Automated Detection and Characterization of Singularities in Functions using Neural Networks - from FFT Signals

Zheng Chen[1a], Seulip Lee[b], Lin Mu[2c]

[a]*Department of Mathematics, University of Massachusetts, Dartmouth, MA, 02747, zchen2@umassd.edu*
[b]*Department of Mathematics, Tufts University, Medford, MA 02155, seulip.lee@tufts.edu*
[c]*Department of Mathematics, University of Georgia, Athens, GA 30602, linmu@uga.edu*

Abstract

Singularities, distinctive features signifying abrupt changes in function behavior, hold pivotal importance across numerous scientific disciplines. Accurate detection and characterization of these singularities are essential for understanding complex systems and performing data analysis. In this manuscript, we introduce a novel approach that employs neural networks and machine learning for the automated detection and characterization of singularities based on spectral data obtained through fast Fourier transform (FFT). Our methodology uses neural networks trained on known singular functions, along with the corresponding singularity information, to efficiently identify the location and characterize the nature of singularities within FFT data from arbitrary functions. Several tests have been provided to demonstrate the performance of our approach, including singularity detection for functions with single singularities and multiple singularities.

Keywords: Deep Neural Network, Singularity detection, Spectral data

1. Introduction

Singularities, characterized by abrupt changes or discontinuities in functions, are fundamental features encountered in various scientific and engineering domains. Accurate identification and characterization of singularities play a crucial role in understanding the behavior and properties of functions. The potential applications of detecting singularities in Fourier signals are diverse and extend across various fields where signal processing and analysis are essential, such as image processing, biomedical signal processing, environmental monitoring, and financial signal processing. Fourier analysis is widely used in *image processing* for compression, filtering, and feature extraction tasks. Detecting singularities in Fourier-transformed images can help in identifying salient features, discontinuities, or edges in the image domain. These features reinforce object detection, image segmentation, or image enhancement. Fourier analysis is often employed for electroencephalography and electrocardiography signal analysis in *biomedical signal processing*. Detecting singularities in Fourier-based biomedical signals can help recognize abnormal patterns or events, which leads to more accurate diagnosis and effective monitoring of medical conditions. Through Fourier analysis, *environmental monitoring* includes analyzing seismic, oceanographic, or atmospheric data signals. Detecting singularities in such data signals can help predict anomalous events such as earthquakes,

[1]Zheng Chen is partially supported by UMass Dartmouth's Marine and Undersea Technology (MUST) Research Program funded by the Office of Naval Research (ONR) under Grant No. N00014-23-1–2141. This material is also partially based upon work supported by the National Science Foundation under Grant No.DMS-1929284 while the author was in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the semester program.
[2]Lin Mu's work was supported in part by the U.S. National Science Foundation grant DMS-2309557.

tsunamis, or atmospheric disturbances. This identification contributes to early warning systems and disaster management efforts. In addition, Fourier analysis enables time series analysis, volatility modeling, and frequency domain analysis in *finance*. Detecting singularities in financial signal processing can help indicate significant events or anomalies in the data, which can be valuable for risk management, trading strategies, and economic forecasting.

Despite their importance and wide applications, identifying and characterizing singularities have been challenging, especially in cases with multiple singularities. Fourier analysis suffers from the Gibbs phenomenon, which leads to spurious oscillations around jumps and singularities in discrete Fourier series. These nonphysical oscillations make it challenging to visually identify the characteristics of the jumps and singularities. Various data reconstruction techniques, such as Gegenbauer reconstructions, have been proposed to recover spectral accuracy up to the jumps [28, 24, 27, 25, 26, 1, 30] and singularities [11, 12, 35]. These techniques have found widespread applications in post-processing numerical simulations [42, 23, 29, 34] and image reconstructions [2, 3, 4, 8]. However, such reconstruction techniques still require accurate information on the location and type of trouble points. Various edge detection methods have been devised for Gegenbauer reconstructions on discontinuous functions [21, 22, 18] based on truncated spectral expansions or collocation point values. These methods have proved successful across diverse input data types, including spectral partial sums and discontinuous Galerkin solutions [8, 7, 5, 45, 6, 19, 43, 41, 46, 14, 39, 20, 16, 17]. However, these methods focus on detecting discontinuities (not general singularities). A local singularity detection algorithm [37] has been developed to obtain high-order evaluations of singularity characteristics, such as location and exponent, through locally supported quasi-interpolation of univariate nonsmooth functions. Moreover, wavelet transforms have been employed effectively to identify Lipschitz regularity and characterize singularities in irregular signals, which leads to successful applications in signal denoising [38, 9, 44]. These techniques have been applied to various domains, including sinogram imaging [33], seismic imaging [32], and cone beam CT breast imaging [49]. Nevertheless, identifying various types of singularities and detecting multiple singularities in a function still need to be studied.

Neural network techniques have shown remarkable success in many fields, including computer vision [31], pattern recognition [40], natural language processing [15], and other tasks related to artificial intelligence. With the success of feature extraction in various research areas [13], neural network techniques have attracted significant attention for data-related applications. Machine learning techniques have recently been used to detect singularities in data, e.g., topological data and patterns [36, 47, 48]. Therefore, this paper proposes a neural network-based approach to automating the detection and characterization of singularities in functions. Our methodology uses a training dataset of randomly generated functions with known singularity locations and exponents. Fourier coefficients are computed for these functions, capturing their frequency domain characteristics. These coefficients, along with the corresponding singularity information, are used to train a neural network model detecting the underlying patterns and relationships. The neural network architecture is designed to handle the detection of one or multiple singularities, taking the Fourier coefficients as inputs and predicting both the singularity locations and exponents. Using neural network structures, we present three singularity detection models:

1. *Single singularity detector*: This detector approximates the singularity locations and exponents for functions with single singularities.
2. *Multiple singularities detector*: This detector extends the above detector to capture the locations and exponents of multiple singularities in singular functions.
3. *Multiple singularities detector with splitting strategy*: This detector enhances the above detector by employing the splitting strategy in detecting location and exponents separately. The location detector approximates the locations of multiple singularities first, and the exponent detector detects the exponents using input Fourier data and the detected locations. With this splitting strategy, we propose a *parallel* learning process to speed up the *serial* detector.

Through experimental results, we discuss how to generate datasets and set up detection models, including the number of Fourier coefficients, the number of generated singular functions, and the number of hidden layers. Performance metrics such as mean squared errors are used to quantify the accuracy of the predictions from the models. To maximize the performance of our models, we use optimization algorithms with adaptive learning rates and batch processing and apply validation sets to decrease learning rates and stop learning iterations. Moreover, we test our detection models while applying artificial noise to Fourier data, mimicking real data processing. The experimental results show that our detectors effectively identify the singularity's location and exponent from Fourier data with noise.

Detailed discussions regarding dataset generation, detection model training and validation are given in our test. These tests show the accuracy, effectivity, efficiency, and robustness of singularity detection. Moreover, for the noisy data, our model is able to predict the singularity with high fidelity. The predicted singularity locations and exponents provide valuable insights into the nature of the singularities in the functions. This automated approach has the potential to greatly simplify the analysis of functions with multiple singularities, enabling researchers and practitioners to gain a deeper understanding of complex systems and phenomena.

The remaining sections of this paper are organized as follows: Several essential and fundamental aspects are introduced in Section 2, including Fourier data, discrete Fourier transform, and details for datasets. In Section 3, we present singularity detection models for single and multiple singularities based on neural network approach. Section 4 presents the experimental setup and results, demonstrating stable and reliable performances of our detection models through extensive validation. We summarize our findings, contributions in this paper, and discuss related future research in Section 5.

2. Preliminaries

In this section, we embark on elucidating the process of dataset preparation essential for training our neural network singularity detectors. A foundational aspect in the pursuit of effective training lies in the meticulous curation of a comprehensive dataset, comprising Fourier data extracted from various singular functions, alongside detailed information on the locations and exponents of the underlying singularities.

The Fourier data to train and test our detection models is generated by applying the Fast Fourier Transform (FFT) to a set of predetermined functions that exhibit specific locations and exponents of singularities. These functions are carefully selected to encompass a diverse range of singular behaviors, ensuring the robustness and generalizability of our approach. By transforming these functions into the frequency domain using FFT, we obtain discrete Fourier coefficients that capture the spectral characteristics associated with each singularity's exponent and location.

Subsection 2.1 provides an insightful review of the behavior of Fourier partial sums in approximating functions. Meanwhile, Subsection 2.2 delves into the Fourier data generation process using Discrete Fourier Transform. Lastly, Subsection 2.3 outlines the methodology employed in preparing our dataset for training.

2.1. Fourier data

For function $f(x)$ on the interval $[-L, L]$, the Fourier partial sum using the first $2N+1$ modes is

$$f_N(x) = \sum_{|\omega| \le N} \tilde{f}_\omega e^{i\omega \frac{\pi}{L} x}$$

with Fourier coefficients $\tilde{f}_\omega$ defined by

$$\tilde{f}_\omega = \frac{1}{2L} \int_{-L}^{L} f(x) e^{-i\omega \frac{\pi}{L} x} \, dx.$$

It is widely acknowledged that the Fourier partial sum provides an excellent approximation to a function with spectral accuracy when the function is periodic and analytic on the interval. However, when dealing with functions containing jumps or singularities, the Fourier series exhibits large oscillations near these discontinuities, a characteristic that remains unchanged even as the number of terms in the partial sum increases. Furthermore, in smooth regions away from the discontinuities, convergence is only of first order, leading to a lack of convergence in the maximum norm. This phenomenon is commonly referred to as the Gibbs phenomenon.

2.2. Discrete Fourier Transform (DFT)

Without loss of generosity, in our test, we assume the singular functions are defined over $\Omega = [0, 1]$, and we apply the fast Fourier transform (FFT) algorithm for computing the discrete Fourier transform (DFT) of the singular functions, which requires the function values at a set of equally spaced points within this interval.

