Article

Parallel Frequency Function-Deep Neural Network for Efficient Approximation of Complex Broadband Signals

Zhi Zeng 1,†, Pengpeng Shi 2,3,*,†, Fulei Ma 1 and Peihan Qi 4

1 School of Mechano-Electronics Engineering, Xidian University, Xi’an 710071, China
2 School of Civil Engineering & Institute of Mechanics and Technology, Xi’an University of Architecture and Technology, Xi’an 710055, China
3 State Key Laboratory for Strength and Vibration of Mechanical Structures, Shaanxi Engineering Research Center of NDT and Structural Integrity Evaluation, School of Aerospace, Xi’an Jiaotong University, Xi’an 710049, China
4 School of Telecommunications Engineering, Xidian University, Xi’an 710071, China

* Correspondence: shpengpeng@xjtu.edu.cn
† These authors contributed equally to this work.

Abstract: In recent years, with the growing popularity of complex signal approximation via deep neural networks, people have begun to pay close attention to the spectral bias of neural networks—a problem that occurs when a neural network is used to fit broadband signals. An important direction taken to overcome this problem is the use of frequency selection-based fitting techniques, of which the representative work is called the PhaseDNN method, whose core idea is the use of bandpass filters to extract frequency bands with high energy concentration and fit them by different neural networks. Despite the method’s high accuracy, we found in a large number of experiments that the method is less efficient for fitting broadband signals with smooth spectrums. In order to substantially improve its efficiency, a novel candidate—the parallel frequency function-deep neural network (PFF-DNN)—is proposed by utilizing frequency domain analysis of broadband signals and the spectral bias nature of neural networks. A substantial improvement in efficiency was observed in the extensive numerical experiments. Thus, the PFF-DNN method is expected to become an alternative solution for broadband signal fitting.

Keywords: PFF-DNN; spectral bias; broadband signals; fast Fourier analysis

1. Introduction

An artificial neural network, also known as a neural network, refers to a mathematical model that imitates animal neural network behavior [1], which is essentially a high-dimensional complex mapping model wherein adjusting network weights allows for feature fitting. A neural network’s basic building blocks are a neuron model, a parameterized model nested by a scalar linear function, and a monotonic nonlinear function (activation function), where the coefficients in the linear function are the connection weights between neurons. Neurons connecting according to a specific topology form a neural network. One of the primary networks is the single-layer network composed of multiple neurons in parallel. Multiple single-layer networks can be stacked to obtain a multi-layer network and further expanded into a deep neural network that contains multiple multi-layer networks. Some advanced neural network models are designed to meet the needs of engineering practice [2–5]. For example, convolutional neural networks (CNN) and their evolution models have achieved unprecedented success in computer vision due to their powerful feature extraction capabilities and are widely used in security monitoring, autonomous driving, human-computer interaction, augmented reality, and many other fields. Additionally, recurrent neural networks, especially the long-short-term memory
model (LSTM), have become mainstream tools in the fields of automatic translation, text generation, and video generation in just a few years.

The related research of the universal approximation [6–10] indicates that assuming sufficient neurons and suitable weights, the neural network could approximate any continuous signals on a compact subset of $\mathbb{R}^n$ with arbitrary precision. However, it is not easy to obtain these appropriate weights via training for a complex network with too many neurons. The convergence speed of the neural network is related to the frequency spectrum of the fitted signal [11]. As shown in Figure 1, the neural network first learns the low-frequency components during the training process. The relationship between the convergence speed and the frequency of the fitted signal has been quantitatively analyzed [12]. When a network is applied to fit a signal, the required training time increases exponentially as the component’s central frequency increases. The spectral bias of the convergence speed in network training leads to unbearable training times for fitting the high-frequency components in broadband signals. In recent years, differential equation solving based on physical knowledge constraints has been successful in many applications, but multi-scale and multi-physics problems need further development. One of the core problems is that it is difficult for fully connected neural networks to learn high-frequency functions, that is, the spectral bias problem [13]. Solving this problem has become one of the core bottlenecks for the further development of A.I. technology.

![Figure 1](image-url)

**Figure 1.** Network fitting results given a different number of updates reflects that neural networks often learn low-frequency components first in the training process: (a) $N = 100$; (b) $N = 300$; (c) $N = 500$; and (d) $N = 1000$. The objective function for fitting: $f_1 = \sin 5x + \sin 7x + \sin 11x$. The network used a 1-40-40-40-1 fully connected structure type. The Adam optimizer was used for optimization under the default learning rate.

