WHEN: A Wavelet-DTW Hybrid Attention Network for Heterogeneous Time Series Analysis

Jingyuan Wang
School of Computer Science and Engineering, Beihang University
Key Laboratory of Data Intelligence and Management (Beihang University), MIIT
Beijing, China

Chen Yang
Xiaohan Jiang
School of Computer Science and Engineering, Beihang University
Beijing, China

Junjie Wu*
School of Economics and Management, Beihang University
Key Laboratory of Data Intelligence and Management (Beihang University), MIIT
Beijing, China

ABSTRACT

Given its broad applications, time series analysis has gained substantial research attention but remains a very challenging task. Recent years have witnessed the great success of deep learning methods, *e.g.,* CNN and RNN, in time series classification and forecasting, but *heterogeneity* as the very nature of time series has not yet been addressed adequately and remains the performance "treadstone". In this light, we argue that the *intra-sequence nonstationarity* and *inter-sequence asynchronism* are two types of heterogeneities widely existed in multiple times series, and propose a hybrid attention network called WHEN as deep learning solution. WHEN features in two attention mechanisms in two different modules. In the WaveAtt module, we propose a novel data-dependent wavelet function and integrate it into the BiLSTM network as the *wavelet attention*, for the purpose of analyzing dynamic frequency components in nonstationary time series. In the DTWAtt module, we transform the dynamic time warping (DTW) technique into the form as the *DTW attention*, where all input sequences are synchronized with a universal parameter sequence to overcome the time distortion problem in multiple time series. WHEN with the hybrid attentions is then formulated as task-dependent neural network for either classification or forecasting tasks. Extensive experiments on 30 UEA datasets and 3 real-world datasets with rich competitive baselines demonstrate the excellent performance of our model. The ability of WHEN in dealing with time series heterogeneities is also elaborately explored via specially designed analysis.

CCS CONCEPTS

• **Mathematics of computing** → **Time series analysis**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Time Series, Wavelet, Dynamic Time Warping, Attentions

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '23, August 6–10, 2023, Long Beach, CA, USA.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599549

ACM Reference Format:
Jingyuan Wang, Chen Yang, Xiaohan Jiang, and Junjie Wu. 2023. WHEN: A Wavelet-DTW Hybrid Attention Network for Heterogeneous Time Series Analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599549

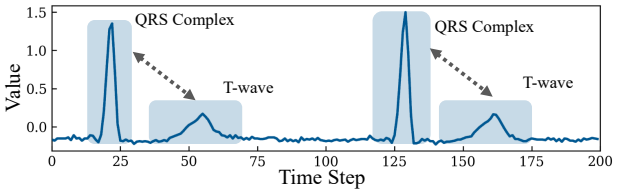
1 INTRODUCTION

Time series data analysis, including time series classification (TSC) and time series forecasting (TSF), is a long-standing and critically important task for industrial, medical, business, and scientific applications [17]. While tremendous algorithms have been proposed elaborately for time series analysis (TSA), high-performance classification and forecasting are still the targets particularly sought by academia and industries, especially when facing newly emerging real-world hypercomplex and highly dynamic time series.

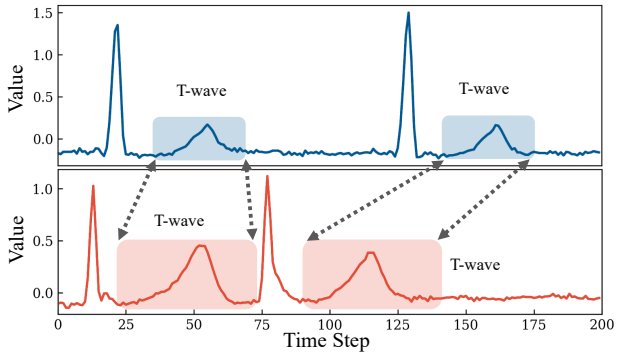
Traditional algorithms for time series analysis usually create hand-crafted features or distances based on original time series and apply machine learning models to classification or forecasting, such as ARIMA with series differences for TSF [27], and k-NN with DTW distance for TSC [34]. Besides, some frequency transform and frequent pattern mining methods, such as Fourier transform [51], wavelet transform [32] and shapelet [45], can also extract shallow features from time series for subsequent classification or forecasting applications.

In recent years, with the booming of deep learning concept, various types of deep neural network models have been introduced to TSA and achieved great success [22]. Compared with the abovementioned shallow feature based methods, deep learning models can automatically learn complex nonlinear features from large-scale and high-dimensional time series data, and therefore gain drastic attention and are widely used in real-world applications. For example, Recurrent Neural Networks (RNN) that are originally designed for general sequential data processing have been employed for time series forecasting [56]. While embodying some advantages, these methods are not designed purposely for time series, and therefore might not fully exploit the potential of deep learning in TSA. Particularly, the very nature of time series, *i.e.,* the *heterogeneity* along the timesteps or among different dimensions, has not been modeled adequately, which indeed motives our study.

In light of this, we propose to explore the heterogeneity of real-life time series and design customized deep learning models for high-performance TSA. We argue that two intractable and widely



(a) Intra-sequence nonstationarity in a time series.



(b) Inter-sequence asynchronism between two time series.

Figure 1: Example of two types of heterogeneities in ECG signals (source: AtrialFibrillation in UEA datasets).

existed phenomena, namely *Intra-sequence Nonstationarity* and *Inter-sequence Asynchronism*, embody important heterogeneities of multiple time series. The intra-sequence nonstationarity originates from the fact that different parts of a time series have inherently heterogeneous properties, such as mean, variance, frequency components, *etc.* Fig. 1(a) gives an example about the ECG time series, where the QRS-complex parts (with high-change frequencies and amplitudes) and the T-wave parts (with low-change frequencies and amplitudes) have obviously heterogeneous frequency components. Traditional deep models often use the same repetitive structures, such as repetitive convolution kernels in CNN and recursive units in RNN, to process heterogeneous components of time series and thus cannot deal with this problem well. The *Inter-sequence Asynchronism* refers to the out-of-sync phenomenon among time series, which might be caused by heterogeneous sampling rate or phase perturbation. As the example shown in Fig. 1(b), the two ECG time series in a same class are very likely to be categorized into different classes for having different sampling rates and initial phases. This is a common phenomenon in time series classification, and traditionally, can be adjusted using the Dynamic Time Warping (DTW) algorithm [10]. However, in deep learning models we yet have no special mechanism to deal with this problem.

To meet the above challenges, we propose the so-called Wavelet-DTW Hybrid attEntion Networks (WHEN) for multiple heterogeneous time series analysis. WHEN is essentially a hybrid attention network that integrates both wavelet transform and the DTW algorithm as attentions for heterogeneity learning. The framework of WHEN is shown in Fig. 2, which contains two core modules. The key component of the WaveAtt module is the novel *Data-dependent Wavelet Attention*, where a wavelet function with data-dependent frequency band parameters is integrated with the BiLSTM network as an attention. This module can reduce the problem of intra-sequence nonstationarity through dynamically extracting heterogeneous frequency components of input sequences. The DTWAtt module features in a *Local Dynamic Time Warping Attention*, where the DTW algorithm is implemented as an attention to synchronize the input sequences with a universal feature sequence and a multi-head attention. This module is proposed to handle the inter-sequence asynchronism problem. The two modules are connected as a pipeline using task-dependent neural networks (TD-NN-1 and TD-NN-2) for either time series classification or forecasting tasks in an end-to-end manner.

The superiority of WHEN is verified empirically over 30 standard time series datasets for time series classification and over 3 real-world datasets for forecasting, with the presence of rich competitive baselines. To the best of our knowledge, WHEN is the first study that considers both nonstationarity and asynchronism heterogeneous problems of time series analysis within a deep leaning framework.

2 RELATED WORK

Heterogeneous Time Series Analysis. In this paper, we consider two types of heterogeneous problems in time series analysis, *i.e.,* non-stationary and asynchronous. Some previous works also study those problems. The traditional approaches to handle the non-stationary problems is using handcrafted transform methods to convert non-stationary time series as stationary statistics, such as time series detrending method [33], time series differencing [20], mapping-based models [27, 46, 67], splitting-based models [2, 3, 61] *etc.* However, for complex, high dimensional and noisy real-world time series data, the non-stationary features could be heterogeneous and dynamic. It is very hard to deal with these challenging conditions for the static handcrafted methods. Deep sequential models such as LSTM [50, 56] and Transformer [60] can automatically learn representation from complex sequential data, therefore have great potential to overcome the shortage of handcrafted methods in non-stationary time series analysis. In recent works, NsTKA [42] proposed a kernelized attention method to solve temporal event prediction problem of non-stationary time series. mWDN incorporates the multilevel discrete Wavelet decomposition with deep neural networks to achieve frequency analysis in complex time series [65]. For the asynchronous problem, the traditional approach is using the DTW algorithms to rectify the asynchronism among time series, however, which is not compatible to the deep learning. To the best of our knowledge, WHEN is the first deep learning model that is designed purposefully for both non-stationary and asynchronous problem in the heterogeneous time series analysis.

Time Series Classification (TSC). As a major type of time series analysis task, traditional TSC methods include three major categories: distance based methods, feature based methods and ensemble methods. Dynamic time warping (DTW) and its variants [10, 34] are the most popular kind of distance based classification methods, combined with the k-Nearest Neighbor algorithm. Feature based methods consider special features for time series including bag of patterns [40], bag of SFA symbols [51], convolution kernel features [19, 59] and time series forest [8], *etc.* Ensemble methods aim to combine the advantages of different methods, including

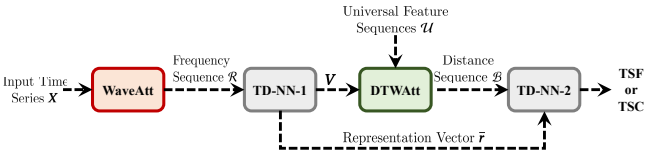


Figure 2: Framework of the WHEN model.

elastic ensemble [41], COTE [5], *etc.* Traditional methods usually depend on handcrafted features such as distance and difference, and therefore are not adequate for large and complex datasets. In the recent, deep neural networks begin to automatically learn complex features in TSC, including supervised feature mining [24], unsupervised feature learning [23], transformer methods [15, 48, 49, 68], *etc.* Although deep learning methods can generate abundant features for classification with the aid of its representational learning ability, some unique features in time series such as warping distance and frequency cannot be explicitly learned by neural network automatically.