Mathematically, we let $f(x)$ be a singular function defined on the interval $[0, 1]$. We determine a positive integer N and choose evenly-spaced sample points at which the function values are known, such as $x_n = n/N$ for $n = 0, 1, \ldots, N-1$. The DFT uses the function values $f(x_n)$ at the sample points x_n and produces N discrete Fourier coefficients. These coefficients represent the amplitudes and phases of the sinusoidal components that make up the function $f(x)$ over the interval $[0, 1]$.

To elaborate further, if we let $\mathtt{F}_\omega$ denote the discrete Fourier coefficients corresponding to the frequency ω, the coefficients are computed as follows:

$$\mathtt{F}_\omega := \frac{1}{N} \sum_{n=0}^{N-1} f(x_n) e^{-2\pi i \omega x_n}, \quad \omega = -\left\lfloor \frac{N}{2} \right\rfloor, \ldots, -1, 0, 1 \ldots, \left\lfloor \frac{N-1}{2} \right\rfloor, \tag{2.1}$$

where i is the imaginary unit, and $x_n = n/N$. For example, if $N = 5$, we obtain discrete Fourier coefficients $\mathtt{F}_{-2}, \mathtt{F}_{-1}, \mathtt{F}_0, \mathtt{F}_1, \mathtt{F}_2$. If $N = 6$, the coefficients are $\mathtt{F}_{-3}, \mathtt{F}_{-2}, \mathtt{F}_{-1}, \mathtt{F}_0, \mathtt{F}_1, \mathtt{F}_2$, with one more coefficient in order of decreasingly negative frequency.

In summary, the FFT algorithm takes as input the function values at equally spaced sample points within the interval $[0, 1]$, and it computes the discrete Fourier coefficients that characterize the frequency content of the function over this interval. These coefficients provide valuable insights into the sinusoidal components that comprise the function and are crucial for various applications in signal processing, spectral analysis, and numerical computation.

2.3. Datasets

We assume that a singular function $f : [0, 1] \to \mathbb{R}$ has a singularity whose location is $c_f \in (0, 1)$ and exponent is $s_f \in (0, 2)$; the singular function $f(x)$ is expressed by

$$f(x) = a(x) \left(|x - c_f|^{s_f} \right) + b(x), \tag{2.2}$$

where $a(x)$ is a smooth function, and $b(x)$ is a periodic smooth function. We randomly generate such singular functions with singularity locations and exponents and then use the one-dimensional equally-spaced N-point discrete Fourier transform (DFT) with the fast Fourier transform (FFT) algorithm. According to the discrete Poisson summation formula [10], the DFT data $\mathtt{F}_\omega$ is getting closer to the Fourier coefficient of frequency ω in a Fourier series as N is larger. Thus, we choose a sufficiently large $N = 1000$ for the FFT algorithm to obtain relatively accurate discrete Fourier coefficients. After getting the 1,000 DFT data $\mathtt{F}_\omega$ for $-500 \le \omega \le 499$, we choose dominant discrete Fourier coefficients by considering their frequencies because low-frequency and zero-frequency coefficients are typically larger than high-frequency ones. Therefore, we define a positive integer $\mathtt{M}$ ($\mathtt{M} \ll N = 1000$) to indicate how many discrete Fourier

coefficients are used as inputs, so the inputs and outputs corresponding to each singular function f are as follows:

$$\textbf{Inputs}: \left\{ \mathtt{F}_\omega : -\left\lfloor \frac{\mathtt{M}-1}{2} \right\rfloor \le \omega \le \left\lfloor \frac{\mathtt{M}}{2} \right\rfloor \right\} \quad \rightarrow \quad \textbf{Outputs}: c_f \text{ and } s_f. \tag{2.3}$$

For example, if we choose $\mathtt{M} = 5$, the inputs are determined as

$$\left\{ \mathtt{F}_\omega : -2 \le \omega \le 2 \right\} = \left\{ \mathtt{F}_{-2}, \mathtt{F}_{-1}, \mathtt{F}_0, \mathtt{F}_1, \mathtt{F}_2 \right\}.$$

If $\mathtt{M} = 6$, the inputs are determined as

$$\left\{ \mathtt{F}_\omega : -2 \le \omega \le 3 \right\} = \left\{ \mathtt{F}_{-2}, \mathtt{F}_{-1}, \mathtt{F}_0, \mathtt{F}_1, \mathtt{F}_2, \mathtt{F}_3 \right\}.$$

With a chosen integer $\mathtt{M} > 0$ in (2.3), we generate a training dataset to train our model, denoted as $\mathcal{F}_{\mathtt{train}}$, which collects singular functions' discrete Fourier coefficients $\mathtt{F}_\omega$ as inputs and singularity's location c_f and exponent s_f as outputs. For testing our trained model, we generate a test dataset, denoted as $\mathcal{F}_{\mathtt{test}}$, which contains singular functions independently of the training dataset. In addition, we define $|\cdot|$ as the number of samples in a dataset, so $|\mathcal{F}_{\mathtt{train}}|$ and $|\mathcal{F}_{\mathtt{test}}|$ denote the number of sample functions in the training and test dataset, respectively.

We also consider one-dimensional singular functions $g : [0,1] \to \mathbb{R}$ with two singularities whose locations are $c_g^1 \in (0, 0.5)$ and $c_g^2 \in (0.5, 1)$, and exponents are $s_g^1, s_g^2 \in (0, 2)$. The general form of such a function is

$$g(x) = a_1(x) \left(\left| x - c_g^1 \right|^{s_g^1} \right) + a_2(x) \left(\left| x - c_g^2 \right|^{s_g^2} \right) + b(x), \tag{2.4}$$

where $a_1(x)$ and $a_2(x)$ are smooth functions, and $b(x)$ is a periodic smooth function. Using the FFT algorithm and a selected integer $\mathtt{M}$, we generate datasets with two singularities in the following form,

$$\textbf{Inputs}: \left\{ \mathtt{G}_\omega : -\left\lfloor \frac{\mathtt{M}-1}{2} \right\rfloor \le \omega \le \left\lfloor \frac{\mathtt{M}}{2} \right\rfloor \right\} \quad \rightarrow \quad \textbf{Outputs}: c_g^1,\ c_g^2,\ s_g^1, \text{ and } s_g^2, \tag{2.5}$$

where $\mathtt{G}_\omega \in \mathbb{C}$ is a discrete Fourier coefficient for $g(x)$ with the frequency ω. In addition, the corresponding training and test datasets, denoted by $\mathcal{G}_{\mathtt{train}}$ and $\mathcal{G}_{\mathtt{test}}$, are generated similarly.

3. Deep neural network singularity detectors

We present our singularity detection models using deep neural network (DNN) structures. A DNN structure consists of the input layer, multiple hidden layers, and the output layer, while each layer includes numerous neurons. In this section, we propose the following singularity detectors for singular functions in (2.2) and (2.4):

3.1. *A DNN detector for single singularity*: We consider singular functions in (2.2) with different locations c_f, exponents s_f, and smooth functions $a(x)$ and $b(x)$. Using the discrete Fourier coefficients $\mathtt{F}_\omega$, we propose a DNN detection model to find the location and exponent of the singularity (see the input and output relation in (2.3)).

3.2. *DNN detectors for multiple singularities*: We try to detect multiple singularities of singular functions in (2.4), requiring high computational costs. With the discrete Fourier coefficients $\mathtt{G}_\omega$, we present two different DNN detection models for multiple singularities:

3.2.1. *A simple generalization of the single singularity detector*: This model is a straightforward generalization of the DNN detector for single singularity. We use the discrete Fourier coefficients $\mathtt{G}_\omega$ as inputs and increase the number of outputs so the detection model detects locations c_g^1, c_g^2, and exponents s_g^1, s_g^2 simultaneously.

3.2.2. *Splitting strategy for detecting multiple singularities*: The splitting strategy presents a separate location detector and another detector for exponents. We first detect locations c_g^1, c_g^2 using the location detector with the discrete Fourier coefficients $\mathtt{G}_\omega$ and then lead the detected locations to help the exponent detector find the corresponding exponents s_g^1, s_g^2.

3.1. A DNN detector for single singularity

In our detection model, we use the function's discrete Fourier coefficients chosen by frequencies,

$$\mathtt{F}_\omega \in \mathbb{C} \quad \text{for} \quad -\left\lfloor \frac{\mathtt{M}-1}{2} \right\rfloor \le \omega \le \left\lfloor \frac{\mathtt{M}}{2} \right\rfloor,$$

where $\mathtt{M}$ indicates the number of coefficients in light of Section 2.3. Hence, the neurons on the input layer are represented as the Fourier data

$$\mathcal{R}e\left(\mathtt{F}_\omega\right) \quad \text{and} \quad \mathcal{I}m\left(\mathtt{F}_\omega\right),$$

and the output layer consists of the singularity's location c_f and exponent s_f (see Figure 1). If we let L denote the number of layers in our DNN detector and n_ℓ denote the number of neurons on the ℓ-th layer for $1 \le \ell \le L$, it is clear that $n_1 = 2\mathtt{M}$ and $n_L = 2$. Moreover, the neurons on two adjacent layers are

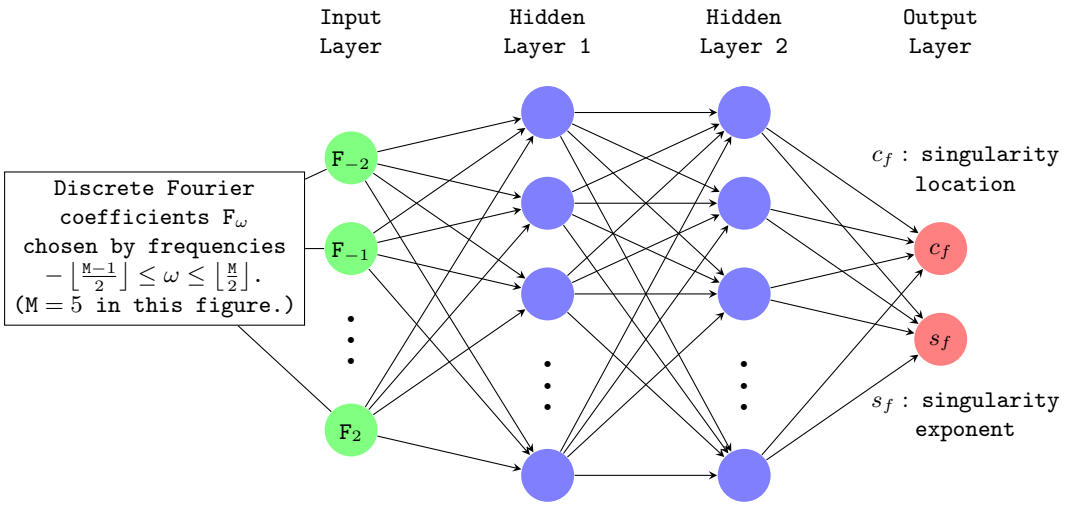

Figure 1: An example of the DNN detection model for a single singularity $\mathbf{z}_f \approx \mathbf{D}_1(\mathbf{X}_f)$.