To this end, Cai et al. proposed the PhaseDNN method for fitting complex signals with high-frequency components via the combination of parallel frequency band extraction and frequency shifting techniques [14]. Numerous numerical experimental results have indicated that the PhaseDNN method successfully avoids the computational cost disaster when fitting signals with high-frequency components. However, although the method has good approximation efficiency for broadband signals, we found in our numerical experiments that for a large number of typical signals, especially those with smooth...
frequency domain distribution, the operation of fitting the inverse transformed signals can be further optimized, thus greatly improving the neural network fitting efficiency.

Therefore, our paper is dedicated to investigating a more efficient candidate method for fitting complex signals—signals with smooth frequency domain distribution. To reach this goal, a parallel frequency function-deep neural network (PFF-DNN) is proposed by utilizing the fast Fourier analysis of broadband signals and the spectral bias nature of neural networks. The effectiveness and efficiency of the proposed method are verified based on detailed experiments for six typical broadband signals. The discussion shows how the neural network adaptively constructs low-frequency smooth curves to interpolate discrete signals. This adaptive low-frequency approximation makes it possible to fit discrete frequency domain signals accurately.

The paper is organized as follows. Related works and the proposed method are introduced in Section 2. Extensive numerical experiments are presented in Section 3. Finally, the paper is concluded in Section 4.

2. Methods

2.1. Recent PhaseDNN

The research mentioned above indicates that it is difficult for neural networks to directly and accurately fit signals with high-frequency components. However, if we can transform the high-frequency components in the signal into smooth low-frequency signals convenient for neural network fitting, by some means such as the application of frequency shift technology, we can approximate the original signal via component-by-component fitting. Following this idea, in 2020, Cai et al. proposed the PhaseDNN method to implement a fitting scheme for signals with high-frequency components [14]. As shown in Figure 2, the PhaseDNN method involves four steps: (1) all high-frequency components in the original objective signal are extracted; (2) each high-frequency component is converted into a low-frequency signal using frequency shift technology; (3) different neural networks with identical structure are used to fit these low-frequency signals in parallel; and (4) inverse frequency shift operations are performed for all network predictions to obtain approximated high-frequency components, which are further summed up to recover the original signal.

Figure 2. Flowchart of the PhaseDNN algorithm and the proposed PFF-DNN method.
Figure 3 shows a comparison between the fitted results from the PhaseDNN method and those obtained via vanilla fitting. As shown in Figure 3a, vanilla fitting cannot recover high-frequency components well. If the signal oscillates even faster, the vanilla fitting becomes entirely ineffective. On the contrary, as shown in Figure 3c,d, the recent PhaseDNN method has shown obvious advantages in being able to completely characterize the information of all frequency bands of the signal.

It should be noted that the extracted frequency bandwidth needs to be minimized to ensure fitting accuracy. On the one hand, the smaller the extracted frequency band’s width $\Delta \omega$, the higher the fitting accuracy. On the other hand, a smaller $\Delta \omega$ indicates more frequency bands are extracted, which directly leads to an increase in computational overhead. When using PhaseDNN for broadband signal fitting, it is often necessary to extract all frequency bands and fit them separately. Therefore, there is a balance between accuracy and computational overhead. For example, consider the task of performing neural network fitting on a signal with a bandwidth of 300 Hz. When $\Delta \omega = 10$, considering the existence of both real and imaginary parts in the spectrum and the conjugate symmetry, 30 $\times$ 2 groups of neural networks need to be trained, which is an acceptable computational overhead. However, for the task of neural network fitting on a signal with a bandwidth of 3000 Hz, 3000 $\times$ 2 groups of neural networks need to be trained, which significantly increases the computational overhead.

2.2. The Proposed PFF-DNN Method

It is preferable to determine a method that can ensure fitting accuracy without requiring a large number of extracted frequency signals to be selected. Using the fast Fourier transformation (FFT) of broadband signals [15], the digital spectrum of the signal can be obtained much more efficiently. It is conceivable that if we perform a piecewise fitting on a signal’s digital spectrum, no computational overhead is required for frequency selection or
frequency shift. This avoids several redundant operations in the PhaseDNN method. In addition, when the frequency spectrum of the signal is not overcomplicated, the efficiency of fitting in the frequency domain will significantly improve. The method proposed here is abbreviated as the PFF-DNN method, and the details of its construction are as follows:

