Application of Bispectrum and Convolutional Neural Network on Time Series Data

Shanglin Zhou, 2334229

April 30, 2018

Abstract

In this experimental study, we address the problem of using Deep Neural Network (DNN) for time series data classification problem. Time series data classification is widely used in many different domains, such as disease detection based on electrocardiogram, emotion analysis using EEG data, earth quake prediction and alert, etc, thus this problem is very critical and attracts a lot of research efforts. In this report, we study one interesting feature of time series data, namely, high-order spectrum of time series. In particular, we study the bi-spectrum behavior of time series data, and further apply a deep learning model on these bi-spectrum features. The goal is to train the deep learning model using bi-spectrum to achieve a high classification accuracy. Experimental results shows that this is a promising method and proved it can achieve the state-of-the-art classification accuracy levels on several real datasets.

1 Introduction

One interesting topic of time series study is the classification of time series data. Example is like when we have a bench of time series datasets collected from the earthquake censor, and we want to identify what kind of features in these dataset indicate the occurrence of earthquake. Then we need to classify which series is recorded just before earthquake and which is recorded during normal time. Other applications in the medical discipline could be heart attack prediction based on the medical electrocardiogram. In the domain of financial analysis, the high frequency stock market data is also deeply mined to find the hidden pattern, thus predict the price trends in the near future.

Due to the wide range of applications, time series classification problem has attracted a huge amount of researchers all around the world. Traditional methods include pattern analysis, frequency analysis, etc. In recent years, machine learning and deep learning approaches are becoming incredibly efficient for this problem, and achieve a very high classification accuracy. Especially for deep learning, thanks to the power of Deep Neural Network, the model could have strong representation ability as long as provided large amount of labeled data.

However, the existing deep learning methods usually directly operate on the time series data itself. This brings a critical problem that the classification accuracy heavily relies on the DNN model itself, making the training process difficult. In other words, only when

providing enough training data and spend long time in the training phase, the DNN could achieve a good accuracy. In real world cases, collecting a large amount of labeled data requires significant efforts. Without a preprocessing the data and digging some features, feed the data into a blackbox is an abuse of the dataset and brings low efficiency.

To address this problem, we marry the traditional feature extraction methods with the existing deep learning approaches, and aim to achieve a very good performance using the extracted features. In particular, we borrow the idea of conventional high-order spectrum analysis, and use the high-order spectrum as the unique signature of each piece of time series data. Then we feed this signature (the high-order spectrum) into DNNs and output the classification results. Due to the uniqueness of high-order spectrum, it provides more information than the 1st order spectrum (the frequency response after a FFT operation) and these information should lead to better classification accuracy.

The rest of the report is organized as following: We first introduce the background knowledge of high-order spectral analysis in Section 2, then we introduce Deep Neural Network, particularly Convolutional Neural Network (CNN) in Section 3. In the next section, we briefly introduce the dataset we are using in this report, and give some examples of the high-order spectrum to show the potential of being used as the feature for classification problem. In Section 5, we perform extensive experiments and report our results. The conclusion is drawn in Section 6

2 High-order Spectral Analysis

2.1 1st Order Spectrum and Fourier Transform

Joseph Fourier showed that any periodic wave can be represented by a sum of simple sine waves. Also, there is a set of sine waves that, when summed together, are equal to any given wave [7]. These sine waves each has its own frequency, amplitude and phase position. To write this property in mathematical form, we can have the equation 1 shown below [9]:

$$f(t) = a_0 + \sum_k a_k * sin(kwt + \rho_k) \tag{1}$$

In this equation, t is the time point, a_k and ρ_k represents the amplitude and phase position of the sine wave with frequency k respectively, and a_0 means the amplitude of the sine wave with frequency equals to zero. Here, amplitude determines the relative presence of a sinusoid in the series, while phase determines how the sinusoids line up relative to one another to form the series.

Sometimes it is difficult to capture the specific feature on time domain of the time series, but when we convert it to spectrum form, those kinds of features may become obvious [8].

In order to decompose the time series to spectrum from, *Fourier Transform* is used in this section. The idea of Fourier transformation is to decompose a stationary time series X_t into a combination of sinusoids, with random and uncorrelated coefficients [3].

$$F_X(k) = \sum_{t=0}^{M-1} X(t) e^{-j\frac{2\pi}{M}tk} \tag{2}$$

Here, $k = 0, 1, \ldots, M-1$, which represents frequency, M is total length of the series, and $j = \sqrt{-1}$, which is the imaginary unit .

The result of this transformation is that corresponding to each sine wave with frequency k, we can get a complex number, with the mod of this complex number is the amplitude, and the angle is the phase.