connected by a linear transformation defined as $\mathbf{T}_\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_{\ell+1}}$ for $1 \le \ell \le L-1$,

$$\mathbf{T}_\ell(\mathbf{x}_\ell) = \mathbf{W}_\ell \mathbf{x}_\ell + \mathbf{b}_\ell, \quad \forall \mathbf{x}_\ell \in \mathbb{R}^{n_\ell},$$

where $\mathbf{W}_\ell \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$ is a parameter matrix, and $\mathbf{b}_\ell \in \mathbb{R}^{n_{\ell+1}}$ is a parameter vector. The DNN detector is a composition of such linear transformations and activation functions such as sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU). Therefore, for the DNN detector, a function $\mathbf{D}_1 : \mathbb{R}^{2\mathtt{M}} \to \mathbb{R}^2$ is defined as

$$\mathbf{D}_1 = \mathbf{T}_{L-1} \circ \underbrace{\boldsymbol{\sigma}_{L-2} \circ \mathbf{T}_{L-2}}_{\mathbb{L}_{L-2}} \circ \cdots \circ \underbrace{\boldsymbol{\sigma}_2 \circ \mathbf{T}_2}_{\mathbb{L}_2} \circ \underbrace{\boldsymbol{\sigma}_1 \circ \mathbf{T}_1}_{\mathbb{L}_1}, \tag{3.1}$$

where $\boldsymbol{\sigma}_\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_\ell}$ is a vector function whose each component is $\sigma : \mathbb{R} \to \mathbb{R}$ the nonlinear ReLU function, and $\mathbb{L}_\ell$ expresses sending signals from ℓ-th layer to $(\ell+1)$-th layer and processing them on the receiving layer. If we let

$$\mathbf{X}_f := \left\langle \mathcal{R}e\left(\mathtt{F}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}\right), \ldots, \mathcal{R}e\left(\mathtt{F}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\right), \mathcal{I}m\left(\mathtt{F}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}\right), \ldots, \mathcal{I}m\left(\mathtt{F}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\right) \right\rangle \in \mathbb{R}^{2\mathtt{M}} \tag{3.2}$$

and $\mathbf{z}_f = \langle c_f, s_f \rangle \in \mathbb{R}^2$, the DNN detector $\mathbf{D}_1(\cdot)$ approximates $\mathbf{z}_f$ using the input data $\mathbf{X}_f$, that is,

$$\mathbf{z}_f \approx \mathbf{D}_1(\mathbf{X}_f).$$

In conclusion, if we define $\mathcal{W}(\mathbf{D}_1) = \{\mathbf{W}_\ell,\ \mathbf{b}_\ell : 1 \le \ell \le L-1\}$, then we solve a least-squares problem to obtain a trained DNN detector $\mathbf{D}_1(\cdot)$,

$$\min_{\mathcal{W}(\mathbf{D}_1)} \frac{1}{|\mathcal{F}_{\mathtt{train}}|} \sum_{f \in \mathcal{F}_{\mathtt{train}}} \|\mathbf{z}_f - \mathbf{D}_1(\mathbf{X}_f)\|_2^2,$$

where $\mathcal{F}_{\mathtt{train}}$ is a training dataset introduced in Section 2.3.

3.2. DNN detectors for multiple singularities

We recall singular functions with two singularities introduced in (2.4),

$$g(x) = a_1(x)\left(\left|x - c_g^1\right|^{s_g^1}\right) + a_2(x)\left(\left|x - c_g^2\right|^{s_g^2}\right) + b(x).$$

Their discrete Fourier data $\mathtt{G}_\omega$ is a given data, and our models' goal is to detect the singularities' locations c_g^1, c_g^2 and exponents s_g^1, s_g^2 employing the Fourier data.

3.2.1. A simple generalization of the single singularity detector

The generalized DNN detection model uses the Fourier data,

$$\mathbf{X}_g := \left\langle \mathcal{R}e\left(\mathtt{G}_{-\lfloor \frac{\mathtt{M}-1}{2} \rfloor}\right), \ldots, \mathcal{R}e\left(\mathtt{G}_{\lfloor \frac{\mathtt{M}}{2} \rfloor}\right), \mathcal{I}m\left(\mathtt{G}_{-\lfloor \frac{\mathtt{M}-1}{2} \rfloor}\right), \ldots, \mathcal{I}m\left(\mathtt{G}_{\lfloor \frac{\mathtt{M}}{2} \rfloor}\right) \right\rangle \in \mathbb{R}^{2\mathtt{M}},$$

as inputs and $\mathbf{z}_g = \langle c_g^1, c_g^2, s_g^1, s_g^2 \rangle$ as outputs. With the DNN structure in (3.1) (or described in Figure 1), a trained DNN detection model for multiple singularities $\mathbf{D}_2(\cdot)$ can be obtained by solving a similar least-squares problem,

$$\min_{\mathcal{W}(\mathbf{D}_2)} \frac{1}{|\mathcal{G}_{\mathtt{train}}|} \sum_{g \in \mathcal{G}_{\mathtt{train}}} \|\mathbf{z}_g - \mathbf{D}_2(\mathbf{X}_g)\|_2^2,$$

where $\mathcal{W}(\mathbf{D}_2)$ is the collection of all parameter matrices and vectors involved in $\mathbf{D}_2$. Therefore, $\mathbf{D}_2(\mathbf{X}_g)$ approximates $\mathbf{z}_g$, which means that the generalized detection model $\mathbf{D}_2(\cdot)$ detects locations and exponents simultaneously.

However, this model that detects all locations and exponents together requires too many training sample functions and hidden layers to achieve a reliable error level (we will show related experimental results in Section 4.3). Hence, we propose a splitting strategy to overcome this computational difficulty.

3.2.2. Splitting strategy for detecting multiple singularities

From a practical viewpoint, detecting singularity locations seems easier than exponents. Thus, we first construct a DNN model to detect the locations using the Fourier data and then apply the detected (or approximate) locations to find the corresponding exponents. Let us define a location vector $\mathbf{c}_g := \langle c_g^1, c_g^2 \rangle$ and a function $\mathbf{D}_2^{\mathbf{c}} : \mathbb{R}^{2\mathtt{M}} \to \mathbb{R}^2$ based on the DNN structure (3.1). As shown in Figure 2, the inputs of $\mathbf{D}_2^{\mathbf{c}}$ are the Fourier data $\mathbf{X}_g$, while its outputs are the locations $\mathbf{c}_g$. Then, the trained DNN location detector $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ approximates $\mathbf{c}_g$ using the Fourier data $\mathbf{X}_g$,

$$\mathbf{c}_g \approx \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g),$$

and the corresponding least-squares problem is

$$\min_{\mathcal{W}(\mathbf{D}_2^{\mathbf{c}})} \frac{1}{|\mathcal{G}_{\mathtt{train}}|} \sum_{g \in \mathcal{G}_{\mathtt{train}}} \|\mathbf{c}_g - \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)\|_2^2,$$

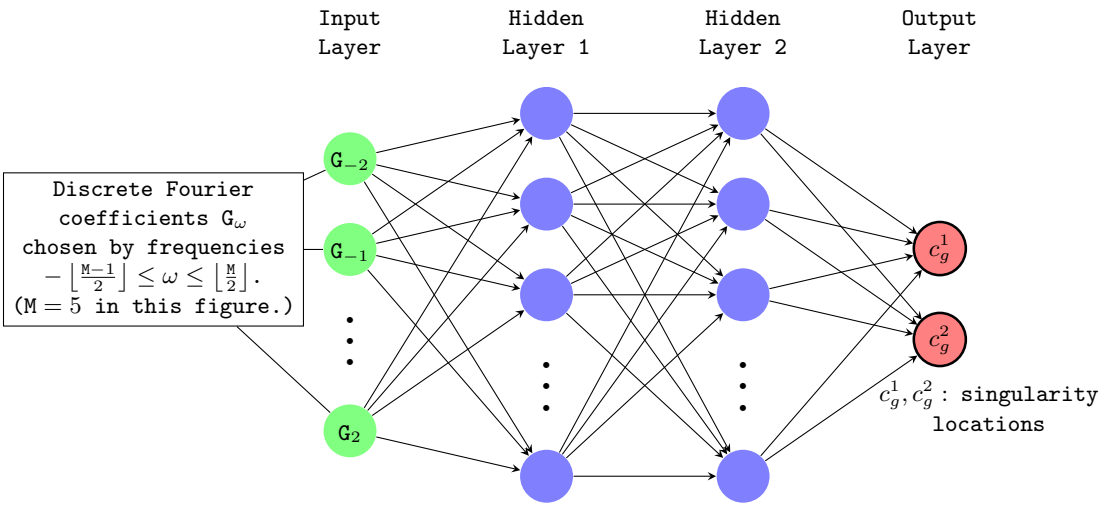

Figure 2: An example of the DNN location detection model for multiple singularities $\mathbf{c}_g \approx \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$.