Time Series Forecasting (TSF). The major function of TSF models is exploiting autoregressive correlations in time series. A classic TSF model for non-stationary time series is ARIMA [27], which considers detrending with a moving average (MA) process and differences time series with an order of differences. Frequency analysis is also an important tool that is usually used in TSF models [57]. Recently, deep learning has become the state-of-the-art method in time series forecasting [1, 12, 18, 28–31, 44, 47, 53, 54, 62–64, 66, 70]. Dynamic programming based differentiable DTW methods are also used in TSF task [16, 36–38], while our DTWAtt uses attention form differentiable DTW which better suits the deep learning environment. Similar as in the deep learning based TSC models, there are rare works that can incorporate heterogeneous time series targeted features with deep learning as the proposed model in this work.

3 DATA-DEPENDENT WAVELET ATTENTION

In this section, we introduce one key component of our WHEN model: *WaveAtt*, which is a wavelet attention enabled representation learning module with BiLSTM as the basic framework.

3.1 Wavelet-based Frequency Analysis

The intra-sequence nonstationarity problem is fundamentally caused by the different parts of a time series having different inherent properties. Frequency is one of the most important inherent property in time series. Theoretically, any time series could be decomposed into a set of frequency components. For a time series with intra-sequence nonstationarity, its frequency components are expected to be diverse and thus could be disclosed by frequency analysis. We therefore adopt a classic frequency analysis tool, *i.e.*, the *Wavelet Transform* [26], to analyze the nonstationarity of time series. In this subsection, we first give a brief introduction to wavelet transform.

The wavelet transform expresses a sequential signal by a set of wavelet bases, which are generated from a basic local frequency function, *i.e.*, the mother wavelet function. Given a mother wavelet function $\psi(t)$, the bases are defined as

$$\psi_{\alpha,\tau}(t) = \frac{1}{\sqrt{\alpha}}\psi\left(\frac{t-\tau}{\alpha}\right), \tag{1}$$

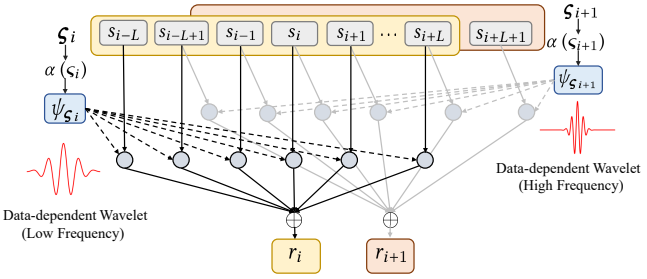


Figure 3: Model procedure of WaveAtt.

where $\alpha \in \mathbb{R}^+$ is a dilated scalar, which controls the frequency band ($\frac{1}{\alpha}$) of a basis. $\tau \in \mathbb{R}$ is a shift, which translates the mother function from t to $t-\tau$, controlling the location of the basis to extract frequency information. Different types of mother functions can be used to generate the wavelet bases, and each defines a wavelet family. The typical mother wavelet functions include Haar, Mexica hat, Coiflets, Meyer, *etc* [43].

Given a sequential signal $f(t)$ and the base $\psi_{\alpha,\tau}(t)$, the wavelet transform extracts its frequency component as

$$r_{\alpha,\tau} = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{\alpha}}\psi\left(\frac{t-\tau}{\alpha}\right)\mathrm{d}t, \tag{2}$$

where $r_{\alpha,\tau}$ is the component intensity of the frequency band $\frac{1}{\alpha}$ at the location τ. In this way, we can extract the component of a sequential signal for any frequency at any location by adjusting the parameters α and τ.

While wavelet transform can extract important frequency information of a time series, it cannot be directly applied to our model for non-stationary time series analysis. The first challenge comes from the intra-series nonstationarity of most real-life time series data, which have diverse frequencies that are essentially dynamic over time. This implies that we'd better set the frequency band parameter α dynamically for *each* time step, but the frequency band given in Eq. (2) is fixed for the entire time series and cannot well capture the dynamics in frequency. Traditional methods often set α manually to go through all possible integers for full frequency bands coverage, which is obviously inefficient. The second challenge comes from the end-to-end learning purpose of our model, which requires to integrate the wavelet transform into a general deep learning framework. We address these below.

3.2 Data-dependent Wavelet Attention for Dynamic Frequency Analysis

Here, we devise a novel *data-dependent wavelet attention* mechanism (WaveAtt) for dynamic frequency analysis. As mentioned in Eq. (1), the parameter α controls the frequency band of wavelet transform, which is often set manually to go through all possible integers for full frequency bands coverage. As a result, many meaningless frequency components might be extracted. As shown in Fig. 3, in our WaveAtt module, we propose a data-dependent mechanism to adaptively adjust α for different steps of time.

The input of the WaveAtt module is a multivariate time series defined as $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_I)^\top$, where $\boldsymbol{x}_i \in \mathbb{R}^{K_X}$ is a K_X dimensional vector. We adopt the Bidirectional Long Short Term

Memory (BiLSTM) [25] network to encode $\boldsymbol{X}$ as a hidden state sequence $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_i, \ldots, \boldsymbol{s}_I)$. The major benefit of BiLSTM is that it can capture both forward and backward temporal dependencies, while unidirectional RNN models can only capture the forward temporal dependencies. Such a benefit is important for many time series applications especially for our wavelet attentions since the temporal dependencies in frequency components are not only unidirectional. Moreover, with the BiLSTM encoding layer, the correlations among different dimensions of the input sequence $\boldsymbol{X}$ could also be exploited.

The input of WaveAtt is the K_S dimensional sequence $\boldsymbol{S}$ generated by the BiLSTM layer. We denote the k-th dimension of $\boldsymbol{S}$ as $\boldsymbol{\varsigma}^{(k)} = \left(s_1^{(k)}, \ldots, s_i^{(k)}, \ldots, s_I^{(k)}\right)$. Given a sliding window of length $2L+1$, we extract a sub-sequence of $\boldsymbol{\varsigma}^{(k)}$ centered on time step i as

$$\boldsymbol{\varsigma}_i^{(k)} = \left(s_{i-L}^{(k)}, \ldots, s_{i-1}^{(k)}, s_i^{(k)}, s_{i+1}^{(k)}, \ldots, s_{i+L}^{(k)}\right), \tag{3}$$

which is the basic unit for dynamic frequency analysis in WaveAtt. In what follows, we omit the dimension mark k in variables for concise description when there is no conflict.

We then set the parameter α as a function of $\boldsymbol{\varsigma}_i$ as follows:

$$\alpha(\boldsymbol{\varsigma}_i) = \mathrm{ReLU}\left(w_b + \sum_{l=-L}^{L} w_l \cdot s_{i+l}\right) + \epsilon, \tag{4}$$

where the vectors $\boldsymbol{w} = (w_{-L}, \ldots, w_L, w_b)^\top$ are trainable parameters. $\mathrm{ReLU}(\cdot)$ is a Rectified Linear Unit activation function and ϵ is a small value so that α will be positive. Next, we define a *data-dependent wavelet function* to extract frequency information from the subsequence $\boldsymbol{\varsigma}_i$, which is given by

$$\psi_{\boldsymbol{\varsigma}_i}(t) = \frac{1}{\sqrt{\alpha\left(\boldsymbol{\varsigma}_i\right)}} \psi\left(\frac{t-i}{\alpha\left(\boldsymbol{\varsigma}_i\right)}\right). \tag{5}$$

Compared with the standard wavelet mother function in Eq. (1), the data-dependent wavelet has two obvious differences. One is to replace the fixed α by the data dependent $\alpha\left(\boldsymbol{\varsigma}_i\right)$, and the other is to set $\tau = i$ to shift the center of the mother wavelet function to the i-th time step. These together enable the extraction of data-dependent frequency information for each time step.

The purpose of setting α as a function of $\boldsymbol{\varsigma}_i$ is to let the frequency band of the wavelet function varies adaptively with the input sequential data. Meanwhile, the model only extracts the most suitable band rather than full-band frequency components. For different input data in a sliding window, the frequency bands are also different, so we call the wavelet function in Eq. (5) as *data-dependent wavelet*. This benefit is very important for analyzing non-stationary time series with heterogeneous frequency components. Moreover, the dilated scalar $\alpha\left(\boldsymbol{\varsigma}_i\right)$ can also be a non-integer, which can further improve the precision of frequency analysis.

After generating the data-dependent Wavelet function $\psi_{\boldsymbol{\varsigma}_i}(t)$ using Eq. (5), we use it to generate a group of attention weights as

$$\mathrm{ATT}\left(\psi_{\boldsymbol{\varsigma}_i}(t)\right) = \frac{\psi_{\boldsymbol{\varsigma}_i}(t)}{\sum_{\tau=i-L}^{i+L} \left|\psi_{\boldsymbol{\varsigma}_i}(\tau)\right|}. \tag{6}$$

Next, the attentions are integrated with the BiLSTM's outputs as

$$r_i = \sum_{t=i-L}^{i+L} \mathrm{ATT}\left(\psi_{\boldsymbol{\varsigma}_i}(t)\right) \cdot s_t, \tag{7}$$

which could be considered as attention form expression of Eq. (2). The output r_i is the frequency component of $\boldsymbol{\varsigma}_i^{(k)}$ extracted by $\psi_{\boldsymbol{\varsigma}_i}(t)$. Since Eq. (7) could be considered as paying attention to $\boldsymbol{\varsigma}_i$ using wavelet frequency as weights, we call it the *Wavelet Attention*.