The objective signal is denoted as a real-valued \( f(x) \) in the domain of \([x_0, x_{n-1}]\). From a digital sampling system, one can obtain the discrete value of the signal \( f(x) \) at the sampling points \([x_0, x_1, \ldots, x_{n-1}]\). Here, the sampling points are assumed to be uniformly distributed on the interval of \([x_0, x_{n-1}]\), and the discrete value of the signal \( f(x) \) at the sampling points are denoted as \( f_0, f_1, \ldots, f_{n-1} \). One can calculate the frequency spectrum \( F(\omega) \) as:

\[
F(\omega_k) = F_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi kj/n},
\]

where \( k = 0, 1, \ldots, n-1 \), and \( \omega_k \) is evenly distributed. The adjacent interval between \( \omega_k \) depends on the sampling interval. \( \omega_k \) is conjugate symmetric when the signal is real-valued. Here, \( \{F_k\} \) is divided into \( m \) segments of length \( \Delta \omega \) in order, in which \( n = m \Delta \omega \):

\[
\{F_k\}_{k=0,1,\ldots,n-1} = \bigcup_{i=0}^{n-1} S_i,
\]

where

\[
S_i = \{F_k\}_{k=i\Delta \omega, i\Delta \omega+1,\ldots,(i+1)\Delta \omega-1},
\]

In the following, we define \( S_i \) as the \( i \)th segment of \( F(\omega) \). Considering the conjugate symmetry, only half of \( \{F_k\} \) needs fitting. Further considering that the sampling frequency is much larger than the bandwidth of the signal \( (\omega_k = b \Delta \omega) \), Equation (2) can be approximated by:

\[
\{F_k\}_{k=0,1,\ldots,n/2} \approx \left( \bigcup_{i=0}^{b/2} S_i \right) \cup \left( \bigcup_{i=b/2}^{m/2} O_i \right),
\]

where

\[
O_i = \{(0,0,\ldots,0)\},
\]

and

\[
F_{n-k} = F_k^* \quad k = 1, 2, \ldots, (n-1)/2,
\]

in which * stands for the conjugate operation. For each slice \( S_i \), a neural network is used to approximate the information \( T_i \) contained. More precisely:

\[
T_i(k) \approx \begin{cases} 
F_k & k = i\Delta \omega, i\Delta \omega + 1, \ldots, i\Delta \omega + \Delta \omega - 1 \\
0 & \text{otherwise}
\end{cases}
\]

where \( i = 1, 2, \ldots, b \). One can see from Equation (7) that, different from PhaseDNN [14], in the proposed method, each neural network \( T_i \) is used to “memorize” the discrete data in the frequency domain. After obtaining these trained neural networks and their predictions \( \{T_i\} \), the following concatenation operation is used to obtain the approximation of \( F(\omega) \):

\[
F_k \approx T(k) = \begin{cases} 
\sum_{i=0}^{b} T_i(k) & k = 0, 1, \ldots, (n-1)/2 \\
\sum_{i=0}^{b} T_i^*(n-k) & k = (n+1)/2, (n+1)/2 + 1, \ldots, n-1
\end{cases}
\]

Finally, each sampling signals \( f_j \) can be approximated using the following inverse FFT (IFFT):

\[
f_j \approx \frac{1}{n} \sum_{k=0}^{n-1} T(k) e^{i2\pi jk/n},
\]
Let us compare the recent PhaseDNN method and the proposed PFF-DNN method in terms of ease of operation. In PhaseDNN, $2m$ convolutions are required to acquire all frequency-selected signals. Then, $2m$ frequency shifts and $2m$ inverse frequency shifts are also involved. In comparison, the proposed method is more straightforward, and only one FFT and one IFFT are required. No frequency selection, frequency shifts, nor inverse shifts are required.

In the PhaseDNN method, DNNs are used to fit continuous signals in the time domain. On the contrary, in the proposed method, DNNs are used to “memorize” discrete data points in the frequency domain. For many commonly used signals (including the signals recommended in [14] to demonstrate the effectiveness of the PhaseDNN method), detailed numerical comparisons are conducted to show that the neural network “memorizes” such discrete frequency values much faster.

3. Numerical Experiments
3.1. Experimental Setting

Six typical signal analyses were performed to evaluate the fitting performance of our PFF-DNN method, accompanied by a comparison with the existing PhaseDNN method. The signals to be analyzed included some periodic and non-periodic analog signals in the form of explicit functions and complex signals described by the classical Mackey–Glass differential dynamic system. For each signal, detailed comparative evaluations were conducted to describe the effectiveness and efficiency of the different methods for fitting the broadband signals with high-frequency components. The detailed comparisons included (1) convergence curves (the changes of the loss $R$ concerning the number of updates $N$ of the neural network weights); (2) the convergence process of different frequency bands; (3) the influence of different $\Delta \omega$ on the convergence speed; and (4) the influence of different updating times $N$ on the fitting accuracy.