To better understand the concept, we can see figure 1 Note that currently the mostly used approach to perform Fourier Transform is called Fast Fourier Transform, or FFT. It utilize the divide-and-conquer idea and split the sequence into odd and even parts, then recursively compute the transformation. Using FFT, the time complexity is reduced to $O(n \log n)$ if the sequence length is n.

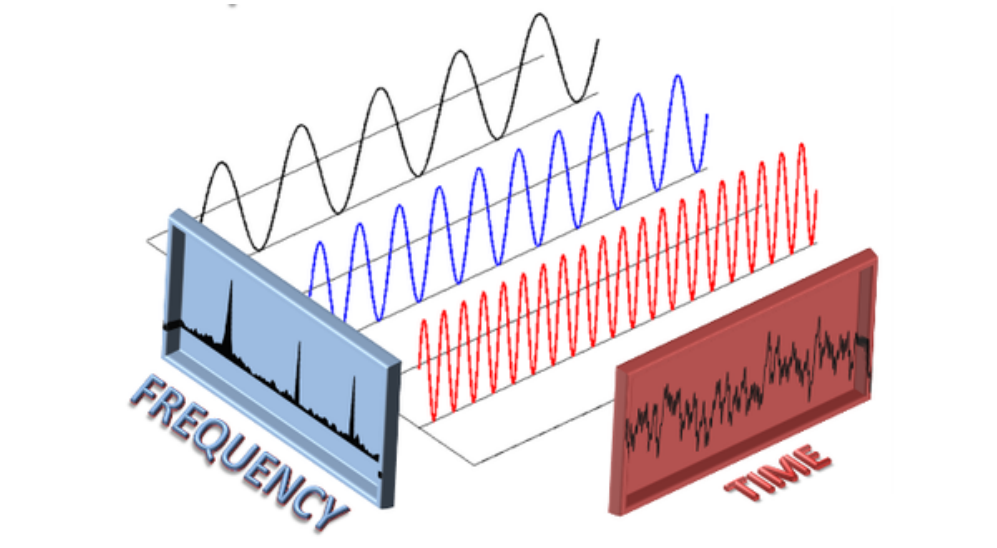

Figure 1: Fourier transformation

2.2 Bispectral Analysis

FFT and the 1st order frequency response is usually good enough for simple analysis, however, the non-linearity behavior of most time series data limit the use of 1st order spectrum analysis [5]. Linearity means the sequence $X(t)$ has a linear relationship such that

$$X(t) = \sum_{i=t-T}^{t-1} \alpha_i X(i) \tag{3}$$

Obviously this is usually not true for most of the time series datasets.

High-order spectrum are brought to address the problem. The complex non-linear autocorrelation of the time series dataset could be well identified by high-order spectrum. Surprisingly in 1st order, the spectrum of certain sequences might has nothing to reveal, meaning that all of them might be a similar nearly-flat line, while in high-order counterpart, it shows very different patterns for different time series sequences. With this huge advantage of high-

order spectrum, it is better to perform a transformation from the time domain to high-order spectrum, thus form and salient the specific feature of different class of time series.

Due to the complexity of the transformation for order > 2, we introduce the 2nd order spectrum, or bi-spectrum. We utilize the bi-spectrum in the rest of the reports. Although the higher orders, the uniqueness it could capture, it comes at a cost that orders more than 2 would results in very large feature space that is hard to feed into a deep learning model. More specifically, 2nd order spectrum will be in $\mathcal{R}^2$, third order in $\mathcal{R}^3$, etc. The trade-off between accuracy and speed is always a huge concern in practical study. So we make use of bi-spectrum and ignore higher orders. Another important fact is that the bi-spectrum in $\mathcal{R}^2$ can be viewed as an image, where image classification is one of the best-performing areas that a deep learning model could do. This also encourages us to use bi-spectrum in our experiments.

First we would provide the equation of bi-spectrum:

$$B_X(k_1, k_2) = \frac{1}{M} F_X(k_1) F_X(k_2) F_X(k_1 + k_2) \tag{4}$$

where $0 \leq k2 \leq k1, k1 + k2 \leq \frac{M}{2}$. Taking a close look, the bi-spectrum happens in $\mathcal{R}^2$, and in our discrete case, the indices are k_1 and k_2. This two indices determines the position in bi-spectrum graph. As it shows, the bi-spectrum response indexed by k_1, k_2 are a function of three 1st order spectrum responses. note that exchanging k_1 and k_2 would result in the same response values, thus we can see that bi-spectrum is a symmetric image.