We denote $r_i^{(k)}$ as the frequency component of $\boldsymbol{\varsigma}_i^{(k)}$. For the K_s dimensions of $\boldsymbol{s}_i$, we have a frequency component vector $\boldsymbol{r}_i = \left(r_i^{(1)}, \ldots, r_i^{(K_S)}\right)^\top$. In order to increase the diversity of frequency components, we adopt multiple wavelet families to implement the data-dependent wavelet function. Given Γ wavelet families, the frequency components at time step i grow into a matrix $\boldsymbol{R}_i = \left(\boldsymbol{r}_{i,1}, \ldots, \boldsymbol{r}_{i,\Gamma}\right)$. As a result, the matrix sequence $\mathcal{R} = (\boldsymbol{R}_1, \ldots, \boldsymbol{R}_i, \ldots, \boldsymbol{R}_I)$ is the final output of WaveAtt in our model.

4 DYNAMIC TIME WARPING ATTENTION

In this section, we introduce another key component of our WHEN model: *DTWAtt*, which is essentially a neuralized dynamic time warping to deal with the asynchronism problem among time series.

4.1 Dynamic Time Warping

We begin by briefly introducing *Dynamic Time Warping* (DTW), which is an algorithm for measuring similarity between two temporal sequences. It can align two temporal sequences non-linearly in time dimension, and therefore is very suitable to two asynchronous time series with different phase positions and sampling rates.

To keep the notations consistent, we use sequences of vectors with a same number of dimensions as example to explain the DTW algorithm. Given two sequences $\boldsymbol{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m, \ldots \boldsymbol{p}_M)$, $\boldsymbol{Q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n, \ldots, \boldsymbol{q}_N)$, the DTW algorithm [55] calculates a *warping path* as a sequence whose elements are pairs of vectors in $\boldsymbol{p}$ and $\boldsymbol{q}$,

$$H = \left(\left(\boldsymbol{p}_{m_1}, \boldsymbol{q}_{n_1}\right), \ldots, \left(\boldsymbol{p}_{m_z}, \boldsymbol{q}_{n_z}\right), \ldots, \left(\boldsymbol{p}_{m_Z}, \boldsymbol{q}_{n_Z}\right)\right), \tag{8}$$

where the subscript indexes satisfy: $m_1 = n_1 = 1$; $m_Z = M$, $n_Z = N$; $0 \le m_{z+1} - m_z \le 1$, $0 \le n_{z+1} - n_z \le 1$. A warping path can also be viewed as a path from position $(1,1)$ to (M,N) in a $M \times N$ matrix.

Given the warping path H of the sequences $\boldsymbol{p}$ and $\boldsymbol{q}$, the distance between the two sequences in DTW is calculated as

$$d_H = \sum_{z=1}^{Z} \left\|\boldsymbol{p}_{m_z} - \boldsymbol{q}_{n_z}\right\|, \tag{9}$$

where $\|\cdot\|$ is the l-2 norm. Let $\mathcal{H}$ be all possible warping paths between $\boldsymbol{P}$ and $\boldsymbol{Q}$. The DTW algorithm finds the path H^* with the shortest distance between the two sequences, *i.e.,*

$$H^* = \arg\min_{H \in \mathcal{H}} d_H. \tag{10}$$

The distance of H^*, *i.e.,* $d^* = d_{H^*}$, is a measurement of similarity between $\boldsymbol{P}$ and $\boldsymbol{Q}$.

The advantage of warping path is its ability in aligning samples of two sequences in a non-linear way. For a pair $(\boldsymbol{p}_{m_z}, \boldsymbol{q}_{n_z})$ in a warping path H, the two elements have the same order index z but could have different temporal indexes, *i.e.,* $m_z \neq n_z$. For example, for a sub-sequence $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4)$, its counterpart in warping path could be $(\boldsymbol{q}_1, \boldsymbol{q}_1, \boldsymbol{q}_1, \boldsymbol{q}_2)$. In this way, the time axis of the sequence $\boldsymbol{P}$ looks like be "warped" from the view of the sequence $\boldsymbol{Q}$. Moreover, the warping path could also be generated from two unequal length sequences, *e.g.,* a sequence and its down sampling sequence. This feature is very useful for handling the asynchronism problem

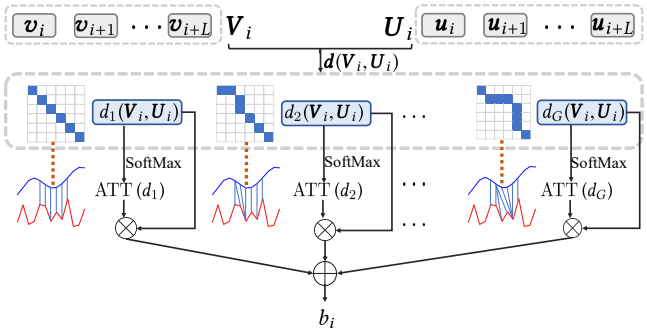


Figure 4: Model procedure of DTWAtt.

between different time series caused by dynamic sampling rate or phrase disturbance.

DTW distance is particularly valuable to reduce the modelling errors derived from asynchronism among time series. Recall the two example sequences in Figure 1(b), they are similar in shape but with different time scales and thus might be misclassified, *e.g.,* by the simple k-Nearest Neighbor (KNN) model, into different classes. With the DTW distance, however, the situation could be quite opposite. This illustrates why we aim to introduce DTW to our time series analysis model. Nevertheless, the current computation of DTW could not be directly applied to our model. How to integrate the DTW component into a deep learning framework is still a challenge. In what follows, we propose a DTW and attentions combined network structure, called *Local Dynamic Time Warping Attentions* (DTWAtt), to leverage features of DTW in our deep learning model.

4.2 Local Dynamic Time Warping Attentions

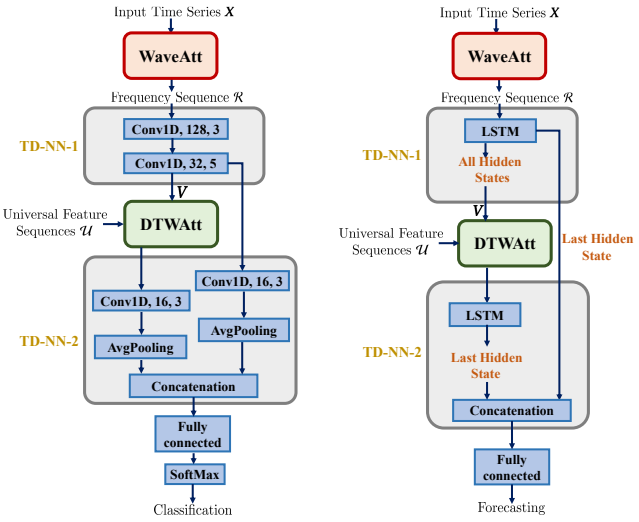
In the DTWAtt model, we first feed the output sequence of the WaveAtt model, *i.e.,* $\mathcal{R} = (\boldsymbol{R}_1, \ldots, \boldsymbol{R}_i, \ldots, \boldsymbol{R}_I)$, into a task-dependent neural network to generate a vector sequence $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_i, \ldots, \boldsymbol{v}_I)$, where $\boldsymbol{v}_i$ is generated by $\boldsymbol{R}_i$ and has K_V dimensions. The structure of the task-dependent neural network is given in Section 3.3. Next, we define a vector sequence $\boldsymbol{U}$, named as the *Universal Feature Sequence*, which is a learnable parameter sequence and has the same dimension as $\boldsymbol{V}$ (*i.e.,* K_V). The idea of DTWAtt is to calculate warped attentions of $\boldsymbol{V}$ to the universal feature sequence $\boldsymbol{U}$ using the DTW distance.

As shown in Fig. 4, to adapt to the dynamic nature of time series, we adopt another sliding window over the input sequence $\boldsymbol{V}$. Assume the window size is $L+1$, the sub-sequence of $\boldsymbol{V}$ in the window is $\boldsymbol{V}_i = (\boldsymbol{v}_i, \boldsymbol{v}_{i+1}, \ldots, \boldsymbol{v}_{i+L})$, which is the basic unit for the local time warping attention centered on time step i. We accordingly extract a vector sequence $\boldsymbol{U}_i$ from $\boldsymbol{U}$ with the same length $L+1$ as $\boldsymbol{U}_i = (\boldsymbol{u}_i, \boldsymbol{u}_{i+1}, \ldots, \boldsymbol{u}_{i+L})$. The distances of all possible warping paths between $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ are therefore calculated as

$$\boldsymbol{d}(\boldsymbol{V}_i, \boldsymbol{U}_i) = \left(d_1(\boldsymbol{V}_i, \boldsymbol{U}_i), \ldots, d_g(\boldsymbol{V}_i, \boldsymbol{U}_i), \ldots, d_G(\boldsymbol{V}_i, \boldsymbol{U}_i)\right). \tag{11}$$

Since we limit the length of $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ using a sliding window, the computation complexity for warping paths traversal is acceptable (the window size is set to 4 in the experiments).

Similar as the DTW algorithm, the DTWAtt network also focuses on finding the short warping path to measure the distance between



(a) Details of WHEN for time series classification.

(b) Details of WHEN for time series forecasting.