Before showing the experimental results, we first introduce the neural network model and training-related parameters for all subsequent experiments. The neural network structure used in this study was consistent with the network used in [14] (i.e., the 1–40–40–40–40–1 fully connected neural network). Each layer (except the output layer) used the ‘ELU’ activation function [16]. The neural network training adopted the Adam optimizer [17] with a learning rate of 0.0002. The length of each training batch was 100. For the method in [14], 5000 training data and 5000 testing data were evenly distributed across the domain. For the method proposed in this paper, each slice’s length of the training sample depended on the slice’s length in the frequency domain. For both methods, we recorded the approximation error in the root mean square error (RMSE) when $\Delta \omega$ of the slice in the frequency domain was 11, 21, 31, 41, or 51, and when the total number of updates were from 1 to 10,000. The $\Delta \omega$ used in the literature [14] was 11. Therefore, we gradually increased $\Delta \omega$ from this value. When $\Delta \omega$ increases to infinite, both methods will degenerate into the vanilla fitting method. Therefore, we set the upper limit of $\Delta \omega$ to 51. The numerical experiments were completed using a PC with an Intel-core-i9-9900X CPU @ 3.50 GHz with 128 GB-RAM and a Titan X GPU.

3.2. Signals with Explicit Expression
3.2.1. Sine on Polynomial Signal

Here, the signal of sine on polynomial was used to test the neural network’s approximation accuracy, which is similar to the function used in the literature [18]. It is described by Equation (10), whose shape and the corresponding frequency spectrum are shown in Figure 4. The signal consists of a high-frequency periodic component and a low-frequency non-periodic component.

$$f(x) = 0.1x^3 - 0.1x^2 - 0.5x + 0.3 + \sin(50x),$$  (10)
Figure 4. Signal described by Equation (10) with its frequency spectrum. (a) The function to be fitted. (b) The spectrum of the function.

As shown in Figure 5, only the low-frequency component can be approximated if we use the vanilla fitting method. No matter how large $N$ is, the high-frequency component cannot be fitted.

Figure 5. Function fitting using vanilla fitting: (a) $N = 100$; (b) $N = 5000$; and (c) $N = 10,000$.

On the contrary, if the PhaseDNN or PFF-DNN method was used, the signal was fitted very well. As mentioned earlier, a larger $\Delta \omega$ indicates fewer neural networks used for fitting, which is desirable. For the current example, the bandwidth of the signal is 50 Hz. When $\Delta \omega = 11$, to approximate the signal precisely, one needs to use at least five networks for fitting. In comparison, when $\Delta \omega = 51$, one only needs two networks for fitting, which saves 80% of the computation resources. However, an increase in $\Delta \omega$ makes the fitting harder.

Figure 6 shows how the convergence process of the PhaseDNN and PFF-DNN methods changed with $\Delta \omega$. It was shown that both algorithms converged fast when $\Delta \omega$ was small. However, when $\Delta \omega$ increased, the convergence speed of both methods slowed down. Note that, compared with the PhaseDNN method, the PFF-DNN algorithm was less sensitive to the increase in $\Delta \omega$. In detail, for the PhaseDNN method, when $\Delta \omega = 11$, the network updated 10,000 times to reach convergence. However, when $\Delta \omega = 41$, it was difficult to converge even if it was trained 100,000 times. In contrast, for the PFF-DNN method, when $\Delta \omega = 41$, it converged within only 2000 updates. That is to say, when $\Delta \omega$ tripled, the convergence time only doubled. This is desirable in practical applications.
Figure 6. The influence of Δω on the convergence process.

Figure 7 shows the fitting results of different frequency bands. Since the frequency spectrum for the neural network to approximate is not overcomplicated, the performance of the PFF-DNN method was better than that of the PhaseDNN method.

Figure 7. Comparison between the fitting results of the PhaseDNN and PFF-DNN methods for different frequency bands. The fitting results after neural networks were updated 5000 times given Δω = 31: (a) first frequency band (PhaseDNN); (b) fourth frequency band (PhaseDNN); (c) first frequency band (PFF-DNN); and (d) fourth frequency band (PFF-DNN).

Figure 8 shows the influence of Δω on the algorithm performance. As shown in Figure 8c,f, when Δω = 11, both methods fit the signal well. However, when Δω increased
to 31, the PhaseDNN method began to show fitting errors. When \( \Delta \omega = 51 \), the PhaseDNN method was hardly able to fit the signal effectively. In comparison, the PFF-DNN method was able to still accurately fit the signal.