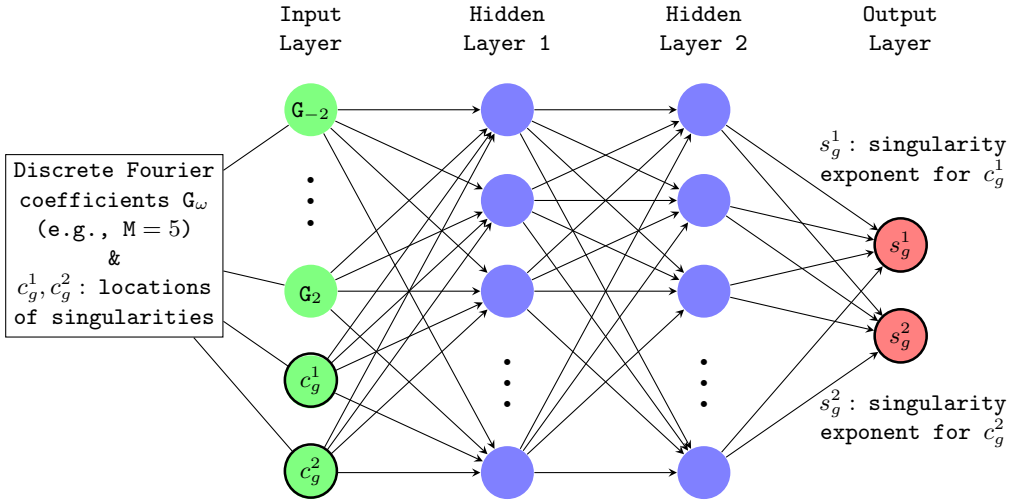

Figure 3: An example of the exponent detection model using the Fourier data and location information.

where $\mathcal{W}(\mathbf{D}_2^{\mathbf{c}})$ is the collection of all parameter matrices and vectors involved in $\mathbf{D}_2^{\mathbf{c}}$, and $\mathcal{G}_{\texttt{train}}$ is a training dataset presented in Section 2.3.

Furthermore, we construct a separate DNN detection model providing singularity exponents using the Fourier data $\mathbf{X}_g$ and location information $\mathbf{c}_g$ (or $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$) as input data. As shown in Figure 3, we define a function $\mathbf{D}_2^{\mathbf{s}} : \mathbb{R}^{2M+2} \to \mathbb{R}^2$ based on the structure (3.1) and apply it to approximate the exponent information $\mathbf{s}_g = \langle s_g^1, s_g^2 \rangle$. We propose two techniques to train the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$: *serial* and *parallel* learning processes (see Figure 4). The *serial* learning means we first train the location detector $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ using the Fourier data $\mathbf{X}_g$ to approximate $\mathbf{c}_g$. Then, we obtain detected locations $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$ and use them as training data for the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$. Therefore, we train the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ using the Fourier data $\mathbf{X}_g$ and detected locations $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$. In other words, the trained exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ minimizes the least-squares problem in the *serial* learning process,

$$\min_{\mathcal{W}(\mathbf{D}_2^{\mathbf{s}})} \sum_{g \in \mathcal{G}_{\texttt{train}}} \|\mathbf{s}_g - \mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))\|_2^2,$$

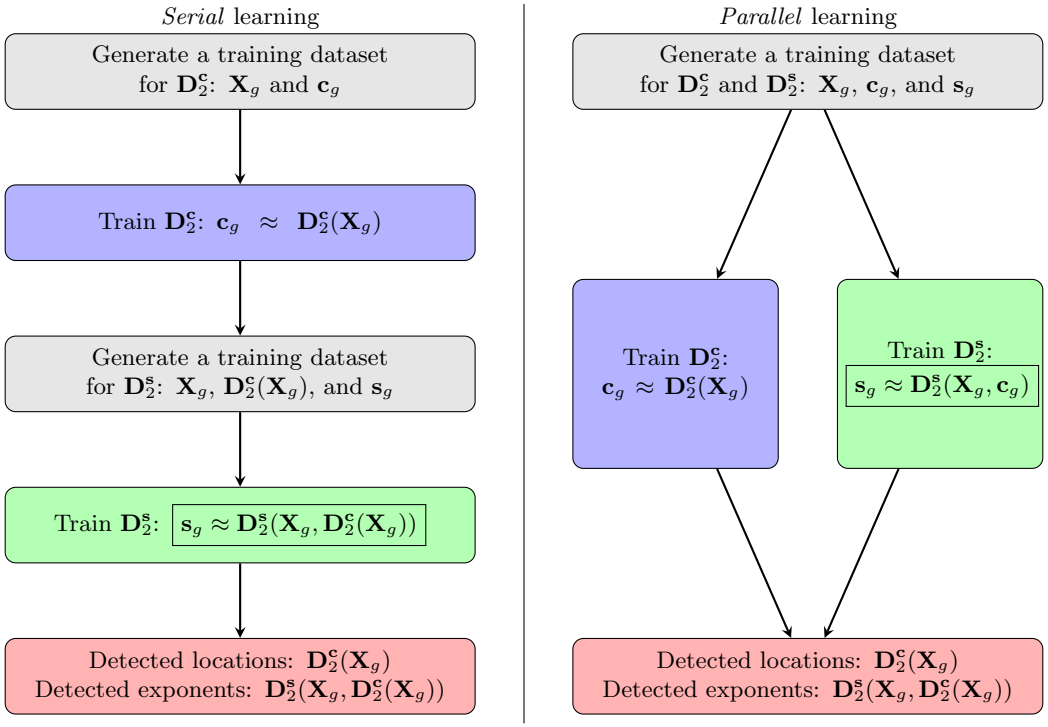

Figure 4: Flow charts of the *serial* and *parallel* learning processes in splitting strategy.

where $\mathcal{W}(\mathbf{D}_2^{\mathbf{s}})$ is the parameter set, and $\mathcal{G}_{\texttt{train}}$ is the training dataset.

On the other hand, as shown in Figure 4, the *parallel* learning means training the location and exponent detectors in parallel processing. While we train the location detector $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ using the Fourier data $\mathbf{X}_g$, we train the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ with the Fourier data $\mathbf{X}_g$ and the exact locations $\mathbf{c}_g$. Thus, the corresponding least-squares problem for the *parallel* learning is

$$\min_{\mathcal{W}(\mathbf{D}_2^{\mathbf{s}})} \sum_{g \in \mathcal{G}_{\texttt{train}}} \|\mathbf{s}_g - \mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{c}_g)\|_2^2.$$

We note that we use the exact locations $\mathbf{c}_g$ to train the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$, but we input the detected locations $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$ for the test dataset. For this reason, the *parallel* learning enhances efficiency in learning time but may result in larger errors of singularity exponents in test datasets (a related experimental result will be provided in Section 4.3). Finally, we emphasize that the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ approximates $\mathbf{s}_g$ only using the Fourier data $\mathbf{X}_g$ in both *serial* and *parallel* learning processes,

$$\mathbf{s}_g \approx \mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)).$$

4. Experimental Results

This section presents the experimental results of our DNN detection models for singular functions with single or multiple singularities. This section consists of three subsections:

4.1. Detection of single singularity: We construct and train a DNN detection model for a single singularity lead by $|x-c|^s$.

4.2. Detecting singularity from Fourier data with noise: We assume that the input Fourier data may contain some noise. Thus, we added artificial noise to the Fourier data and checked the performance of our DNN detector for a single singularity.

4.3. Detection of multiple singularities and a splitting strategy: We mainly consider singular functions with two singularities. To overcome the computational difficulty, we developed DNN detection models with a splitting strategy and compared them with the model detecting everything together.

4.1. Detection of single singularity

We begin by letting our DNN detection model detect a singularity of a singular function in a fractional power form $|x-c|^s$. We discuss experimental details for our detector's satisfactory performance, such as the number of discrete Fourier coefficients, the size of datasets, hidden layers, activation functions, and training methods. Based on the experimental results, we enhance our detector to capture singularities hidden by smooth functions (e.g., $a(x)(|x-c|^s)$ and $a(x)(|x-c|^s)+b(x)$, where $a(x)$ and $b(x)$ are smooth functions) by using more sample functions in datasets, discrete Fourier coefficients, and hidden layers.

Example 4.1 (*A singular function in a fractional power form*)**.** We consider one-dimensional singular functions, including single fractional power singularity with its location $c_f \in [0.1, 0.9]$ and exponent (or type) $s_f \in [0.1, 2.0)$:

$$f(x) = |x - c_f|^{s_f}, \qquad x \in [0, 1]. \tag{4.1}$$

We note that a singular function in Example 4.1 may include a jump discontinuity at the boundaries because the function is not periodic in $[0, 1]$. Such a jump discontinuity may confuse and complicate our detection models when the singularity location c_f is close to the boundaries. Hence, we consider the singularity location c_f on $[0.1, 0.9]$ to distinguish such an internal singularity from the jump discontinuity at the boundaries.

We generate datasets by randomly choosing such singular functions (with c_f and s_f) and using the one-dimensional equally-spaced N-point FFT algorithm with $N = 1000$ (see Section 2.3 for details). As explained in Section 3.1, we construct and train a DNN singularity detection model with

- Inputs: discrete Fourier coefficients chosen by frequencies,

$$\mathtt{F}_\omega \in \mathbb{C} \quad \text{for } -\left\lfloor \frac{\mathtt{M}-1}{2} \right\rfloor \le \omega \le \left\lfloor \frac{\mathtt{M}}{2} \right\rfloor,$$

  where $\mathtt{M}$ means the number of coefficients. The coefficients form a vector $\mathbf{X}_f \in \mathbb{R}^{2\mathtt{M}}$ introduced in (3.2);

- Outputs: singularity's location c_f and exponent s_f;

- Hidden layers and number of neurons: two hidden layers with 128×64;

- Activation functions: ReLU.

To maximize the accuracy of our trained detection model, we use the optimization algorithm *Adam* with adaptive learning rates and batch processing. We also hold 10% of the singular functions in a training dataset as a validation set and check the mean squared error (MSE) on the validation set to change learning rates and stop learning iterations. If the MSE on the validation set does not decrease with the tolerance 10^{-6} in 200 iterations, the learning rate is reduced by a factor of 10, and the learning process stops when the learning rate reaches less than 10^{-6}. Figure 5 shows how the MSEs (or losses) decrease during the learning process with different learning rates.