Figure 5: Model Framework for WHEN.

two sequences. To ensure the DTWAtt structure is compatible to the deep learning framework, we adopt an attention method to implement DTW. Specifically, we calculate the attention weight of each warping path using a SoftMax normalization as

$$\text{ATT}(d_g) = \frac{\exp(-d_g)}{\sum_{j=1}^{G} \exp(-d_j)}. \tag{12}$$

Obviously, the warping path with a shorter d_g has a larger ATT (d_g) since we multiply -1 with d_g in the SoftMax. Next, we use the attention weight ATT (d_g) to sum the distance in $\boldsymbol{d}(\boldsymbol{V}_i, \boldsymbol{U}_i)$ as

$$b = \sum_{g=1}^{G} \text{ATT}(d_g)\, d_g. \tag{13}$$

The output b is distance between $\boldsymbol{V}_i$ and $\boldsymbol{U}_i$, so is better denoted as b_i. In Eq. (13), the warping paths with shorter d_g would be paid more attention to, which is indeed similar to the DTW algorithm but implemented in a differentiable way.

In DTWAtt, we also adopt a multi-head attention mechanism to calculate the distance of $\boldsymbol{V}$ with multiple universal feature sequences. Given N parameter sequences $\mathcal{U} = (\boldsymbol{U}(1), \ldots, \boldsymbol{U}(N))$, the output of DTWAtt is a vector sequence $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_I)$, where the n-th element of $\boldsymbol{b}_i$ is the distance of $\boldsymbol{V}_i$ to $\boldsymbol{U}(n)_i$ measured by DTWAtt.

Intuitively, the DTWAtt is a neural network version of DTW that calculates the distance between the input sequence $\boldsymbol{V}$ with the parameter sequence $\boldsymbol{U}$ through an attention-like approach. DTWAtt could be considered as synchronizing any input sequence $\boldsymbol{V}$ with a universal feature sequence $\boldsymbol{U}$. The error caused by disturbances in sampling rate and phrase could be calibrated by synchronization mechanism. Moreover, DTWAtt is differentiable and thus can be plugged into a neural network and be trained by the BP algorithm.

4.3 The Model Output Layer

The WHEN model could be used for a wide variety of time series analysis tasks, such as Time Series Classification (TSC) and Time Series Forecasting (TSF). As shown in Fig. 2, in the WHEN model, the WaveAtt and DTWAtt modules were connected as a pipeline by two task-dependent neural networks (TD-NN). We design different TD-NN and output layers for the two tasks below.

For the TSC task, as illustrated in Fig. 5 (a), the TD-NN-1 contains two 1D convolutional layers. One consists of 128 convolution kernels in the size of 3 and the other consists of 32 convolution kernels in the size of 5. In the TD-NN-2, the outputs of WaveAtt and DTWAtt are respectively connected with a 1D convolutional layers consisting of 16 convolution kernels in the size of 3. We use the average pooling to convert outputs of the 1D convolutional layers as two vectors, and concatenate them as input of a fully connected layer with SoftMax classification outputs. For all the convolutional layers, we use ReLU as the active function and execute batch normalization for each layer.

For the TSF task, as illustrated in Fig. 5 (b), we use a LSTM network with 128 units as TD-NN-1, where the frequency sequence $\mathcal{R}$ generated by WaveAtt is converted as the input sequence $\boldsymbol{V}$ of DTWAtt. The TD-NN-2 uses the other LSTM network with 128 units to convert the output sequence DTWAtt as a vector and concatenates it with the last hidden state of LSTM in TD-NN-1. The concatenated vector is fed into a fully connected layer as the output. The active function of the LSTM networks is ReLU too.

We use a convolutional neural network as TD-NN of the classification task since the information for all parts of a time series is theoretically equally important. Oppositely, the information in the segment that is closer to "current" is more important for forecasting, so we adopt a unidirectional LSTM network as TD-NN of the forecasting tasks.

5 EXPERIMENTS

In this section, we evaluate the performance of WHEN over both time series classification and time series forecasting tasks.

5.1 Task I: Time Series Classification

In time series classification tasks, a multivariate time series $\boldsymbol{X} \in \mathbb{R}^{K \times I}$ is used as the model input to predict a classification label.

5.1.1 Datasets. We evaluate our model on the UEA multivariate time series classification archive [4], which is a large size multivariate time series dataset covering the areas of Human Activity Recognition, Motion classification, ECG classification, EEG/MEG classification, Audio Spectra classification, and among others. The UEA archive is designed to provide a standard archive for multivariate time series classification. In this way, we can compare the performance of our model with the baselines over a standard setup.

We use all 30 datasets of the UEA archive in our experiments. The sample sizes of the datasets are in the range of 27 - 50000, the series lengths are in 8 - 17984, and the dimensions are in 2 -1345.

5.1.2 Baselines. Eight baseline methods are adopted for the comparative experiments, including a DTW-based algorithm, a pattern-based algorithm, a feature-based algorithm, two ensemble methods, two deep learning models, and a wavelet-based deep learning model. All of them achieved the state-of-the-art performance in the recent literature.

• DTW_D [55]: A classifier based on 1-Nearest Neighbor with dimension-dependent DTW distance function. The 1-NN with DTW was ever the state-of-the-art algorithm of univariate time series classification [34]. For multivariate time series classification, the DTW-based algorithm also showed notable performance and was reported to achieve the best performance over UEA in Ref. [4].

• *WEASEL+MUSE* [52]: A bag-of-pattern multivariate time series classification algorithm, which achieved the state-of-the-art performance compared with the competitors of the same type on multivariate time series datasets [7]. We select it as the representative baseline of the pattern-based algorithms.

• *CMFMTS+RF* [6]: A classifier based on a set of complexity measures and descriptive features with random forest. We select it as a representative of ensemble methods for it achieved the state-of-the-art performance on the UEA datasets in Ref. [6].

• *LCEM* [21]: A hybrid ensemble method that combines boosting-bagging, divide-and-conquer and decision tree. It outperformed other random forest competitors such as RFM and XGBM on UEA datasets in Ref. [21]. We select it as anther representative of ensemble methods.

• *TapNet* [69]: A deep learning model for multivariate time series classification with attentional prototype network. The algorithm extracts low-dimensional features from multivariate time series with limited training samples. It achieved the best performance over the UEA datasets compared with other deep learning models in Ref. [69]. We select it as a representative of deep learning models.

• *TST* [68]: Time Series Transformer, a transformer-based framework for multivariate time series representation learning, which is a state-of-the-art deep learning model for many sequential data analysis tasks. TST is a representative transformer method in time series modeling.

• *MINIROCKET* [19]: A state-of-the-art model which extracts time series features with convolution kernels for better classification. With the help of ridge regression classifier, MINIROCKET performs well among several traditional methods in time series classification [19]. We select it as the representative baseline of the feature-based methods with ridge regression classifier.

• *mWDN* [65]: A deep learning model that employs a multilevel Wavelet decomposition neural network to extract frequency information form time series. This method shares similar idea with our WaveAtt module, but was designed just for univariate time series and can only extract features for several fixed frequency bands. We modify mWDN to fit our experiments by treating a multivariate series as multiple separate univariate series.

We also adopt two variants of WHEN as baselines, which are *WaveAtt* and *DTWAtt*. WaveAtt uses only the WaveAtt module of our model for classification while DTWAtt only uses the DTWAtt module. For baselines that have been evaluated over the UEA datasets in extant papers, we directly adopt the best scores from the papers for comparison.

5.1.3 Results and Analysis. We adopt *classification accuracy* as a metric to evaluate the model performance, which is defined as $accuracy = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\left(y_n = f(\boldsymbol{x_n})\right)$, where y_n is the class of the time series x_n, and $\mathbb{1}(\cdot)$ is an indicative function which equal to 1

Table 1: Performance comparison in the time series classification task.

Datasets	DTW_D	WEASEL +MUSE	TapNet	LCEM	CMFMTS +RF	mWDN	TST	MINI ROCKET	WaveAtt	DTWAtt	WHEN
ArticularyWordRecognition	0.987	**0.993**	**0.993**	**0.993**	0.990	0.990	0.987	0.990	**0.993**	0.983	**0.993**
AtrialFibrillation	0.200	0.333	0.333	**0.467**	0.200	0.400	0.400	0.400	**0.467**	0.400	**0.467**
BasicMotions	0.975	**1**	**1**	**1**	0.975	**1**	**1**	**1**	**1**	**1**	**1**
CharacterTrajectories	0.990	0.990	**0.997**	0.979	0.971	0.994	0.994	0.991	0.991	0.995	0.996
Cricket	**1**	0.986	0.986	0.986	0.972	0.972	0.986	0.986	0.986	**1**	**1**
DuckDuckGeese	0.600	0.575	0.600	0.375	0.520	0.600	0.660	0.640	0.600	0.680	**0.700**
EigenWorms	0.618	0.890	0.824	0.527	0.885	0.817	0.794	**0.962**	0.863	0.901	0.893
Epilepsy	0.964	0.993	0.964	0.986	**1**	0.949	0.971	**1**	0.993	0.993	0.993
ERing	0.133	0.133	0.133	0.200	0.930	0.930	0.948	**0.981**	0.933	0.893	0.959
EthanolConcentration	0.323	0.316	0.361	0.372	0.335	0.445	0.326	**0.475**	0.407	0.399	0.422
FaceDetection	0.529	0.545	0.556	0.614	0.548	0.615	**0.689**	0.620	0.610	0.635	0.658
FingerMovements	0.530	0.540	0.600	0.590	0.520	0.580	0.550	0.530	0.600	0.610	**0.660**
HandMovementDirection	0.231	0.378	0.378	**0.649**	0.284	0.568	0.595	0.378	0.419	0.351	0.554
Handwriting	**0.607**	0.531	0.388	0.287	0.282	0.305	0.359	0.511	0.431	0.566	0.561
Heartbeat	0.717	0.727	0.751	0.761	0.766	0.737	0.776	0.766	0.771	0.766	**0.780**
InsectWingbeat	0.115	0.128	0.222	0.228	0.640	0.656	**0.687**	0.633	0.640	0.650	0.657
JapaneseVowels	0.949	0.978	0.968	0.978	0.876	0.981	**0.997**	0.984	0.989	0.986	0.995
Libras	0.872	0.894	0.900	0.772	0.867	0.906	0.861	0.917	0.928	0.922	**0.933**
LSST	0.551	0.628	0.597	0.652	0.652	0.550	0.571	0.652	0.575	0.618	**0.663**
MotorImagery	0.500	0.500	0.590	0.600	0.510	0.540	0.540	0.550	0.600	0.620	**0.630**
NATOPS	0.883	0.883	0.939	0.916	0.817	0.950	0.933	0.928	0.967	0.972	**0.978**
PEMS-SF	0.711	0.705	0.751	0.942	**1**	0.890	0.896	0.827	0.884	0.902	0.925
PenDigits	0.977	0.969	0.980	0.977	0.951	**0.987**	0.981	0.979	0.963	0.982	**0.987**
Phoneme	0.151	0.190	0.175	0.288	0.287	0.178	0.180	0.289	0.278	0.271	**0.293**
RacketSports	0.803	0.914	0.842	**0.941**	0.809	0.868	0.849	0.875	0.908	0.921	0.934
SelfRegulationSCP1	0.775	0.744	0.863	0.839	0.812	0.891	**0.922**	0.904	0.891	0.846	0.908
SelfRegulationSCP2	0.539	0.522	0.550	0.550	0.417	0.561	**0.604**	0.506	0.583	0.578	0.589
SpokenArabicDigits	0.963	0.982	0.983	0.973	0.969	0.995	**0.998**	0.991	0.992	0.995	0.997
StandWalkJump	0.200	0.333	0.400	0.400	0.333	0.333	0.400	0.400	0.467	**0.533**	**0.533**
UWaveGestureLibrary	0.903	0.903	0.903	0.897	0.772	0.891	0.913	**0.925**	0.816	0.875	0.919
Avg. Rank	9.1	7.7	6.7	6.3	8.3	6.3	5.1	5	5	4.3	2.1
Wins/Ties	2	2	3	5	2	2	7	6	3	3	14