Figure 8. The influence of \( \Delta \omega \) on the algorithm performance. Fitting results of the PhaseDNN method after 10,000 updates: (a) \( \Delta \omega = 51 \); (b) \( \Delta \omega = 31 \); and (c) \( \Delta \omega = 11 \). The fitting results of the PFF-DNN method after 10,000 updates: (d) \( \Delta \omega = 51 \); (e) \( \Delta \omega = 31 \); and (f) \( \Delta \omega = 11 \).

Figure 9 shows how the number of updates \( N \) influences the algorithm performance. In the beginning, both algorithms were unable to fit the high-frequency components very well. The comparison between Figure 9b,e shows that the PFF-DNN method converged faster compared with the PhaseDNN method. The red curves in Figure 6 also confirm this point.

Figure 9. The influence of \( N \) on the algorithm performance. Fitting results of the PhaseDNN method when \( \Delta \omega = 11 \): (a) \( N = 100 \); (b) \( N = 1000 \); and (c) \( N = 10,000 \). Fitting results of the PFF-DNN method when \( \Delta \omega = 11 \): (d) \( N = 100 \); (e) \( N = 1000 \); and (f) \( N = 10,000 \).
3.2.2. ENSO Signal

The signal determined by the following equation is often used to approximate the ENSO data set [19]:

\[
 f(x) = \begin{align*}
 4.7 \cos(2\pi x/12) + & 1.1 \sin(2\pi x/12) \\
 + & 0.2 \cos(2\pi x/1.7) + 2.7 \sin(2\pi x/1.7) \\
 + & 2.1 \cos(2\pi x/0.7) + 2.1 \sin(2\pi x/0.7) \\
 - & 0.5
\end{align*}
\] (11)

This signal is more complicated than the one described by Equation (10). The shape of the signal and its corresponding frequency spectrum is shown in Figure 10.

According to the trend of the solid red line and the red dashed line in Figure 11, it was shown that the convergence speed of the PFF-DNN method was about an order of magnitude larger. With the increase in \( \Delta \omega \), the convergence time of the PFF-DNN method only doubled. Thus, the larger \( \Delta \omega \), the more training time the PFF-DNN method saves as compared to the PhaseDNN method. Figures 12 and 13 visually show the convergence process of the two networks.

**Figure 10.** Signal described by Equation (11) with its frequency spectrum. (a) The function to be fitted. (b) The spectrum of the function.

**Figure 11.** The influence of \( \Delta \omega \) on the convergence process.
Figure 12. The influence of Δω on the algorithm performance. The fitting results of the PhaseDNN method after 10,000 updates: (a) Δω = 51; (b) Δω = 31; (c) Δω = 11. The fitting results of the PFF-DNN method after 10,000 updates: (d) Δω = 51; (e) Δω = 31; (f) Δω = 11.

Figure 13. The influence of N on the algorithm performance. Fitting results of the PhaseDNN method when Δω = 11: (a) N = 100; (b) N = 1000; and (c) N = 10,000. Fitting results of the PFF-DNN method when Δω C= 11: (d) N = 100; (e) N = 1000; and (f) N = 10,000.
3.2.3. Signal with Increasing Frequency

Next, a signal with fast frequency changes was analyzed. This signal is often used in system identification [20], and its shape and frequency spectrum are shown in Figure 14. The explicit expression of the signal used here is:

\[ f(x) = \cos \left( \pi \left( f_0 + \frac{f_T - f_0}{T} x^3 \right) x^3 \right), \quad (12) \]

in which, \( T = 1 \), \( f_0 = 0.01 \), and \( f_T = 50 \). Figure 14a shows that as \( x \) increases, the oscillation frequency of the signal increases sharply. Figure 14b shows that the amplitude and the oscillation frequency of the signal’s spectrum gradually decrease.

![Figure 14](image-url)

**Figure 14.** Signal described by Equation (12) with its frequency spectrum. (a) The function to be fitted. (b) The spectrum of the function.

From the curve in Figure 15, it is shown that when \( \Delta \omega = 11 \), the efficiency and accuracy of the PFF-DNN method were better than that of the PhaseDNN method. When \( \Delta \omega \) increased, the convergence speed of both methods slowed down rapidly. Note that the convergence speed of the PFF-DNN network was relatively faster. Figures 16 and 17 visually compare the convergence process of the two methods.

![Figure 15](image-url)

**Figure 15.** The influence of \( \Delta \omega \) on the convergence process.
3.2.4. Piecewise Signal

The piecewise signal was used in [14] to test the neural networks’ performance. The shape and its corresponding frequency spectrum are shown in Figure 18. The signal is described by:

Figure 16. The influence of $\Delta \omega$ on the algorithm performance. The fitting results of the PhaseDNN method after 10,000 updates: (a) $\Delta \omega = 51$; (b) $\Delta \omega = 31$; and (c) $\Delta \omega = 11$. The fitting results of the PFF-DNN method after 10,000 updates: (d) $\Delta \omega = 51$; (e) $\Delta \omega = 31$; and (f) $\Delta \omega = 11$.

