning techniques to tackle the issue of decomposing a non-stationary signal into oscillatory components. Specifically, we first construct the IRCNN inner loop block for obtaining the local average of a given signal, and then these blocks are cascaded into a deeper network, called IRCNN, which is used to decompose a given signal.

IRCNN is a supervised and deep-learning-based model, it has the advantages of these two types of models. First, the convolutional kernel weights of IRCNN are learnt according to the input signals, which makes the proposed method more adaptive. Second, some common tools in deep learning, like the residual structure and the nonlinear activation function, can be added to increase the expressive ability and flexibility of IRCNN. Third, IRCNN can be customized according to the application. For example, when processing signals composed of orthogonal components, an inner product term can be added to the loss function to enhance the orthogonality of the derived components. To verify the performance of the proposed model, we compare it with other existing models from ten aspects. And the artificial and real data are used in the experiments. All results show that IRCNN works better in handling the boundaries, mode mixing effects and the orthogonality of the decomposed components, and is more robust than the existing methods.

However, IRCNN has the common limitations of supervised deep learning models. Theoretically speaking, deep learning models generally have deficiencies such as immature mathematical theoretical analysis and poor interpretability of models. From the technical point of view, first, the labels must be given in advance in the training phase. This makes it difficult for IRCNN to flexibly process real signals because their labels are hard to obtain. Second, almost all structures in the IRCNN model are limited to the aggregation and extraction of information in the signal time domain, and the frequency domain information is not considered, which cannot be ignored in the time–frequency analysis of non-stationary signals. In addition, in the field of non-stationary signal decomposition, there is a lack of a publicly available dataset to objectively evaluate the performance of decomposition methods so far. Almost all of these methods are based on picking typical signals to evaluate their effectiveness.

In the future, we will improve this work by working in the following three directions. First, we will improve IRCNN by revising its network to incorporate the time and frequency information of the signal. Second, we will publish a dataset of orthogonal, noisy, and jumping types of signals, as well as some signals from real-world applications, to establish a set of data standards for the field of non-stationary signal decomposition. Furthermore, given that the main limitation of the IRCNN approach is that it requires other decomposition methods to calibrate the ground truths before the training phase, we plan to work in the future on the development of an unsupervised learning method to produce a new technique with similar performance to the IRCNN algorithm, but that does not require any training based on other techniques.

CRediT authorship contribution statement

Feng Zhou: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Methodology, Funding acquisition, Data curation, Conceptualization. **Antonio Cicone:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Formal analysis. **Haomin Zhou:** Writing – review & editing, Supervision, Project administration, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

F. Zhou research was partially supported by the National Natural Science Foundation of China [grant number 11901113, 12171488], the Guangdong Province philosophy and social science planning project, China [grant number GD23XGL017], and the China Scholarship Council, China [grant number 202008440024]. A. Cicone is member of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM), was partially supported through the GNCS-INdAM Project, Italy, CUP_E53C22001930001 and CUP_E53C23001670001, and thanks European Union for the financial support through “Next Generation EU” and the Italian Ministry of the University and Research for the financial support under the PRIN PNRR 2022 grant number E53D23018040001 ERC field PE1 project P2022XME5P titled “Circular Economy from the Mathematics for Signal Processing prospective”, and the Ministry of Foreign Affairs and the International Cooperation (MAECI), Italy for the financial support, CUP E13C24000350001, under the “Grande Rilevanza” Italy – China Science and Technology

Cooperation Joint Project titled "sCHans – Solar loading infrared thermography and deep learning teCHniques for the noninvAsive iNSpection of precious artifacts", PGR02016. H. Zhou research was partially supported by the National Science Foundation of US [grant number DMS-1830225], and the Office of Naval Research, United States [grant number N00014-18-1-2852].