when $y_n = f(\boldsymbol{x}_n)$ and equal to 0 when $y_n \neq f(\boldsymbol{x}_n)$. Table 1 shows the experimental results, with the summarized information listed in the bottom two lines. Note that the best performance for each dataset is highlighted in bold.

First, it is clear that among all the competitors, WHEN achieves the best performance in terms of both the largest number of wins (the best in 14 out of 30 datasets) and the highest average rank (2.1). The rank index indicates even for the cases where our model is not the best, its performances are still very competitive.

Second, the two variants of WHEN also have competitive performances. They are the second and the third best in terms of the average rank. The results verify the advantage of the proposed WaveAtt and DTWAtt structures, and as an ablation study, explain why the two components are essential to the success of WHEN. It seems that DTWAtt is more effective than WaveAtt. It is reasonable because DTWAtt is designed to overcome the inter-sequence asynchronism problem, which is more important than the intra-sequence nonstationarity problem for the classification task.

Finally, the deep learning methods including WHEN, mWDN, TST and TapNet have overall better performances. The reason might be the representational learning ability of deep learning is very suitable for feature extracting from high-dimensional large-scale multivariate datasets.

We also give a Nemenyi test for the experiment results in Fig. 6. On the Nemenyi critical difference diagrams, two methods have statistically significant difference if the difference between their average ranks is larger than critical difference (the line segment with "CD" on the top left corner of the diagrams). Models that are connected by a bold line do not have statistically significant difference. The performance of WHEN is significantly better than the no-WHEN baselines with a 5% significance level.

5.2 Task II: Time Series Forecasting

In the forecasting task, we use the fragments of time series in a sliding window $t - N$ to t as inputs of our model to forecast the value of the time series from $t + 1$ to future several time steps.

5.2.1 Datasets. In the experiments, we compare our model with the baselines over three real-world multivariate time series datasets. All of the three datasets are publicly available.

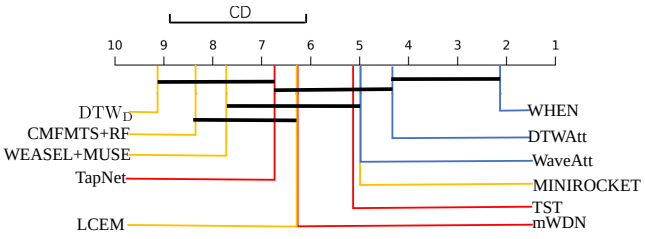


Figure 6: Critical difference diagram of the TSC experiments. Blue lines correspond to WHEN and its variants. Red lines correspond to deep learning baselines. Orange lines correspond to traditional baselines.

• *Temperature*: This dataset contains time series of monthly average temperature of all 50 US states from Jan. 2000 to Aug. 2013[1]. It is a 50 dimensional time series with a length of 164.

• *AQI (Air Quality Indexes)*: This dataset contains time series of daily air quality indexes collected from 1 Jan. 2014 to 1 Dec. 2021 in the Shanghai city of China[2]. The dataset has 6 features including PM2.5, PM10, O3, NO2, SO2, CO. It is a 6 dimensional time series with a length of 2815.

• *Traffic*: This dataset contains the time series of traffic speeds of 214 road segments in the Guangzhou city of China in Aug. 2016 [13]. The traffic speeds are collected every 10 minutes. It is a 214 dimensional time series with a length of 4464.

In the experiments of AQI and Traffic, we set the sliding window size to 100, and use our model to forecast the next 1, 5, 10 steps. Since the sampling time period in Temperature dataset is much longer, we use the data of the last 50 months to predict the future 1, 3 and 5 months. We set the first 8:1:1 of the datasets as training, validation, and test sets, respectively.

5.2.2 Baselines. We consider the following eight baseline methods:

• *ARIMA* [27]: ARIMA is a classic time series analysis model that combines an autoregressive (AR) and a moving average (MA) processes for forecasting. ARIMA contains an initial differencing in the "integrated" part to eliminate the nonstationarity in mean of time series. In ARIMA, we treat the multivariate data as multiple univariate time series for forecasting.

• *FC-LSTM* [58]: FC-LSTM is a basic sequence to sequence model that forecasts time series with a LSTM layer and fully connected layer.

• *NRDE* [44]: We select NRDE as a representative neural differential equation based method, which applies rough path theory to improve neural controlled differential equations [35] in long time series.

• *STRIPE++* [38]: To produce plausible and diverse time series predictions for nonstationary time series, STRIPE++ also applies DTW algorithm [36] for forecasting. We select STRIPE++ as a representative method for DTW related methods [16, 36–39].

• *ESG* [66]: Recently, Graph neural network based methods becomes popular in time series forecasting [11, 14, 53, 54, 66], where the graph network is used to model the relationship among graph nodes. We select ESG as a representative method that considers the multi-scale interactions of time series.

• *GTS* [53]: GTS is another representative graph method that forecasting time series without given graph.

• *TST* [68]: We select TST as a representative transformer based time series forecasting method.

• *mWDN* [65]: mWDN adopts a multilevel wavelet decomposition network to extract frequency components from time series, and feeds the frequency components into a group of LSTM networks for forecasting. We select mWDN as a baseline as it is the state-of-the-art method that has a similar idea with the WaveAtt component of our model, *i.e.,* using wavelet method to extract frequency component from time series.

We also adopt the two variants of WHEN, *i.e.,* WaveAtt and DTWAtt, as baselines.

5.2.3 Results and Analysis. We use the Root Mean Square Error (RMSE) as a metric to evaluate the model performance in time series forecasting, which is defined as: $\text{RMSE} = \sqrt{\frac{1}{T}\frac{1}{K}\sum_{t=1}^{T}\|\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t\|^2}$,

where $\|\cdot\|$ is the l-2 norm, $\boldsymbol{y}_t$ is the actual value at step t in test set, $\hat{\boldsymbol{y}}_t$ is the predicted value and K is the dimension of $\boldsymbol{y}_t$.

Table 2 presents the results of all the methods, where P1, P3, P5, P10 denote the performance of forecasting at the next 1, 3, 5, 10 steps, respectively. First, as shown in the table, the WHEN model achieves the best performance compared with the baselines, indicating the effectiveness of our model. Second, as an ablation study of our model, we observe that the WaveAtt and DTWAtt modules both contribute to the performance of WHEN. Specifically, WaveAtt is more effective than DTWAtt, which indicates the importance of frequency analysis in time series forecasting. It is also supported by the fact that the models using frequency analysis of wavelet functions, *i.e.,* WHEN, WaveAtt and mWDN, have better performances than other baselines. Third, all deep learning methods work better than the ARIMA method, since ARIMA can only handle the nonstationarity in the sense of mean, but the nonstationarity in real-world time series is indeed much more complex.

5.3 Exploratory Analysis

Here, we illustrate how WaveAtt and DTWAtt work in the model via exploratory analysis.

5.3.1 Data-Dependent Frequencies Extraction. The data- dependent frequency analysis is an important feature of our model. In the WaveAtt module, the frequency band $1/\alpha$ in the data-dependent wavelet function (Eq. (5)) is controlled by the input sequence ς_i. We demonstrate this mechanism in Fig. 7. Here, Fig. 7(c) shows the input time series ς of the WaveAtt module, which is selected from the Heartbeat dataset of UEA archive. Fig. 7(a) and Fig. 7(b) visualize the data-dependent frequency band sequences, *i.e.,* $1/\alpha(\varsigma_i)$, of the data-dependent wavelet function based on Complex Gaussian 1 and Daubechies 2 mother functions [43].