Figure 17. The influence of $N$ on the algorithm performance. Fitting results of the PhaseDNN method when $\Delta \omega = 11$: (a) $N = 100$; (b) $N = 1000$; and (c) $N = 10,000$. Fitting results of the PFF-DNN method when $\Delta \omega = 11$: (d) $N = 100$; (e) $N = 1000$; and (f) $N = 10,000$. 
3.2.4. Piecewise Signal

The piecewise signal was used in [14] to test the neural networks’ performance. The shape and its corresponding frequency spectrum are shown in Figure 18. The signal is described by:

\[
f(x) = \begin{cases} 
10(\sin x + \sin 3x) & x \in [-\pi, 0] \\
10(\sin 23x + \sin 137x + \sin 203x) & x \in [0, +\pi]
\end{cases}, \tag{13}
\]

![Figure 18. Signal described by Equation (13) with its frequency spectrum. (a) The function to be fitted. (b) The spectrum of the function.](image)

The spectrum of this signal has several obvious spikes and many slight oscillations. Figure 19 shows that for such a signal, the accuracy of the proposed method was much less sensitive to changes in \(\Delta \omega\). Figures 20 and 21 visually compare the convergence process of the two networks.

![Figure 19. The influence of \(\Delta \omega\) on the convergence process.](image)
Figure 20. The influence of $\Delta \omega$ on the algorithm performance. The fitting results of the PhaseDNN method after 10,000 updates: (a) $\Delta \omega = 51$; (b) $\Delta \omega = 31$; and (c) $\Delta \omega = 11$. The fitting results of the PFF-DNN method after 10,000 updates: (d) $\Delta \omega = 51$; (e) $\Delta \omega = 31$; and (f) $\Delta \omega = 11$.

Figure 21. The influence of $N$ on the algorithm performance. Fitting results of the PhaseDNN method when $\Delta \omega = 11$: (a) $N = 100$; (b) $N = 1000$; and (c) $N = 10,000$. Fitting results of the PFF-DNN method when $\Delta \omega = 11$: (d) $N = 100$; (e) $N = 1000$; and (f) $N = 10,000$.
3.2.5. Square Wave Signal

The following example was used to test the proposed method’s approximation accuracy on discontinuous signals such as square waves [14]. The shape and its corresponding frequency spectrum are shown in Figure 22. The signal is described by:

\[ f(x) = \sin(x) + \text{sign}[\sin(13x)] + \text{sign}[\sin(23x)] + \text{sign}[\sin(47x)], \]  

(14)

![Figure 22](image.png)

**Figure 22.** Signal described by Equation (14) with its frequency spectrum. (a) The function to be fitted. (b) The spectrum of the function.

The spectrum of this signal is composed of many irregular spikes. From Figure 23, it is shown that the increase in \( \Delta \omega \) had a limited impact on the PFF-DNN method but had a significant impact on the PhaseDNN method. Figures 24 and 25 visually compare the convergence process of the two networks.

![Figure 23](image.png)

**Figure 23.** The influence of \( \Delta \omega \) on the convergence process.
3.3. Dynamic System

The Mackey-Glass equation is a time-delayed differential equation first proposed to model white blood cell production [21]. This equation has been used to test signal approximations in many related works [22–24]. It is described by the solution of the following lagged differential equation:

Figure 24. The influence of $\Delta \omega$ on the algorithm performance. The fitting results of the PhaseDNN method after 10,000 updates: (a) $\Delta \omega = 51$; (b) $\Delta \omega = 31$; and (c) $\Delta \omega = 11$. The fitting results of the PFF-DNN method after 10,000 updates: (d) $\Delta \omega = 51$; (e) $\Delta \omega = 31$; and (f) $\Delta \omega = 11$.

Figure 25. The influence of $N$ on the algorithm performance. Fitting results of the PhaseDNN method when $\Delta \omega = 11$: (a) $N = 100$; (b) $N = 1000$; and (c) $N = 10,000$. Fitting results of the PFF-DNN method when $\Delta \omega = 11$: (d) $N = 100$; (e) $N = 1000$; and (f) $N = 10,000$.