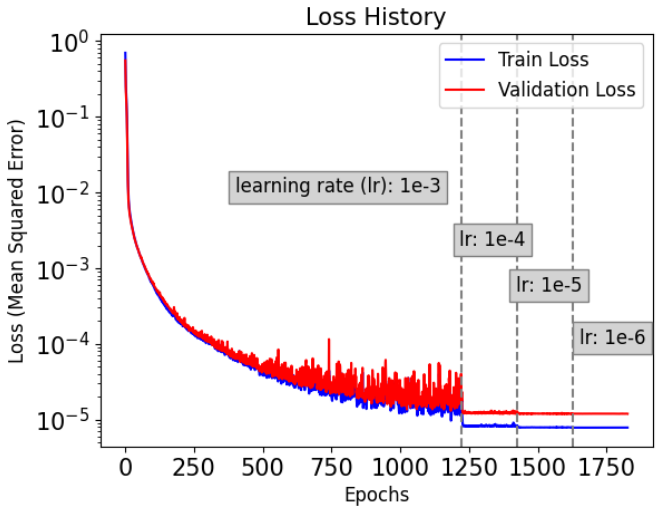

Figure 5: An example of the MSEs (or losses) on the training and validation sets during the learning process in Example 4.1 with $\mathtt{M} = 5$.

We first test how we generate a dataset, addressing the number of discrete Fourier coefficients ($\mathtt{M}$) and the number of singular functions in a training dataset ($|\mathcal{F}_{\mathtt{train}}|$). Figure 6 shows the final losses on the training and validation sets with different dataset sizes after finishing the learning process. We change the number of Fourier coefficients from 1 to 10 of 3,000 singular functions in a training dataset, i.e., $|\mathcal{F}_{\mathtt{train}}| = 3000$ and $1 \leq \mathtt{M} \leq 10$. As shown in Figure 6, the training and validation losses with five coefficients are less than 2×10^{-5}. When the number of coefficients increases, the losses fluctuate in a similar range. Therefore, we discover that our detection model with five discrete Fourier coefficients ($\mathtt{M} = 5$ and the chosen frequencies are $\omega = -2, -1, 0, 1, 2$) reliably detects singularities in Example 4.1.

Moreover, we change the number of singular functions in a training dataset from 200 to 5,000 while using their five discrete Fourier coefficients, that is, $200 \leq |\mathcal{F}_{\mathtt{train}}| \leq 5000$ and $\mathtt{M} = 5$. The training and validation losses decrease as more singular functions are in a training dataset from 200 to 3,000, but they fluctuate when a dataset includes more than 3,000 functions. Therefore, for the satisfactory performance of our model, we set a training dataset including 3,000 singular functions and their five discrete Fourier coefficients ($|\mathcal{F}_{\mathtt{train}}| = 3000$, and $\mathtt{M} = 5$).

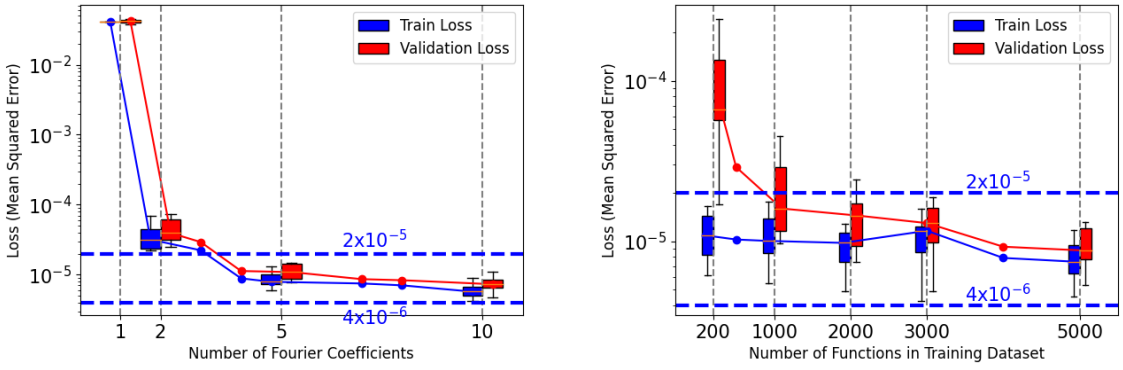

Figure 6: Training and validation losses with different numbers of discrete Fourier coefficients chosen by frequencies (left) and those of singular functions in a training dataset (right) in Example 4.1.

To check the performance of our trained detection model $\mathbf{D}_1(\cdot)$, we randomly generate 200 singular functions in the fractional form in Example 4.1 for a test dataset and compare their exact singularity

locations and exponents $\mathbf{z}_f = \langle c_f, s_f \rangle$ with the ones detected by our detector $\mathbf{D}_1(\mathbf{X}_f)$. In Figure 7, a blue dot indicates each singular function's detected location and exponent compared to the exact, and red lines mean the detected location and exponent are the same as the exact. Thus, if our detector correctly detects singularities' locations and exponents, blue dots are supposed to be densely clustered on the red lines. As shown in Figure 7, our trained detection model detects the singularity locations and exponents in the test dataset very well while providing small MSEs on the test dataset.

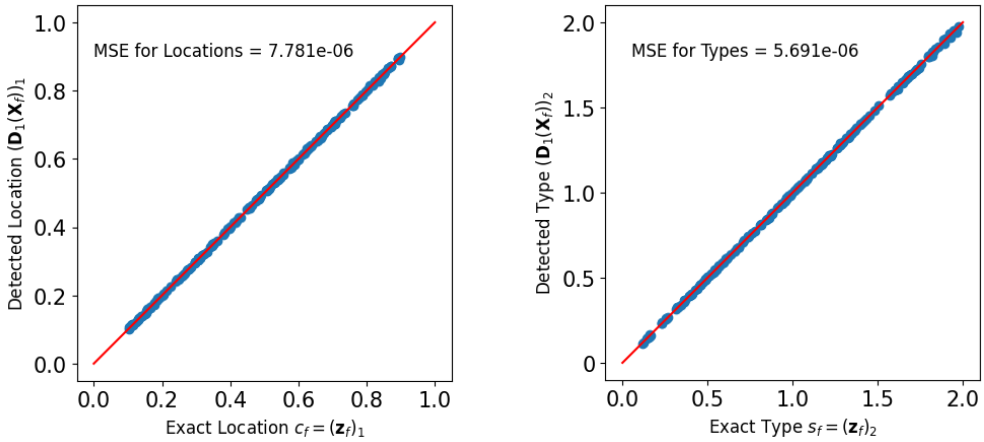

Figure 7: Comparisons of detected locations and exponents (or types) with the exact ones of 200 test functions in Example 4.1.

Example 4.2 (*Singular function multiplied by a smooth function*)**.** We consider a more complicated case, a singular function multiplied by a smooth function with its location $c_f \in [0.1, 0.9]$ and exponent $s_f \in [0.1, 2.0)$:

$$f(x) = a(x)\left(|x - c_f|^{s_f}\right), \tag{4.2}$$

where $a(x)$ is a smooth function. We choose different smooth functions, such as $a(x) = 1$, $\cos(2\pi x)$, $\sin(2\pi x)$, $\arctan(\pi x)$, and e^x. The location c_f is chosen on $[0.1, 0.9]$ due to a possible jump discontinuity at the boundaries. The choice $a(x) = 1$ implies that a dataset will contain the fractional power form in Example 4.1.

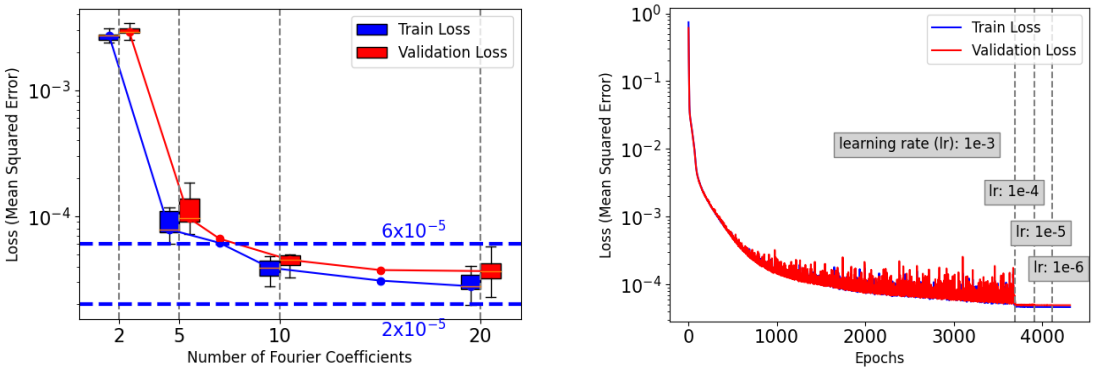

Figure 8: Losses on the training and validation sets depending on the number of Fourier coefficients (left) and during the learning process with different learning rates and $\mathtt{M} = 11$ (right) in Example 4.2.

In Example 4.2, it may be challenging to confirm the singularity's location and exponent visually (see Figure 9 for an example). We apply the same DNN structure $\mathbf{D}_1(\cdot)$ as in Section 3.1, having the same

inputs (Fourier data $\mathbf{X}_f$), outputs (location c_f & exponent s_f), hidden layers (128×64), and activation functions (ReLU). We check how many discrete Fourier coefficients are enough for the satisfactory performance of our trained model. Based on the previous result of the number of singular functions, we generate 15,000 singular functions combined with smooth functions to set up a training dataset. Figure 8 shows the losses on the training and validation sets depending on the number of Fourier coefficients and during the learning process with different learning rates. The losses decrease from two to ten coefficients and fluctuate after ten, implying that our detection model with ten or more Fourier coefficients ($\mathtt{M} \geq 10$) performs well in Example 4.2.