References

[1] N.E. Huang, Z. Shen, S.R. Long, M.C. Wu, H.H. Shih, Q. Zheng, N.C. Yen, C.C. Tung, H.H. Liu, The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 454 (1971) (1998) 903–995.
[2] M. Thilagaraj, M.P. Rajasekaran, An empirical mode decomposition (EMD)-based scheme for alcoholism identification, Pattern Recognit. Lett. 125 (2019) 133–139.
[3] J.S. Smith, The local mean decomposition and its application to EEG perception data, J. R. Soc. Interface 2 (5) (2005) 443–454.
[4] E. Deléchelle, J. Lemoine, O. Niang, Empirical mode decomposition: an analytical approach for sifting process, IEEE Signal Process. Lett. 12 (11) (2005) 764–767.
[5] S.D. El Hadji, R. Alexandre, A.-O. Boudraa, Analysis of intrinsic mode functions: A PDE approach, IEEE Signal Process. Lett. 17 (4) (2009) 398–401.
[6] H. Hong, X. Wang, Z. Tao, Local integral mean-based sifting for empirical mode decomposition, IEEE Signal Process. Lett. 16 (10) (2009) 841–844.
[7] A. Cicone, E. Pellegrino, Multivariate fast iterative filtering for the decomposition of nonstationary signals, IEEE Trans. Signal Process. 70 (2022) 1521–1531.
[8] G. Tu, X. Dong, S. Chen, B. Zhao, L. Hu, Z. Peng, Iterative nonlinear chirp mode decomposition: A Hilbert-Huang transform-like method in capturing intra-wave modulations of nonlinear responses, J. Sound Vib. 485 (2020) 115571.
[9] X. Guanlei, W. Xiaotong, X. Xiaogang, On analysis of bi-dimensional component decomposition via BEMD, Pattern Recognit. 45 (4) (2012) 1617–1626.
[10] S. Peng, W.-L. Hwang, Null space pursuit: An operator-based approach to adaptive signal separation, IEEE Trans. Signal Process. 58 (5) (2010) 2475–2483.
[11] T. Oberlin, S. Meignen, V. Perrier, An alternative formulation for the empirical mode decomposition, IEEE Trans. Signal Process. 60 (5) (2012) 2236–2246.
[12] T.Y. Hou, Z. Shi, Adaptive data analysis via sparse time-frequency representation, Adv. Adapt. Data Anal. 3 (01n02) (2011) 1–28.
[13] N. Pustelnik, P. Borgnat, P. Flandrin, A multicomponent proximal algorithm for empirical mode decomposition, in: Proceedings of the 20th European Signal Processing Conference, EUSIPCO, IEEE, 2012, pp. 1880–1884.
[14] N. Pustelnik, P. Borgnat, P. Flandrin, Empirical mode decomposition revisited by multicomponent non-smooth convex optimization, Signal Process. 102 (2014) 313–331.
[15] K. Dragomiretskiy, D. Zosso, Variational mode decomposition, IEEE Trans. Signal Process. 62 (3) (2013) 531–544.
[16] N. ur Rehman, H. Aftab, Multivariate variational mode decomposition, IEEE Trans. Signal Process. 67 (23) (2019) 6039–6052.
[17] F. Zhou, L. Yang, H. Zhou, L. Yang, Optimal averages for nonlinear signal decompositions—another alternative for empirical mode decomposition, Signal Process. 121 (2016) 17–29.
[18] I. Daubechies, J. Lu, H.-T. Wu, Synchrosqueezed wavelet transforms: An empirical mode decomposition-like tool, Appl. Comput. Harmon. Anal. 30 (2) (2011) 243–261.
[19] J. Gilles, Empirical wavelet transform, IEEE Trans. Signal Process. 61 (16) (2013) 3999–4010.
[20] P. Singh, S.D. Joshi, R.K. Patney, K. Saha, The Fourier decomposition method for nonlinear and non-stationary time series analysis, Proc. R. Soc. A: Math., Phys. Eng. Sci. 473 (2199) (2017) 20160871.
[21] P. Singh, Novel Fourier quadrature transforms and analytic signal representations for nonlinear and non-stationary time-series analysis, R. Soc. Open Sci. 5 (11) (2018) 181131.
[22] Z. Wang, C.M. Wong, A. Rosa, T. Qian, F. Wan, Adaptive Fourier decomposition for multi-channel signal analysis, IEEE Trans. Signal Process. 70 (2022) 903–918.
[23] M. Xu, S. Yoon, A. Fuentes, D.S. Park, A comprehensive survey of image augmentation techniques for deep learning, Pattern Recognit. 137 (2023) 109347.