3.3. Dynamic System

The Mackey-Glass equation is a time-delayed differential equation first proposed to model white blood cell production [21]. This equation has been used to test signal

\[ f(x) = \frac{x(t-18)}{1 + x(t-18)^10} - \frac{x(t-8)}{1 + x(t-8)^10} \]
approximations in many related works [22–24]. It is described by the solution of the following lagged differential equation:

\[ y(x) = \frac{0.2y(x - 30)}{1 + y^{10}(x - 30)} - 0.1y(x), \]  

(15)

The shape of the signal and its corresponding frequency spectrum is shown in Figure 26.

Figure 26. Signal described by Equation (15) with its frequency spectrum. (a) The function to be fitted. (b) The spectrum of the function.

As shown, this signal’s frequency spectrum is more complicated. It is distributed irregularly within the interval of [0, 200]. Thus, this problem is relatively hard for both the PhaseDNN and PFF-DNN method.

Figure 27 shows that when \( \Delta \omega = 21 \) and \( \Delta \omega = 41 \), the training advantage of the PFF-DNN over the PhaseDNN method was relatively obvious. However, when \( \Delta \omega = 11 \) and \( \Delta \omega = 51 \), the advantage was less obvious. Figures 28 and 29 show that in the middle of training, the convergence speed of the PFF-DNN method was also slightly better.

Figure 27. The influence of \( \Delta \omega \) on the convergence process.
Figure 28. The influence of $\Delta \omega$ on the algorithm performance. The fitting results of the PhaseDNN method after 10,000 updates: (a) $\Delta \omega = 51$; (b) $\Delta \omega = 31$; and (c) $\Delta \omega = 11$. The fitting results of the PFF-DNN method after 10,000 updates: (d) $\Delta \omega = 51$; (e) $\Delta \omega = 31$; and (f) $\Delta \omega = 11$.

Figure 29. The influence of $N$ on the algorithm performance. Fitting results of the PhaseDNN method when $\Delta \omega = 11$: (a) $N = 100$; (b) $N = 1000$; and (c) $N = 10,000$. Fitting results of the PFF-DNN method when $\Delta \omega = 11$: (d) $N = 100$; (e) $N = 1000$; and (f) $N = 10,000$.

From this example, it is observed that when the signal’s frequency spectrum is relatively complicated, the performance gain of the PFF-DNN method was not significant. Therefore, we believe that the PFF-DNN and PhaseDNN methods may have their advantages for different applications.
3.4. Further Discussions

From the above experiments of fitting a large number of typical signals of different categories, it was observed that the proposed method had two advantages when the frequency domain distribution of the signal is smoother: (1) the proposed method had obvious training efficiency advantages without losing accuracy; (2) the proposed method was less sensitive to the choice of bandwidth. On the surface, this was due to the smoothness of the signal frequency domain distribution, but the deeper reason comes from the difference in complexity of the same segment of the signal in different domains.

In the following, we explain why the PFF-DNN method performed better in the above examples from the perspective of the spectral bias of neural networks. To this end, Figure 30 shows how the PhaseDNN and PFF-DNN methods approximated the third frequency band when $\Delta \omega = 11$ during the training process of the example in Section 3.2.3. Figure 30a,b shows that the PhaseDNN method essentially handled a continuous signal in the time domain, while the PFF-DNN method essentially interpolated a frequency domain discrete signal. Figure 30d,f shows the approximated results from the PFF-DNN method as a continuous function, which means that the PFF-DNN method interpolated the discrete spectrum with a smooth curve. Compared with Figure 30c,e, it is observed that the curve approximated by the PhaseDNN method oscillated more frequently in the time domain. Thus, it was more difficult for the neural network to learn the signals in the time domain by considering the spectral bias. This discussion may explain why the PFF-DNN method performed better than the PhaseDNN method for these examples.

![Figure 30](attachment:image.png)

**Figure 30.** Approximation performance during training. (a) The third frequency band of the fitted function (the real part after frequency shift); (b) the spectrum (real part) of the function in (a); (c) fitting results of PhaseDNN after 1000 updates; (d) fitting results of PFF-DNN after 1000 updates; (e) fitting results of PhaseDNN after 10,000 updates; and (f) fitting results of PFF-DNN after 10,000 updates.
4. Conclusions

Due to the existence of spectral bias, problems occur when a neural network is used for the approximation of broadband signals. That is, the high-frequency parts are hard to fit. For this reason, people often use frequency-selective filters to extract signals in different frequency bands and then use different neural networks to fit them. However, we found in a large number of numerical experiments that this method is inefficient for fitting signals with broad and smooth spectrums. For this reason, this paper proposed a novel alternative method based on parallel fitting in the frequency domain by utilizing frequency domain analysis of broadband signals and the spectral bias nature of neural networks. A substantial improvement in efficiency was observed in extensive numerical experiments. Thus, the PFF-DNN method is expected to become an alternative solution for broadband signal fitting.