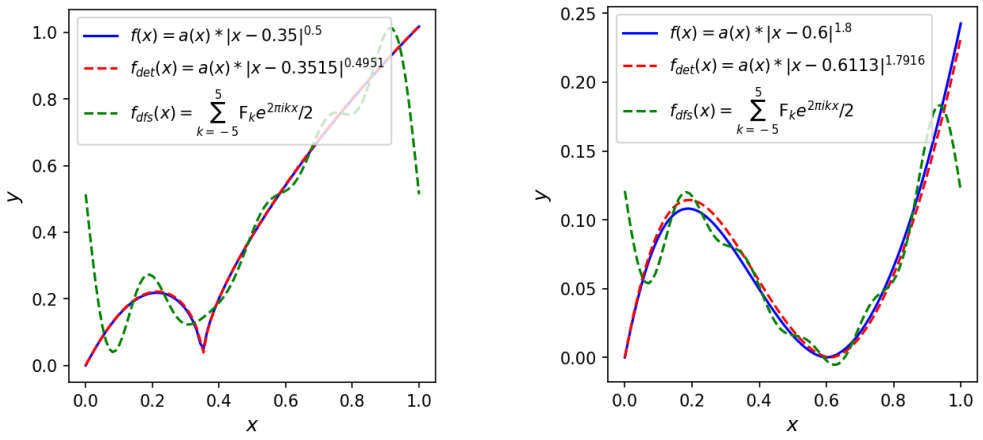

Figure 9: Comparison of example functions, detected functions, and their discrete Fourier series in Example 4.2 ($a(x) = \arctan(\pi x)$).

With fixed $\mathtt{M} = 11$, Figure 9 compares example singular functions, detected functions from $\mathbf{D}_1(\mathbf{X}_f)$, and their discrete Fourier series. When the example function has a cusp, we can see that capturing the singularity using the discrete Fourier series is tricky due to an oscillatory behavior (the Gibbs phenomenon) caused by a jump discontinuity at the boundaries. However, a function detected by our trained detector comparatively shows the accurate location and exponent of cusp. Moreover, as displayed in Figure 9, our trained detector effectively detects the hidden singularity with $c_f = 0.6$ and $s_f = 1.8$ that may not be visually detected.

Example 4.3 (*Singular function hidden by multiplication and addition*)**.** We treat a combination of a singular function and smooth functions with its location $c_f \in [0.1, 0.9]$ and exponent $s_f \in [0.1, 2.0)$:

$$f(x) = a(x)\,(|x - c_f|^{s_f}) + b(x), \tag{4.3}$$

where $a(x)$ is a smooth function, and $b(x)$ is a periodic smooth function such as $\sin(2\pi x)+1$ and $\cos(2\pi x)+1$.

This combination's singularity is more elusive than a singular function in Example 4.1 or Example 4.2. For example, Figure 10 displays a singular function $|x-0.6|^{1.6}$ in Example 4.1 and different combinations with smooth functions $a(x) = \arctan(\pi x)$ and $b(x) = \cos(2\pi x) + 1$. As shown in Figure 10, the singularity of a function in Example 4.3 looks completely hidden. Furthermore, comparing the singular functions' Fourier data, we can see that adding a periodic smooth function to a singular function causes the amplification of discrete Fourier coefficients of frequencies between -1 and 1 due to the linearity of the Fourier transform. This alteration in the Fourier coefficients makes the other coefficients relatively negligible, so detecting such hidden singularities may be more challenging.

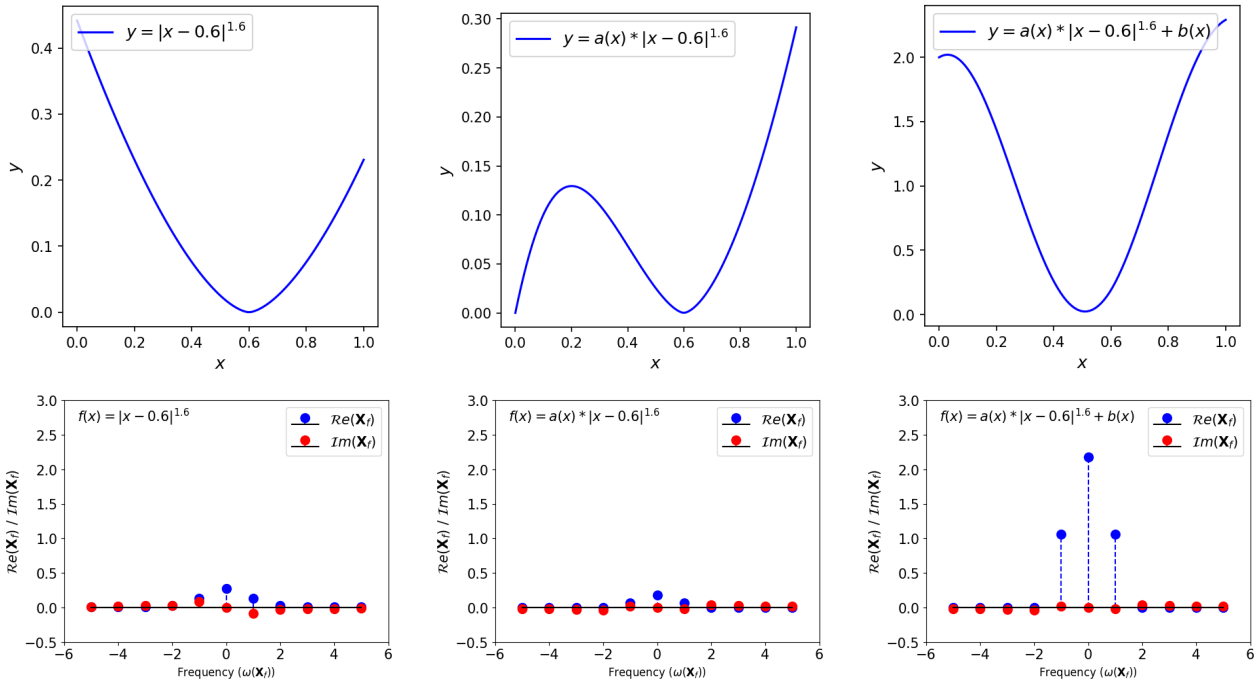

Figure 10: A singular function $|x - 0.6|^{1.6}$ and different combinations with smooth functions $a(x) = \arctan(\pi x)$ and $b(x) = \cos(2\pi x) + 1$ (top row), and their discrete Fourier coefficients with $\mathtt{M} = 11$ (bottom row).

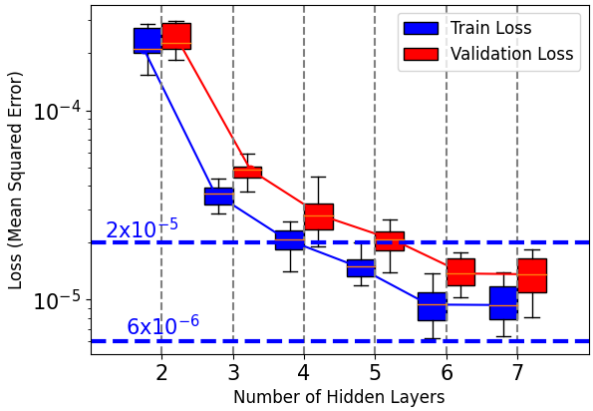

Figure 11: Training and validation losses with different numbers of hidden layers in Example 4.3.

In Example 4.3, our detection model $\mathbf{D}_1(\cdot)$ follows the DNN structure in Section 3.1, having the same inputs (Fourier data $\mathbf{X}_f$ with $\mathtt{M} = 11$), outputs (location c_f & exponent s_f), and activation functions (ReLU). We also generate 30,000 singular functions for a training dataset based on the result of Example 4.1. However, due to the challenge in detecting singularities in Example 4.3, our detection model requires more hidden layers for stable and satisfactory performance, even though we generate enough functions and discrete Fourier coefficients for training datasets. Figure 11 shows that the losses on the training and validation decrease as more hidden layers are used, and the losses fluctuate after six layers are applied. Based on this experimental result, we decide to apply six hidden layers with the number of neurons $256 \times 128 \times 128 \times 128 \times 128 \times 64$ for satisfactory performance of our trained model $\mathbf{D}_1(\cdot)$ for the complex case in Example 4.3.

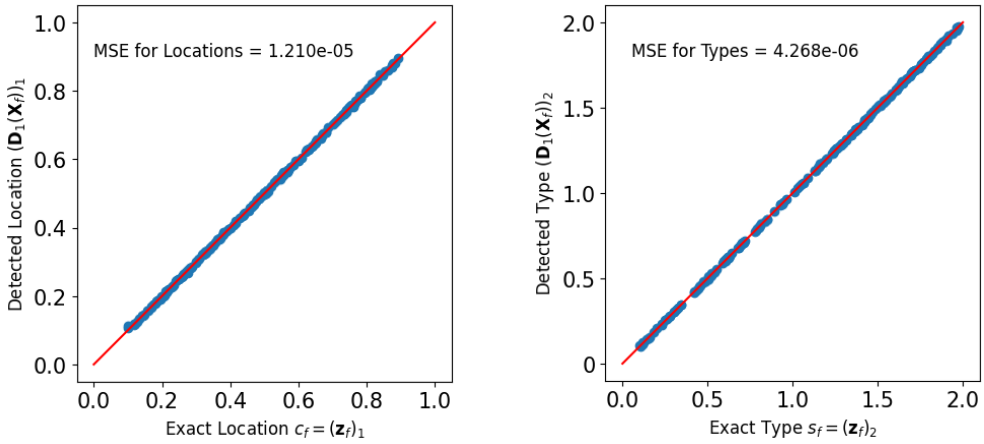

Figure 12: Comparisons of detected locations and exponents (or types) with the exact ones of 200 test functions in Example 4.3.

Figure 12 compares the detected locations and exponents $\mathbf{D}_1(\mathbf{X}_f)$ with the exact ones $\mathbf{z}_f$ in a test dataset, including 200 randomly chosen functions in Example 4.3. As shown in Figure 12, the MSEs for locations and exponents are relatively small, and the blue dots are very close to the red lines. These results imply that our DNN detection model $\mathbf{D}_1(\cdot)$ effectively and accurately detects the singularity's location and exponent in Example 4.3.