3 Deep Learning Method

3.1 Convolutional Neural Network

Convolutional neural networks, which can be abbreviated as CNN, are deep artificial neural networks that are used primarily to classify images, cluster them by similarity and perform object recognition within scenes. Because bi-spectrum form of the time series data is essentially a 2-D image, then the time series classification problem could be converted to the image classification problem [2]. In an intuitive way, the algorithm works by analyzing and comparing several previously classified images (training process), testing and fine tuning the model (updating the weights), until converging into an accurate classification [10].

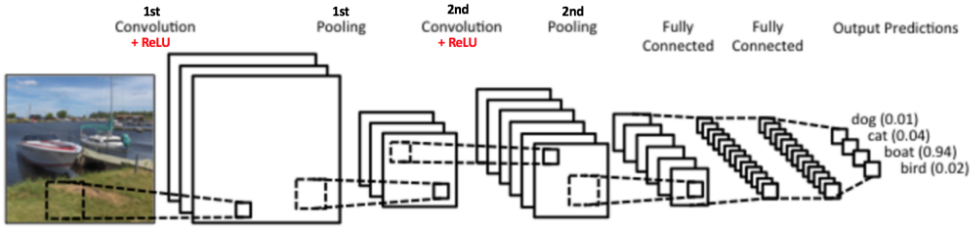

Figure 2: CNN Basic Procedure [6]

3.1.1 Convolution layer

The main purpose of adding the convolution layer is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. A filter slides over the input image to produce a feature map. The convolution of another filter over the same image gives a different feature map as shown [4].

3.1.2 Activation layer

The activation layer is mainly used to introducing non-linearity. Non-linear means that the output cannot be reproduced from a linear combination of the inputs. Without a non-linear activation function in the network, a CNN, no matter how many layers it had, would behave just like a single-layer perception, because summing these layers would give us just another linear function.

ReLU function, with full name being Rectified Linear Units, is the most popular function used in the activation layer to introduce non-linearity [1].

$$f(x) = \max(0, x) \tag{5}$$

It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

3.1.3 Pooling layer

Spatial Pooling can be of different types: Max, Average, Sum etc. Pooling layer can reduce the dimensionality of each feature map but retains the most important information. Also, it controls overfit and also introduces non-linearity (only Max pooling).

3.1.4 Fully Connected layer

Fully connected layer uses features captured before classifying the input image into various classes. "Fully Connected implies every neuron in the previous layer is connected to every neuron on the next layer. The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

4 Datasets and Their Spectrum

4.1 Data

The datasets we used are from UEA & UCR Time Series Classification Repository [1]. It is an ongoing project to develop a comprehensive repository for research into time series classification. Among the datasets, we chose Beef and Earthquakes datasets.

[1] http://timeseriesclassification.com/dataset.php.

Beef dataset is a quite small dataset, with train and test datasets both contain 30 series and length of each series is 470 . The beef dataset consists of four classes of beef spectrograms, from pure beef and beef adulterated with varying degrees of offal. Because food spectrograph are used in chemometrics to classify food types, this dataset is used in food safety and quality assurance.

To show that the bi-spectrum differs in each category, we plot the original time series sequence, along with its corresponding bi-spectrum, in Figure 3. From the figures we can see that although they looks similar in the original time domain, they shows different patterns in the bi-spectrum domain. In time domain, it might be difficult to tell the pattern but in the 2-D domain, the heatmap shows different concentration positions (high density peaks) for different categories.

Earthquakes is a larger dataset, with train size being 322 and test size being 139, and the length is 512. The data is taken from Northern California Earthquake Data Center and each data is an averaged reading for one hour, with the first reading taken on Dec 1st 1967, the last in 2003, and the total number of classes in this dataset is two.

5 Experiments

5.1 Experiment Settings

For the **Beef** dataset, first, we convert the series to stationary by taking a difference between the data points. So let us say, if the original time series was: $(X_1, X_2, X_3, \ldots, X_n)$, then series with difference of degree 1 becomes: $(X_2 - X_1, X_3 - X_2, X_4 - X_3, \ldots, X_n - X_{n-1})$. After that, we apply the bi-spectral transformation mentioned above and convert both the train set and the test set to bispectrum matrices.

Then, a five-layer convolutional neural network model is built based on those bispectrum matrices. In the first layer, we use 32 filters with each window size be 3*3, and choose activation function be ReLU. We then select pooling layer be maxpooling with pooling size be 2*2 to reduce the dimensionality. To avoid overfitting, the dropout size was chosen to be 25%. Then in the following two layers, same structure is applied in addition that we change the filter size be 64 and 32 respectively. After converting all the 2-D plots to 1-D vectors, two fully connected layers are then applied. We let the first fully connected unit size be 512 and dropout 50%, then the second fully connected unit size be 5, which is the total number of classes in this dataset, and used *softmax* to be the activation function in order to normalized the output to be in the range of 0 to 1.