As the figure shows, there are three low frequency ranges with high amplitude: around time step 10, 80 and 175, which are shown in the dashed line blocks. Accordingly, the frequency curves in Fig. 7(a) and Fig. 7(b) also reach their bottoms at close time steps. The observation indicates that WaveAtt can dynamically change its frequency bands with input data and extract suitable frequency components of the input time series. Moreover, the frequency components extracted by different Wavelet families are different. In

[1] https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data
[2] https://aqicn.org/data-platform/

Table 2: Performance comparison in the time series forecasting task.

Datasets	Forecasting time period	ARIMA	FC-LSTM	NRDE	STRIPE++	ESG	TST	GTS	mWDN	WaveAtt	DTWAtt	WHEN
Temperature	P1	0.338	0.298	0.276	0.279	0.245	0.265	0.256	0.247	0.254	0.238	**0.236**
	P3	0.491	0.329	0.278	0.287	0.263	0.269	0.264	0.264	0.258	0.263	**0.249**
	P5	0.555	0.334	0.287	0.300	**0.265**	0.286	0.275	0.271	0.267	0.273	0.268
AQI	P1	0.092	0.087	0.080	0.084	0.083	0.082	0.083	0.081	0.080	0.080	**0.078**
	P5	0.104	0.092	0.088	0.091	0.087	0.088	0.091	0.089	0.085	0.090	**0.084**
	P10	0.121	0.097	0.094	0.095	0.092	0.092	0.094	0.095	0.092	0.091	**0.090**
Traffic	P1	0.114	0.120	0.118	0.108	**0.102**	0.108	0.105	0.107	0.105	0.107	0.105
	P5	0.127	0.129	0.120	0.122	0.113	0.113	0.114	0.114	0.114	0.119	**0.110**
	P10	0.152	0.140	0.124	0.125	0.119	0.127	0.129	0.118	0.115	0.126	**0.112**
Avg. Rank		10.7	10.2	6.8	8.4	3.3	5.9	6.3	5.2	3.1	4.7	1.4
Wins/Ties		0	0	0	0	2	0	0	0	0	0	7

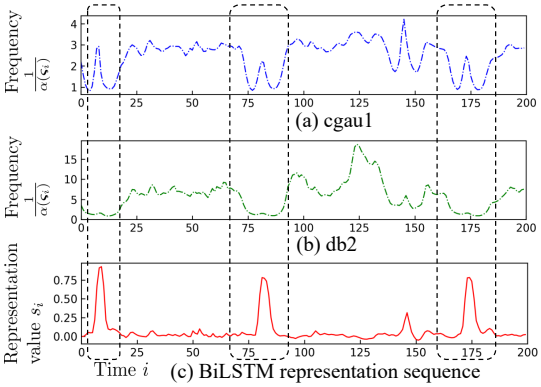


Figure 7: Data-dependent frequencies extracted by different wavelet families. (a) and (b): The frequencies extracted by Complex Gaussian 1 wavelet and Daubechies 2 wavelet. (c): The BiLSTM representation sequence S of a sample.

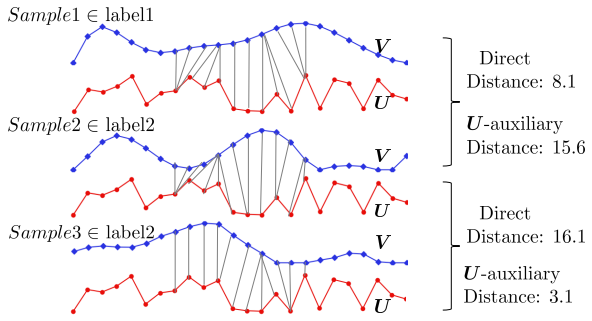


Figure 8: Direct and warping distance comparison among different samples.

general, the frequency bands in Fig. 7 (b) are higher than that in Fig. 7(a), indicating that the multiple wavelet families mechanism in WaveAtt indeed provides frequency components diversity to our model. For real-life time series containing multiple frequency components, our model can use different wavelet families to cover their frequency bands.

5.3.2 Warping Distance Comparison in DTW Attention. In Fig. 8, we demonstrate the nature of DTWAtt using an example of warping distance comparison in the DTW attention. As the figure shows, there are three series samples selected from the Heartbeat dataset. Here, *Sample*1 is with the label1 while *Sample*2 and *Sample*3 are with the same label2. If we directly compare the distance among the three samples, the distance of *Sample*2 to *Sample*1 is smaller than that of *Sample*2 to *Sample*3 (*i.e.,* 8.1<16.1). The phase drift between *Sample*2 and *Sample*3 causes a large distance between the samples with the same label.

In DTWAtt, this problem is solved through introducing an auxiliary universal feature sequence $\boldsymbol{U}$. As the figure shows, instead of comparing the distance between samples directly, DTWAtt calculates the warping distances of the samples with the sequence $\boldsymbol{U}$. We can see after adjusting the phase drift of sequences by the DTW attention, the distance between *Sample*2 and *Sample*3 (in fact, it is the distance of *Sample*2 $\rightarrow \boldsymbol{U} \rightarrow$ *Sample*3) is much smaller than that between *Sample*2 and *Sample*1 (*i.e.,* 3.1≪15.6). This indicates DTWAtt is very useful to handle the asynchronism in multiple time series due to irregular sampling rate and the phase drift.

6 CONCLUSIONS

In this paper, we aim to build a deep learning model for nonstationary and asynchronous time series analysis. We designed two types of attentions combined with task-dependent neural networks constitute the novel WHEN model for both time series classification and forecasting tasks. Extensive experiments on abundant real-world datasets demonstrated the superiority of our model to numerous competitors.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2021ZD0111200). Prof. Wang's work was supported by the National Natural Science Foundation of China (No. 72222022, 72171013). Prof. Wu's work was supported by the National Natural Science Foundation of China (No. 72031001, 72242101).

REFERENCES

[1] Ilan Alon, Min Qi, and Robert J Sadowski. 2001. Forecasting aggregate retail sales: a comparison of artificial neural networks and traditional methods. *JRCS* 8, 3 (2001), 147–156.

[2] Xueli An, Dongxiang Jiang, Chao Liu, and Minghao Zhao. 2011. Wind farm power prediction based on wavelet decomposition and chaotic time series. *Expert Syst. Appl.* 38, 9 (2011), 11280–11285.

[3] Abdourrahmane M Atto and Yannick Berthoumieu. 2011. Wavelet packets of nonstationary random processes: Contributing factors for stationarity and decorrelation. *IEEE TIT* 58, 1 (2011), 317–330.

[4] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. (2018). arXiv:1811.00075

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

[6] Francisco J Baldán and José M Benítez. 2020. Multivariable times series classification through an interpretable representation. (2020). arXiv:2009.03614

[7] Mustafa Gokce Baydogan. 2017. *Multivariate time series classification datasets*. Retrieved 12-1, 2020 from http://www.mustafabaydogan.com

[8] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. 2013. A bag-of-features framework to classify time series. *TPAMI* 35, 11 (2013), 2796–2802.

[9] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. 2016. Should we really use post-hoc tests based on mean-ranks? *JMLR* 17, 1 (2016), 152–161.

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

[11] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *NeurIPS'20* 33 (2020), 17766–17778.

[12] Jatin Chauhan, Aravindan Raghuveer, Rishi Saket, Jay Nandy, and Balaraman Ravindran. 2022. Multi-Variate Time Series Forecasting on Variable Subsets. In *KDD'22*. 76–86.

[13] Xinyu Chen, Zhaocheng He, and Jiawei Wang. 2018. Spatial-temporal traffic speed patterns discovery and incomplete data recovery via SVD-combined tensor decomposition. *TR_C* 86 (2018), 59–77.

[14] Yuzhou Chen, Ignacio Segovia, and Yulia R Gel. 2021. Z-GCNETs: Time Zigzags at Graph Convolutional Networks for Time Series Forecasting. In *International Conference on Machine Learning*. PMLR, 1684–1694.

[15] Ranak Roy Chowdhury, Xiyuan Zhang, Jingbo Shang, Rajesh K Gupta, and Dezhi Hong. 2022. TARNet: Task-Aware Reconstruction for Time-Series Transformer. In *KDD'22*. 14–18.

[16] Marco Cuturi and Mathieu Blondel. 2017. Soft-dtw: a differentiable loss function for time-series. In *ICML'17*. PMLR, 894–903.

[17] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Annh Ratanamahatana, and Eamonn Keogh. 2019. The UCR time series archive. *IEEE-CAA JAS* 6, 6 (2019), 1293–1305.

[18] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. 2020. Normalizing kalman filters for multivariate time series analysis. *NeurIPS'20* 33 (2020), 2995–3007.

[19] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *KDD'21*. 248–257.

[20] David A Dickey and Sastry G Pantula. 1987. Determining the order of differencing in autoregressive processes. *JBES* 5, 4 (1987), 455–461.

[21] Kevin Fauvel, Élisa Fromont, Véronique Masson, Philippe Faverdin, and Alexandre Termier. 2020. Local Cascade Ensemble for Multivariate Data Classification. (2020). arXiv:2005.03645

[22] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *DMKD* 33, 4 (2019), 917–963.

[23] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. In *NeurIPS'19*. 4650–4661.

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

[25] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM networks. In *IJCNN'05*, Vol. 4. 2047–2052.

[26] Christopher E Heil and David F Walnut. 1989. Continuous and discrete wavelet transforms. *SIREV* 31, 4 (1989), 628–666.

[27] SL Ho and Min Xie. 1998. The use of ARIMA models for reliability forecasting and analysis. *Comput. Ind. Eng.* 35, 1-2 (1998), 213–216.

[28] Jiahao Ji, Jingyuan Wang, Chao Huang, Junjie Wu, Boren Xu, Zhenhe Wu, Junbo Zhang, and Yu Zheng. 2023. Spatio-Temporal Self-Supervised Learning for Traffic Flow Prediction. In *AAAI'23*.