4.2. Detecting singularity from Fourier data with noise

This subsection deals with cases closer to application issues. While we consider singular functions in Example 4.3,

$$f(x) = a(x)\,(|x - c_f|^{s_f}) + b(x),$$

we agree its Fourier data $\mathbf{X}_f \in \mathbb{R}^{2\mathtt{M}}$ may contain noises, that is,

$$\begin{aligned}\mathbf{X}_f + \boldsymbol{\epsilon} := \Big\langle &\mathcal{R}e\left(\mathtt{F}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}\right) + \mathtt{e}^{\mathcal{R}}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}, \ldots, \mathcal{R}e\left(\mathtt{F}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\right) + \mathtt{e}^{\mathcal{R}}_{\lfloor\frac{\mathtt{M}}{2}\rfloor},\\ &\mathcal{I}m\left(\mathtt{F}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}\right) + \mathtt{e}^{\mathcal{I}}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}, \ldots, \mathcal{I}m\left(\mathtt{F}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\right) + \mathtt{e}^{\mathcal{I}}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\Big\rangle.\end{aligned}$$

In our experiments, the noise $\boldsymbol{\epsilon} \in \mathbb{R}^{2\mathtt{M}}$ is a standard normal random vector scaled by the median of the components of $\mathbf{X}_f$. More precisely, if we define

$$\begin{aligned}M_{\mathcal{R}} &:= \text{median}\left(\left|\mathcal{R}e\left(\mathtt{F}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}\right)\right|, \ldots, \left|\mathcal{R}e\left(\mathtt{F}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\right)\right|\right),\\ M_{\mathcal{I}} &:= \text{median}\left(\left|\mathcal{I}m\left(\mathtt{F}_{-\lfloor\frac{\mathtt{M}-1}{2}\rfloor}\right)\right|, \ldots, \left|\mathcal{I}m\left(\mathtt{F}_{\lfloor\frac{\mathtt{M}}{2}\rfloor}\right)\right|\right),\end{aligned}$$

then $\mathtt{e}^{\mathcal{R}}_{\omega} \sim \mathcal{N}(0, (M_{\mathcal{R}})^2)$ and $\mathtt{e}^{\mathcal{I}}_{\omega} \sim \mathcal{N}(0, (M_{\mathcal{I}})^2)$ independent and identically distributed (i.i.d.) for each ω, respectively.

We test the DNN detection model $\mathbf{D}_1(\cdot)$ constructed and trained in Example 4.3. Thus, when $\mathbf{z}_f = \langle c_f, s_f\rangle$ denotes the exact location and exponent in a test dataset, $\mathbf{D}_1(\mathbf{X}_f)$ approximates $\mathbf{z}_f$, and $\mathbf{D}_1(\mathbf{X}_f + \boldsymbol{\epsilon})$ means the detected location and exponent from the Fourier data with noise. Figure 13 compares the detected locations and exponents $\mathbf{D}_1(\mathbf{X}_f + \boldsymbol{\epsilon})$ with the exact ones $\mathbf{z}_f$ in a test dataset with 200 singular functions. Even though noises disrupt our model's ability to detect the singularity's location and exponent, our detector still provides small MSEs and detects relatively accurate locations and exponents.

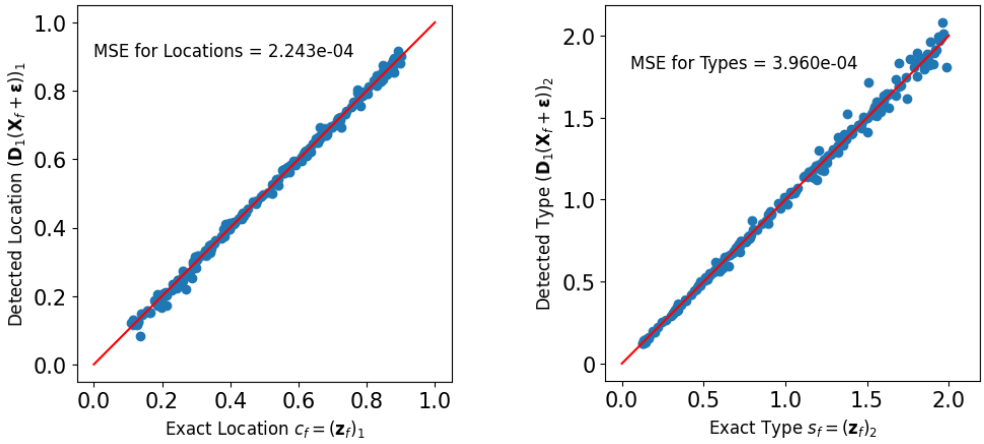

Figure 13: Comparisons of detected locations and exponents (or types) with the exact ones of 200 test functions in Example 4.3 using Fourier data with noise.

In addition, we pick several locations and exponents as examples from the test dataset: **(a)** $\mathbf{z}_f = \langle 0.3, 0.5\rangle$ and **(b)** $\mathbf{z}_f = \langle 0.8, 1.7\rangle$. Table 1 displays discrete Fourier coefficients, the Fourier data with noise, and detected locations and exponents. The noise levels (relative differences) are also shown in bold in Table 1. It is clear that noises alter the Fourier coefficients; the coefficients with the noises are no longer symmetric, and the coefficients that were originally zero are no longer zero. Such noises depend on the magnitude of the coefficients, and their noise levels vary in frequencies. Dominant coefficients have relatively lower noise levels, while the others have higher ones. Despite many coefficients exhibiting noise levels exceeding 1 percent, our detector detects the locations and exponents with an accuracy of around 1 percent or less. Therefore, we can conclude that our detector effectively identifies the singularity's location and exponent from Fourier data with noise.

(a) Exact location and exponent: $\mathbf{z}_f = \langle 0.3, 0.5\rangle$ $(f(x) = \arctan(\pi x)|x-0.3|^{0.5} + \cos(2\pi x) + 1)$

ω ($\mathtt{M} = 11$)	-5	-4	-3	-2	-1	0	1	2	3	4	5
$\mathcal{R}e(\mathtt{F}_\omega)$	4.141e-3	-3.275e-3	-1.384e-2	9.157e-3	5.313e-1	1.479e+0	5.313e-1	9.157e-3	-1.384e-2	-3.275e-3	4.141e-3
$\mathcal{R}e(\mathtt{F}_\omega) + \mathtt{e}_\omega^{\mathcal{R}}$	3.606e-3	-3.222e-3	-1.275e-2	8.222e-3	5.314e-1	1.478e+0	5.314e-1	9.569e-3	-1.380e-2	-2.774e-3	4.003e-3
Noise level	**12.92%**	**1.59%**	**7.89%**	**10.20%**	**0.01%**	**0.04%**	**0.02%**	**4.50%**	**0.33%**	**1.53%**	**3.35%**
$\mathcal{I}m(\mathtt{F}_\omega)$	-3.289e-2	-4.928e-2	-4.958e-2	-6.221e-2	-1.943e-1	0	1.943e-1	6.221e-2	4.958e-2	4.928e-2	3.289e-2
$\mathcal{I}m(\mathtt{F}_\omega) + \mathtt{e}_\omega^{\mathcal{I}}$	-2.914e-2	-4.947e-2	-4.967e-2	-6.299e-2	-1.939e-1	6.212e-3	1.933e-1	5.768e-2	5.599e-2	4.893e-2	3.153e-2
Noise level	**11.42%**	**0.38%**	**0.19%**	**1.26%**	**0.18%**	**NaN**	**0.53%**	**7.27%**	**12.93%**	**0.70%**	**4.14%**

$\Downarrow$

Detected ones without noise: $\mathbf{D}_1(\mathbf{X}_f) = \langle 0.2999, 0.4964\rangle$ with relative errors $\langle \mathbf{0.03}\%, \mathbf{0.72}\%\rangle$
Detected ones with noise: $\mathbf{D}_1(\mathbf{X}_f + \boldsymbol{\epsilon}) = \langle 0.2965, 0.4966\rangle$ with relative errors $\langle \mathbf{1.17}\%, \mathbf{0.68}\%\rangle$

(b) Exact location and exponent: $\mathbf{z}_f = \langle 0.8, 1.7\rangle$ $(f(x) = e^x|x-0.8|^{1.7} + \sin(2\pi x) + 1)$

ω ($\mathtt{M} = 11$)	-5	-4	-3	-2	-1	0	1	2	3	4	5
$\mathcal{R}e(\mathtt{F}_\omega)$	2.436e-3	3.986e-3	8.025e-3	1.743e-2	3.946e-2	1.268e+0	3.946e-2	1.743e-2	8.025e-3	3.986e-3	2.436e-3
$\mathcal{R}e(\mathtt{F}_\omega) + \mathtt{e}_\omega^{\mathcal{R}}$	2.541e-3	4.146e-3	9.210e-3	1.789e-2	4.080e-2	1.267e+0	3.963e-2	1.756e-2	7.331e-3	5.648e-3	4.372e-3
Noise level	**4.29%**	**4.00%**	**14.77%**	**2.60%**	**3.41%**	**0.09%**	**0.43%**	**0.71%**	**8.64%**	**41.67%**	**79.47%**
$\mathcal{I}m(\mathtt{F}_\omega)$	1.645e-2	2.030e-2	2.781e-2	4.752e-2	6.311e-1	0	-6.311e-1	-4.752e-2	-2.781e-2	-2.030e-2	-1.645e-2
$\mathcal{I}m(\mathtt{F}_\omega) + \mathtt{e}_\omega^{\mathcal{I}}$	1.606e-2	1.927e-2	2.670e-2	4.433e-2	6.340e-1	2.313e-3	-6.355e-1	-4.363e-2	-2.787e-2	-2.062e-2	-1.451e-2
Noise level	**2.39%**	**5.05%**	**3.99%**	**6.72%**	**0.46%**	**NaN**	**0.70%**	**8.19%**	**0.21%**	**1.61%**	**11.76%**

$\Downarrow$

Detected ones without noise: $\mathbf{D}_1(\mathbf{X}_f) = \langle 0.8002, 1.7029\rangle$ with relative errors $\langle \mathbf{0.02}\%, \mathbf{0.17}\%\rangle$
Detected ones with noise: $\mathbf{D}_1(\mathbf{X}_f + \boldsymbol{\epsilon}) = \langle 0.7959, 1.7099\rangle$ with relative errors $\langle \mathbf{0.51}\%, \mathbf{0.58}\%\rangle$

Table 1: Comparisons of detected locations and exponents using Fourier data with and without noise, displaying Fourier data, the Fourier data with noise, and noise levels of two test functions in Example 4.3.