We implement our model in a popular deep learning framework, Keras [4], backended by Tensorflow. The reason we choose this framework is its strong modeling capability and ease of use. For instance, all layers are stacked using one line of code each, and the computation graph will be automatically built and the data will be fed into the graph for processing.

To fit the model, *Adam* optimizer was used and also, *cross-entropy* was applied as the loss function when doing the model compiling. Batch size equals to 30 because of the size of training and testing sets, while 1000 epochs was used.

Earthquakes dataset is quite larger than the Beef one. Based on the CNN model built for *Beef*, we added two addition convolution layers between the first and second layers and

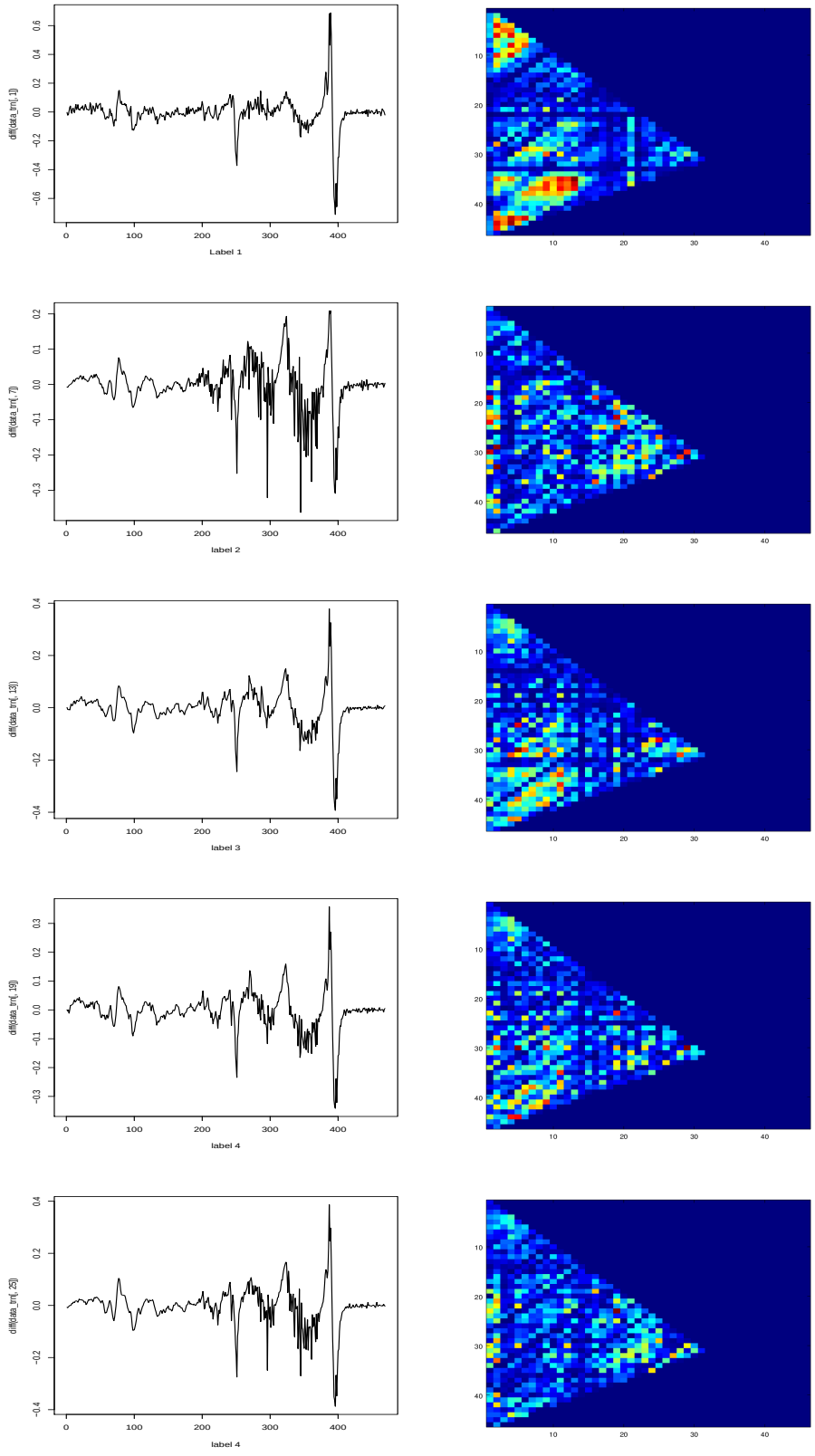

Figure 3: Time series plot and bispectrum heatmap. From the upper to the bottom, each row represents a label(i.e the first row in feature of series with label 1)