In the future, we will further investigate how to determine whether the same signal is easier to fit by a neural network in the time domain or frequency domain and extend our findings to more complex representation spaces.

Author Contributions: Conceptualization, Z.Z. and P.S.; methodology, Z.Z. and P.S.; validation, F.M. and P.Q.; formal analysis, Z.Z. and P.S.; investigation, F.M. and P.Q.; writing—original draft preparation, Z.Z. and P.S.; writing—review and editing, Z.Z.; supervision, Z.Z.; project administration, P.S; funding acquisition, Z.Z. and P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant numbers 61805185, 11802225, and 51805397.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The details of the approach discussed herein and the specific values of the parameters considered have been provided in the paper. Hence, we are confident that the results can be reproduced. Readers interested in the code are encouraged to contact the authors by e-mail.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning; MIT Press: Cambridge, UK, 2016.
2. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Comput. 2017, 29, 2352–2449. [CrossRef] [PubMed]
3. Graves, A.; Mohamed, A.R.; Hinton, G. Speech Recognition with Deep Recurrent Neural Networks. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings, Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5999–6009.
5. Sirignano, J.; Spiliopoulos, K. DGM: A Deep Learning Algorithm for Solving Partial Differential Equations. J. Comput. Phys. 2018, 375, 1339–1364. [CrossRef]
6. Kidger, P.; Lyons, T. Universal Approximation with Deep Narrow Networks. In Proceedings of the Thirty Third Conference on Learning Theory, Graz, Austria, 9–12 July 2020; pp. 2306–2327.
7. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. Math. Control. Signals Syst. 1989, 2, 303–314. [CrossRef]
8. Barbu, A.; Mou, H. The Compact Support Neural Network. Sensors 2021, 21, 8494. [CrossRef] [PubMed]
9. Weinan, E.; Yu, B. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. Commun. Math. Stat. 2018, 6, 1–12. [CrossRef]
10. Cai, W.; Xu, Z.-Q.J. Multi-Scale Deep Neural Networks for Solving High Dimensional PDEs. arXiv 2019, arXiv:1910.11710.
11. Rahaman, N.; Baratin, A.; Arpit, D.; Draxler, F.; Lin, M.; Hamprecht, F.A.; Bengio, Y.; Courville, A. On the Spectral Bias of Neural Networks. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; pp. 9230–9239.
12. Xu, Z.Q.J.; Zhang, Y.; Luo, T.; Xiao, Y.; Ma, Z. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. Commun. Comput. Phys. 2020, 28, 1746–1767. [CrossRef]
13. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-Informed Machine Learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [CrossRef]

14. Cai, W.; Li, X.; Liu, L. A Phase Shift Deep Neural Network for High Frequency Approximation and Wave Problems. *SIAM J. Sci. Comput.* **2020**, *42*, A3285–A3312. [CrossRef]

15. Nussbaumer, H.J. Fast Fourier Transform and Convolution Algorithms. *Proc. IEEE* **1982**, *70*, 527.

16. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016—Conference Track Proceedings, San Juan, Puerto Rico, 2–4 May 2016.

17. Kingma, D.P.; Ba, J.L. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015.

18. Yang, S.; Ting, T.O.; Man, K.L.; Guan, S.U. Investigation of Neural Networks for Function Approximation. *Procedia Comput. Sci.* **2013**, *17*, 586–594. [CrossRef]

19. Fritsch, F.N.; Kahaner, D.; Moler, C.; Nash, S. Numerical Methods and Software. *Math. Comput.* **1990**, *55*, 865. [CrossRef]

20. Xia, X.G. System Identification Using Chirp Signals and Time-Variant Filters in the Joint Timefrequency Domain. *IEEE Trans. Signal Process.* **1997**, *45*, 820. [CrossRef]

21. Mackey, M.C.; Glass, L. Oscillation and Chaos in Physiological Control Systems. *Science* **1977**, *197*, 287–289. [CrossRef] [PubMed]

22. Lu, Y.; Sundararajan, N.; Saratchandran, P. A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks. *Neural Comput.* **1997**, *9*, 461–478. [CrossRef] [PubMed]

23. Jones, R.D.; Lee, Y.C.; Barnes, C.W.; Flake, G.W.; Lee, K.; Lewis, P.; Qian, S. Function Approximation and Time Series Prediction with Neural Networks. In Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks, San Diego, CA, USA, 17–21 June 1990; pp. 649–665.

24. Wu, S.; Er, M.J. Dynamic Fuzzy Neural Networks—A Novel Approach to Function Approximation. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2000**, *30*, 358–364. [CrossRef]