[29] Jiahao Ji, Jingyuan Wang, Zhe Jiang, Jiawei Jiang, and Hu Zhang. 2022. STDEN: Towards physics-guided neural networks for traffic flow prediction. In *AAAI'22*, Vol. 36. 4048–4056.

[30] Jiahao Ji, Jingyuan Wang, Zhe Jiang, Jingtian Ma, and Hu Zhang. 2020. Interpretable spatiotemporal deep learning model for traffic flow prediction based on potential energy fields. In *ICDM'20*. IEEE, 1076–1081.

[31] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. 2023. PDFormer: Propagation Delay-aware Dynamic Long-range Transformer for Traffic Flow Prediction. In *AAAI'23*.

[32] Tae Woo Joo and Seoung Bum Kim. 2015. Time series forecasting based on wavelet filtering. *Expert Syst. Appl.* 42, 8 (2015), 3868–3874.

[33] Jan W Kantelhardt, Stephan A Zschiegner, Eva Koscielny-Bunde, Shlomo Havlin, Armin Bunde, and H Eugene Stanley. 2002. Multifractal detrended fluctuation analysis of nonstationary time series. *Physica A* 316, 1-4 (2002), 87–114.

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

[35] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. 2020. Neural controlled differential equations for irregular time series. *NeurIPS'20* 33 (2020), 6696–6707.

[36] Vincent Le Guen and Nicolas Thome. 2019. Shape and Time Distortion Loss for Training Deep Time Series Forecasting Models. In *NeurIPS'19*, Vol. 4191.

[37] Vincent Le Guen and Nicolas Thome. 2020. Probabilistic Time Series Forecasting with Structured Shape and Temporal Diversity. *NeurIPS'20* 33 (2020).

[38] Vincent Le Guen and Nicolas Thome. 2022. Deep Time Series Forecasting with Shape and Temporal Criteria. *TPAMI* (2022).

[39] Dongha Lee, Seonghyeon Lee, and Hwanjo Yu. 2021. Learnable dynamic temporal pooling for time series classification. In *AAAI'21*, Vol. 35. 8288–8296.

[40] Jessica Lin, Rohan Khade, and Yuan Li. 2012. Rotation-invariant similarity in time series using bag-of-patterns representation. *JIIS* 39, 2 (2012), 287–315.

[41] Jason Lines and Anthony Bagnall. 2015. Time series classification with ensembles of elastic distance measures. *DMKD* 29, 3 (2015), 565–592.

[42] Yu Ma, Zhining Liu, Chenyi Zhuang, Yize Tan, Yi Dong, Wenliang Zhong, and Jinjie Gu. 2022. Non-stationary Time-aware Kernelized Attention for Temporal Event Prediction. In *KDD'22*. 1224–1232.

[43] Michel Misiti, Yves Misiti, Georges Oppenheim, and Jean-Michel Poggi. 2007. *Wavelets and their Applications*. Vol. 330. Wiley Online Library.

[44] James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. 2021. Neural rough differential equations for long time series. In *ICML'21*. PMLR, 7829–7838.

[45] Abdullah Mueen, Eamonn Keogh, and Neal Young. 2011. Logical-shapelets: an expressive primitive for time series classification. In *KDD'11*. 1154–1162.

[46] Eduardo Ogasawara, Leonardo C Martinez, Daniel De Oliveira, Geraldo Zimbrão, Gisele L Pappa, and Marta Mattoso. 2010. Adaptive normalization: a novel data normalization approach for non-stationary time series. In *IJCNN'10*. 1–8.

[47] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs M Bergmann, and Roland Vollgraf. 2021. Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. *ICLR'21* (2021).

[48] Houxing Ren, Jingyuan Wang, and Wayne Xin Zhao. 2022. Generative Adversarial Networks Enhanced Pre-training for Insufficient Electronic Health Records Modeling. In *KDD'22*. 3810–3818.

[49] Houxing Ren, Jingyuan Wang, Wayne Xin Zhao, and Ning Wu. 2021. Rapt: Pre-training of time-aware transformer for learning robust healthcare representation. In *KDD'21*. 3503–3511.

[50] Alaa Sagheer and Mostafa Kotb. 2019. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing* 323 (2019), 203–213.

[51] Patrick Schäfer. 2015. The BOSS is concerned with time series classification in the presence of noise. *DMKD* 29, 6 (2015), 1505–1530.

[52] Patrick Schäfer and Ulf Leser. 2017. Multivariate time series classification with WEASEL+ MUSE. (2017). arXiv:1711.11343

[53] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete graph structure learning for forecasting multiple time series. In *ICLR'21*.

[54] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training Enhanced Spatial-temporal Graph Neural Network for Multivariate Time Series Forecasting. In *KDD'22*. 1567–1577.

[55] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. 2015. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *SDM'15*. 289–297.

[56] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2018. A comparison of ARIMA and LSTM in forecasting time series. In *ICMLA'18*. 1394–1401.

[57] Skander Soltani. 2002. On the use of the wavelet decomposition for time series prediction. *Neurocomputing* 48, 1-4 (2002), 267–277.

[58] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *NeurIPS'14* 27 (2014).

[59] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I Webb. 2022. MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *DMKD* 36, 5 (2022), 1623–1646.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS'17*.

[61] Haizhong Wang, Lu Liu, Shangjia Dong, Zhen Qian, and Heng Wei. 2016. A novel work zone short-term vehicle-type specific traffic speed prediction model through the hybrid EMD–ARIMA framework. *Transportmetrica B-Transp. Dyn.* 4, 3 (2016), 159–186.

[62] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic speed prediction and congestion source exploration: A deep learning method. In *ICDM'16*. IEEE, 499–508.

[63] Jingyuan Wang, Jiahao Ji, Zhe Jiang, and Leilei Sun. 2022. Traffic Flow Prediction Based on Spatiotemporal Potential Energy Fields. *TKDE* (2022).

[64] Jingyuan Wang, Xiaoda Wang, Chao Li, Junjie Wu, et al. 2020. Deep Fuzzy Cognitive Maps for Interpretable Multivariate Time Series Prediction. *TFS* (2020), 1–14.

[65] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In *KDD'18*. 2437–2446.

[66] Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. 2022. Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting. In *KDD'22*. 2296–2306.

[67] In-Kwon Yeo and Richard A Johnson. 2000. A new family of power transformations to improve normality or symmetry. *Biometrika* 87, 4 (2000), 954–959.

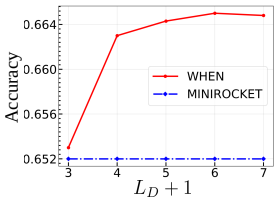
[68] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *KDD'21*. 2114–2124.

[69] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. 2020. TapNet: Multivariate Time Series Classification with Attentional Prototypical Network.. In *AAAI'20*. 6845–6852.

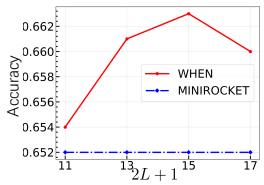
[70] Fan Zhou, Liang Li, Kunpeng Zhang, Goce Trajcevski, Fuming Yao, Ying Huang, Ting Zhong, Jiahao Wang, and Qiao Liu. 2020. Forecasting the Evolution of Hydropower Generation. In *KDD'20*. 2861–2870.

Table 3: Pairwise statistical significance and comparison of the TSC task with Wilcoxon signed rank test.

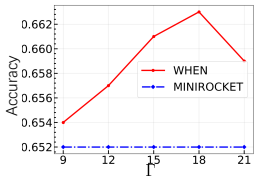
p-value	WHEN	DTWAtt	WaveAtt	TST	MINI ROCKET	mWDN	CMFMTS +RF	LCEM	TapNet	WEASEL +MUSE	DTW_D
WHEN	-	0.000	0.000	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000
DTWAtt	0.000	-	0.148	0.414	0.145	0.024	0.000	0.040	0.000	0.000	0.000
WaveAtt	0.000	0.148	-	0.895	0.473	0.046	0.001	0.067	0.000	0.003	0.000
MINIROCKET	0.001	0.414	0.895	-	0.713	0.086	0.000	0.137	0.003	0.000	0.000
TST	0.001	0.145	0.473	0.713	-	0.062	0.004	0.313	0.073	0.021	0.000
mWDN	0.000	0.024	0.046	0.086	0.062	-	0.024	0.804	0.269	0.060	0.000
CMFMTS+RF	0.000	0.000	0.001	0.000	0.004	0.024	-	0.144	0.116	0.284	0.274
LCEM	0.000	0.040	0.067	0.137	0.313	0.804	0.144	-	0.467	0.069	0.011
TapNet	0.000	0.000	0.000	0.003	0.073	0.269	0.116	0.467	-	0.181	0.000
WEASEL+MUSE	0.000	0.000	0.003	0.000	0.021	0.060	0.284	0.069	0.181	-	0.018
DTW_D	0.000	0.000	0.000	0.000	0.000	0.000	0.274	0.011	0.000	0.018	-



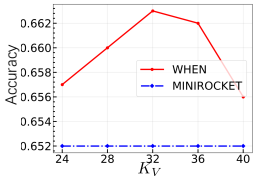
(a) The DTWAtt sliding window length $L_D + 1$.



(b) The sliding window size $2L + 1$.



(c) Total Wavelet families Γ.



(d) Dimension number K_V of $\boldsymbol{V}$.

Figure 9: Parameter sensitivity on LSST dataset.

A SUPPLEMENTAL MATERIALS

A.1 Derivatives Calculating

In DTWAtt, we implement the DTW algorithm as a neural attention network form. As we mentioned in the main body, the DTW attention is derivable and therefore could be trained in the BP algorithm. DTWAtt has the input $\boldsymbol{V}$, output $\boldsymbol{B}$ and learnable parameter $\boldsymbol{U}$. Here, we illustrate that $\boldsymbol{B}$ is differentiable with respect to $\boldsymbol{V}$ and $\boldsymbol{U}$. This is equal to the statement that b_i is differentiable with respect to each component $v_{i+t}^{(k)}$ and $u_{i+t}^{(k)}$ in $\boldsymbol{V}_i$ and $\boldsymbol{U}_i$.