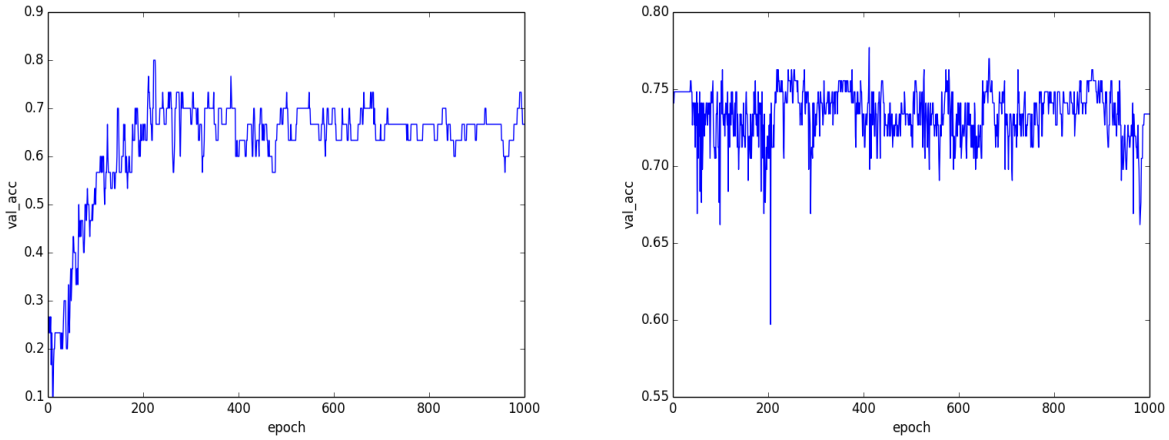

Figure 4: Validation accuracy of CNN model. Left is Beef, right is Earthquakes

let the filter size be 32 and 64 respectively. Then we changed the batch size be 32 while 1000 epochs and other parts kept the same. Validation accuracy curves of the two models are shown in figure 4.

5.2 Experiment Results

The best accuracy of **Beef** model in the whole learning process is 0.8, and the average accuracy in the last 100 learning is approximately 0.66. While the best accuracy of **Earthquakes** model in the whole learning process is 0.777, and the average accuracy in the last 100 learning is 0.73329. Result is summarized in Table 1.

Table 1: Summary of Result

Dataset	Train Size	Test Size	No of Classes	Best Accuracy	Average Accuracy
Beef	30	30	5	0.8	0.65968
Earthquakes	322	139	2	0.777	0.73329

6 Conclusions

In this study, we bring the idea of bi-spectrum for time series data analysis. Due to the fact that 1st order spectrum could not fully capture the hidden features of non-linear time series sequence, the bi-spectrum is a better signature for unique identification of non-linear time series sequence.

Deep learning models including Convolutional Neural Network has the advantage of high model capacity, thus we use CNN as our discriminator for the classification purpose.

Combining both idea, we first perform bi-spectrum transformation to map each time series sequence into bi-spectrum domain. The bi-spectrum could be viewed as an image. The

second phase is to use CNN to perform image classification, resulting in the classification of time series data. Comprehensive experiments are carried out to show the potential of our method, and proves that the bi-spectrum could be used as the extracted feature feeding into a CNN model, that provide satisfactory accuracy performance.

In the future, we can further tune the CNN model, as well as trying other DNN models, such as ResNet. The original time series data could also be used in addition to the bi-spectrum, to further boost the classification accuracy.

References

[1] Convolutional neural networks for visual recognition, 2018.

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, Online First, 2016.

[3] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[4] François Chollet. Keras official document, 2015.

[5] Jane L Harvill, Nalini Ravishanker, and Bonnie K Ray. Bispectral-based methods for clustering time series. *Computational Statistics & Data Analysis*, 64:113–131, 2013.

[6] Ujjwal Karn. An intuitive explanation of convolutional neural networks. *ujjwalkarn, August*, 2016.

[7] Joo Neto. Fourier transform: A r tutorial, 2018.

[8] Maurice Bertram Priestley. Spectral analysis and time series. 1981.

[9] Petre Stoica. *Introduction to spectral analysis*. Prentice hall, 1997.

[10] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1578–1585. IEEE, 2017.