[24] D. Khurana, A. Koli, K. Khatter, S. Singh, Natural language processing: State of the art, current trends and challenges, Multimedia Tools Appl. 82 (2023) 3713–3744.
[25] M.A. Bansal, D.R. Sharma, D.M. Kathuria, A systematic review on data scarcity problem in deep learning: solution and applications, ACM Comput. Surv. 54 (10s) (2022) 1–29.
[26] W. Zhu, S.M. Mousavi, G.C. Beroza, Seismic signal denoising and decomposition using deep neural networks, IEEE Trans. Geosci. Remote Sens. 57 (11) (2019) 9476–9488.
[27] X. Zheng, W. Chen, Y. You, Y. Jiang, M. Li, T. Zhang, Ensemble deep learning for automated visual classification using EEG signals, Pattern Recognit. 102 (2020) 107147.
[28] L. Ding, Z. Chen, Y. Pan, B. Song, Mine microseismic time series data integrated classification based on improved wavelet decomposition and ELM, Cogn. Comput. 14 (2022) 1526–1546.
[29] X. Song, P. Sun, S. Song, V. Stojanovic, Finite-time adaptive neural resilient DSC for fractional-order nonlinear large-scale systems against sensor-actuator faults, Nonlinear Dynam. 111 (13) (2023) 12181–12196.
[30] X. Song, P. Sun, S. Song, V. Stojanovic, Quantized neural adaptive finite-time preassigned performance control for interconnected nonlinear systems, Neural Comput. Appl. 35 (21) (2023) 15429–15446.
[31] D. Zhang, W. Ding, B. Zhang, C. Liu, J. Han, D. Doermann, Learning modulation filter networks for weak signal detection in noise, Pattern Recognit. 109 (2021) 107590.
[32] S. Bubeck, Convex optimization: Algorithms and complexity, Found. Trends Mach. Learn. 8 (3–4) (2015) 231–357.
[33] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[34] P. Ablin, J.-F. Cardoso, A. Gramfort, Faster ICA under orthogonal constraint, in: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2018, pp. 4464–4468.
[35] B. Gao, X. Liu, X. Chen, Y. x. Yuan, A new first-order algorithmic framework for optimization problems with orthogonality constraints, SIAM J. Optim. 28 (1) (2018) 302–332.
[36] N.E. Huang, Z. Shen, S.R. Long, A new view of nonlinear water waves: the Hilbert spectrum, Annu. Rev. Fluid Mech. 31 (1999) 417–457.
[37] J.-L. Wang, Z.-J. Li, Extreme-point symmetric mode decomposition method for data analysis, Adv. Adapt. Data Anal. 5 (3) (2013) 1350015.
[38] N.E.H. Jia-rong Yeh, Jiang-shing Shieh, Complementary ensemble empirical mode decomposition: A novel noise enhanced data analysis method, Adv. Adapt. Data Anal. 2 (2) (2010) 135–156.
[39] N. Rehman, D.P. Mandic, Multivariate empirical mode decomposition, Proc. R. Soc. A: Math., Phys. Eng. Sci. 466 (2117) (2010) 1291–1302.
[40] Z. Wu, N.E. Huang, Ensemble empirical mode decomposition: A noise-assisted data analysis method, Adv. Adapt. Data Anal. 1 (1) (2009) 1–41.

Feng Zhou received the Ph.D. degree in Computational Mathematics from Sun Yat-sen University, China, in 2015. Now he is an Associate Professor at the School of Information Science, Guangdong University of Finance and Economics. His research interests include signal processing, deep learning and ensemble learning.

Antonio Cicone is an Associate Professor in Numerical Analysis at the University of L'Aquila. His research is at the intersection of Numerical Linear Algebra and Signal Processing. He works to the development, mathematical analysis and application of new algorithms for the decomposition and time–frequency analysis of signals.

Haomin Zhou is a professor in Mathematics at Georgia Tech. He received his B.S. in mathematics from Peking University in 1991, his M.Phil from CUHK and Ph.D. from UCLA, both in applied mathematics in 1996 and 2000 respectively. He was a postdoc at Caltech before joining Georgia Tech in 2003.