We first give the derivative of $v_{i+t}^{(k)}$ as

$$\begin{aligned}\frac{\partial b_i}{\partial v_{i+t}^{(k)}} &= \sum_{g=1}^{G} \frac{\partial \mathrm{ATT}\left(d_g\right)}{\partial v_{i+t}^{(k)}} d_g + \mathrm{ATT}\left(d_g\right) \frac{\partial d_g}{\partial v_{i+t}^{(k)}} \\ &= \sum_{g=1}^{G} \frac{\partial d_g}{\partial v_{i+t}^{(k)}} \frac{\partial \mathrm{ATT}\left(d_g\right)}{\partial d_g} d_g + \mathrm{ATT}\left(d_g\right) \frac{\partial d_g}{\partial v_{i+t}^{(k)}},\end{aligned} \tag{14}$$

where $\partial \mathrm{ATT}\left(d_g\right)/\partial d_g$ can be calculated, thus $\partial b_i/\partial v_{i+t}^{(k)}$ exists if $\partial d_g\left(\boldsymbol{V}_i, \boldsymbol{U}_i\right)/\partial v_{i+t}^{(k)}$ exists.

In Eq. (14), d_g can be denoted as the sum of the l-2 norms, *i.e.,*

$$d_g\left(\boldsymbol{V}_i, \boldsymbol{U}_i\right) = \sum_{z=1}^{Z} \left\| \boldsymbol{v}_{i+p_z} - \boldsymbol{u}_{i+q_z} \right\|, \tag{15}$$

where the choices of vector pairs $\{\boldsymbol{v}_{p_z}, \boldsymbol{u}_{q_z}\}, z \in \mathbb{Z}^+$ are determined by the details of g-th warping path. Concretely, we can denote d_g in another form which considers the vector pairs by a constant δ as

$$d_g\left(\boldsymbol{V}_i, \boldsymbol{U}_i\right) = \sum_{p,q=0}^{L} \delta_{p,q} \left\| \boldsymbol{v}_{i+p} - \boldsymbol{u}_{i+q} \right\|, \tag{16}$$

where $\delta_{p,q}$ is 0 or 1, based on the form of g-th path. $\delta_{p,q}$ equals 1 if and only if the vector pair $\{\boldsymbol{v}_{i+p}, \boldsymbol{u}_{i+q}\}$ is contained in the warping path. Since l-2 norm is differentiable and the derivative of constant $\delta_{p,q}$ is always 0, $\partial d_g\left(\boldsymbol{V}_i, \boldsymbol{U}_i\right)/\partial v_{i+t}^{(k)}$ exists for each g, i, k and t. Thus $\partial b_i/\partial v_{i+t}^{(k)}$ exists. The existence for derivative $\partial b_i/\partial u_{i+t}^{(k)}$ can be calculated in the same method.

A.2 Parameter Sensitivity

In our model, there are several parameters to tune. Fig. 9 reports the sensitivity of the classification accuracy to these parameters on the LSST dataset. We also incorporate the best no-WHEN baseline MINIROCKET on LSST dataset, which is the best method in the baselines, for comparison.

Here, we vary the number of four key parameters, which are the number of DTWAtt sliding window length $L_D + 1$ in the set {3, 4, 5, 6, 7}, the number of WaveAtt sliding window size $2L + 1$ in the set {11, 13, 15, 17}, the number of used wavelet families Γ in the set {9, 12, 15, 18, 21} and the dimension number of DTWAtt $\boldsymbol{V}$, *i.e.,* K_V, in the set {24, 28, 32, 36, 40}. As shown in the figures, our model is relatively stable and consistently competitive to the MINIROCKET baseline when we vary the four parameters. The sensitivities of other parameters are similar for the four parameters.

For TSF task, Fig. 10 reports the sensitivity of the RMSE to these parameters on the AQI dataset when forecasting the next 5 steps. We also incorporate the best no-WHEN baseline ESG, which is the best method in the baselines, for comparison. The key parameter

Table 4: Notations, explanations, and configurations in our experiments.

Part	Notation	Explanation	Configuration
WaveAtt	$\boldsymbol{X} \in \mathbb{R}^{K_X \times I}$	Time series input.	K_X, I depend on datasets.
	$\boldsymbol{S} \in \mathbb{R}^{K_S \times I}$	BiLSTM representation sequence.	$K_S = \min(192, 2 \times K_X)$
	$\boldsymbol{\mathcal{S}_i} \in \mathbb{R}^{K_S \times (2L+1)}$	BiLSTM representation sub-sequence in sliding window.	$2L + 1 = 15$
	$\boldsymbol{w} \in \mathbb{R}^{2L+2}$	The parameters to calculate α in WaveAtt.	See above
	$\boldsymbol{\mathcal{W}} \in \mathbb{R}^{(2L+2) \times K_S \times \Gamma}$	The parameters in different dimensions and Wavelet families.	$\Gamma = 18$
	$\boldsymbol{\mathcal{R}} \in \mathbb{R}^{K_S \times \Gamma \times I}$	Frequency components sequence.	See above
DTWAtt	$\boldsymbol{V} \in \mathbb{R}^{K_V \times I}$	Vector sequence for time series classification & forecasting.	$K_V = 32$ or 128
	$\boldsymbol{V_i} \in \mathbb{R}^{K_V \times (L_D+1)}$	Vector sequence in sliding window.	$L_D + 1 = 4$
	$\boldsymbol{\mathcal{U}} \in \mathbb{R}^{N \times K_V \times I}$	The multi-head parameter sequence in DTWAtt.	$N = 5$
	$\boldsymbol{U_i} \in \mathbb{R}^{K_V \times (L_D+1)}$	Parameter sequence for one head in sliding window.	See above
	$\boldsymbol{d}(\boldsymbol{V_i}, \boldsymbol{U_i}) \in \mathbb{R}^G$	Distance vector of all warping paths.	$G = 9$
	$\boldsymbol{B} \in \mathbb{R}^{N \times I}$	Attentive distance sequence.	See above

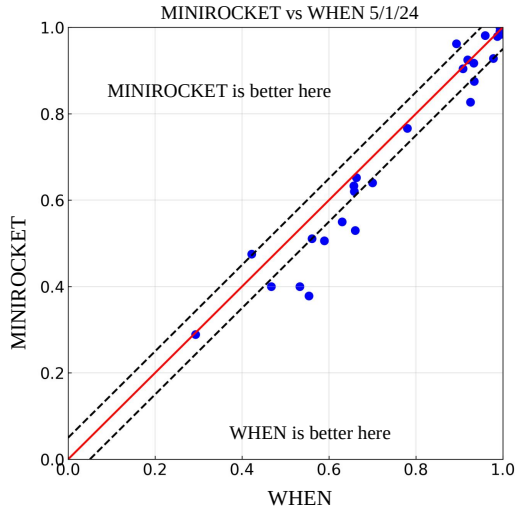
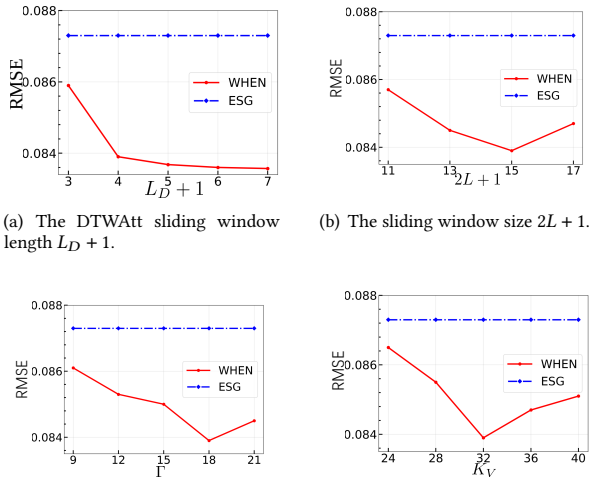


Figure 11: Pairwise plots with win/tie/loss counts between WHEN and MINIROCKET.



(a) The DTWAtt sliding window length $L_D + 1$.

(b) The sliding window size $2L + 1$.

(c) Total Wavelet families Γ.

(d) Dimension number K_V of $\boldsymbol{V}$.

Figure 10: Parameter sensitivity on AQI dataset.

settings are similar to the experiments in TSC task. As shown in the figures, our model is relatively stable and consistently competitive to the ESG baseline when we vary the four parameters.

A.3 Additional Comparison Results

We also conduct a two-sided Wilcoxon signed rank test [9] for the TSC task as shown in Table 3. The p-value, indicating the statistical significance of the test, is reported in the table. Our results show that WHEN has a significant difference compared to the baselines, indicating its effectiveness in TSC task.

To enhance the demonstration of WHEN's performance, we present pairwise plot in Fig. 11, illustrating the win/tie/loss counts between WHEN and MINIROCKET. Each point on the plot represents the accuracy value of the two methods on a specific dataset. The dotted lines indicate ±5% intervals around the classification accuracy. Our results indicate that WHEN outperforms MINIROCKET on several datasets, while MINIROCKET performs better on some others. The pairwise plot provides valuable insights into the strengths and weaknesses of both methods.

A.4 Experiment Configuration

Table 4 lists the notations of the hyper-parameter of our model. We organize the notations in two groups, namely WaveAtt and DTWAtt. The last column presents the parameter configurations which leads to the reported experimental results in our paper.

Our software environment contains ubuntu 16.04, Pytorch v1.6.0 and python 3.6.12. All of the experiments are conducted on a machine with four GPUs (*NVIDIA GeForce GTX 1080 Ti * 4*), one CPU (*Inter i7 6700k*) and 32G memory.