4.3. Detection of multiple singularities and a splitting strategy:

We propose DNN detection models to find singularities of singular functions in the form

$$g(x) = a_1(x)\left(\left|x - c_g^1\right|^{s_g^1}\right) + a_2(x)\left(\left|x - c_g^2\right|^{s_g^2}\right) + b(x),$$

where

- $c_g^1 \in (0.1, 0.5)$; $c_g^2 \in (0.5, 0.9)$; $s_g^1, s_g^2 \in [0.1, 2.0)$;
- $a_1(x), a_2(x)$ are smooth functions chosen in Example 4.2;
- $b(x)$ is a periodic smooth function chosen in Example 4.3.

In light of Section 2.3, we generate datasets by randomly choosing singular functions, including output data $\mathbf{z}_g = \langle c_g^1, c_g^2, s_g^1, s_g^2 \rangle$, and applying the FFT algorithm,

$$\mathbf{X}_g := \left\langle \mathcal{R}e\left(\mathtt{G}_{-\lfloor \frac{\mathtt{M}-1}{2} \rfloor}\right), \ldots, \mathcal{R}e\left(\mathtt{G}_{\lfloor \frac{\mathtt{M}}{2} \rfloor}\right), \mathcal{I}m\left(\mathtt{G}_{-\lfloor \frac{\mathtt{M}-1}{2} \rfloor}\right), \ldots, \mathcal{I}m\left(\mathtt{G}_{\lfloor \frac{\mathtt{M}}{2} \rfloor}\right) \right\rangle \in \mathbb{R}^{2\mathtt{M}},$$

Based on Section 3.2, we test a generalized DNN detection model $\mathbf{D}_2(\cdot)$ approximating locations and exponents simultaneously, $\mathbf{z}_g \approx \mathbf{D}_2(\mathbf{X}_g)$. We also present the splitting strategy for detecting multiple singularities, including two separate detectors $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ and $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ for $\mathbf{c}_g = \langle c_g^1, c_g^2 \rangle$ and $\mathbf{s}_g = \langle s_g^1, s_g^2 \rangle$, respectively. We summarize the proposed models as follows (see also Figure 1-4):

- `Detecting together`: $\mathbf{D}_2(\cdot)$ is trained to minimize $\|\mathbf{z}_g - \mathbf{D}_2(\mathbf{X}_g)\|_2$. $\mathbf{D}_2(\mathbf{X}_g)$ is an approximation of $\mathbf{z}_g$.
- `Splitting strategy` (*serial learning*): $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ is trained to minimize $\|\mathbf{c}_g - \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)\|_2$. Then, with obtained $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$, $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ is trained to minimize $\|\mathbf{s}_g - \mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))\|_2$. For testing, $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$ is an approximation of $\mathbf{c}_g$, and $\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))$ is an approximation of $\mathbf{s}_g$.
- `Splitting strategy` (*parallel learning*): $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ is trained to minimize $\|\mathbf{s}_g - \mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{c}_g)\|_2$, while $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ is trained to minimize $\|\mathbf{c}_g - \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)\|_2$. For testing, $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$ is an approximation of $\mathbf{c}_g$, and $\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))$ is an approximation of $\mathbf{s}_g$.

Table 2 compares the details of the model $\mathbf{D}_2(\cdot)$ `detecting together` and `splitting strategy` (*serial* and *parallel*) using $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ and $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$. For a fair comparison, we apply the same number of neurons (1,148 neurons) to all the models and use the same training set and algorithm. In Table 2, the

	`Detecting together`	`Splitting` (*serial*)		`Splitting` (*parallel*)	
	$\mathbf{D}_2(\mathbf{X}_g)$	$\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$	$\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))$	$\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$	$\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{c}_g)$
Hidden layers	8 layers	4 layers	4 layers	4 layers	4 layers
Training time	129.9 min	68.1 min	86.4 min	75.3 min (*parallel*)	
Training loss	1.296e-3 ($\mathbf{c}_g$ & $\mathbf{s}_g$)	1.776e-4 ($\mathbf{c}_g$)	4.179e-3 ($\mathbf{s}_g$)	1.776e-4 ($\mathbf{c}_g$)	2.526e-3 ($\mathbf{s}_g$)
Validation loss	4.484e-3 ($\mathbf{c}_g$ & $\mathbf{s}_g$)	3.740e-4 ($\mathbf{c}_g$)	8.030e-3 ($\mathbf{s}_g$)	3.740e-4 ($\mathbf{c}_g$)	6.201e-3 ($\mathbf{s}_g$)
Training set	40,000 singular functions & 15 discrete Fourier coefficients				

Table 2: Details of `detecting together` using $\mathbf{D}_2(\cdot)$ and `splitting strategy` using $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ and $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ for multiple singularities.

training and validation losses of trained $\mathbf{D}_2(\cdot)$ include the errors for locations and exponents. At the same time, the losses of trained $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ (or $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$) mean the errors only for locations (or exponents), so the losses of `detecting together` are between those of location and exponent detectors in the `splitting strategy`. Nevertheless, the trained location detectors $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ in the `splitting strategy` have much

less training and validation losses, which leads us to expect more accurate detected locations from the `splitting strategy`. Moreover, as shown in Table 2, the *parallel* process shows efficiency in training the models compared to the other training ways. The *parallel* process also produces fewer losses than the *serial* process because the *parallel* uses the exact locations as training data (while the *serial* process uses the approximate locations).

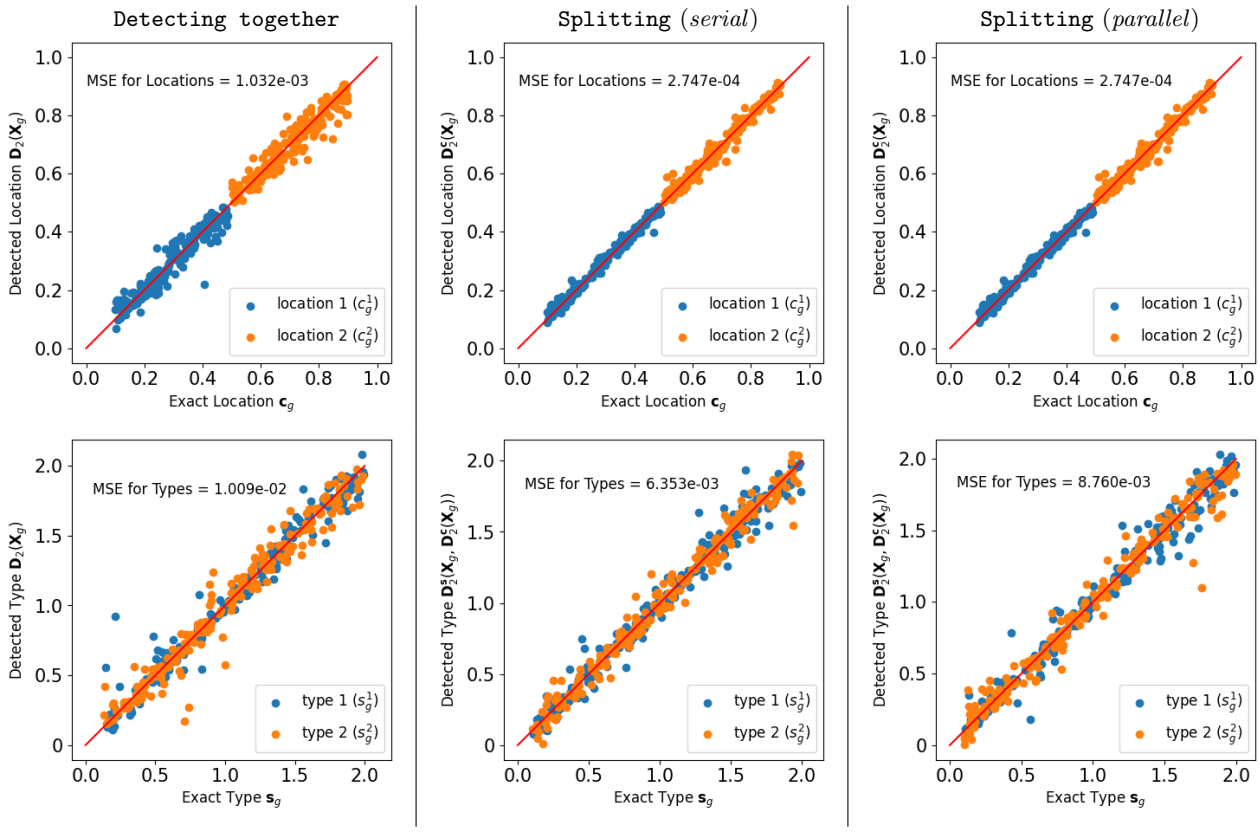

Figure 14: Comparisons of detected locations and exponents (or types) with the exact ones of 200 test functions with multiple singularities.

Figure 14 shows detected locations and exponents in a test dataset of 200 singular functions chosen independently of the training dataset, comparing the model `detecting together` $\mathbf{D}_2(\cdot)$ and `splitting strategy` $\mathbf{D}_2^{\mathbf{c}}(\cdot) \to \mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ with *serial* and *parallel* processes. As shown in Figure 14, the detected locations using the `splitting strategy` are closer to the red line than detecting locations and exponents together, which confirms that the `splitting strategy` provides more accurate locations. Also, such an accuracy comparison can be verified by the MSEs for locations.

While testing the `splitting strategy` with *serial* and *parallel* processes, we first detect locations $\mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g)$ and then use the detected locations to find exponents with $\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))$. For the *parallel* process, the results of the exponent detector $\mathbf{D}_2^{\mathbf{s}}(\cdot,\cdot)$ are more accurate if the exact locations $\mathbf{c}_g$ are applied, that is, $\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{c}_g)$ is more accurate than $\mathbf{D}_2^{\mathbf{s}}(\mathbf{X}_g, \mathbf{D}_2^{\mathbf{c}}(\mathbf{X}_g))$. However, it is natural that we do not know the exact locations, but we get approximate locations from our location detector $\mathbf{D}_2^{\mathbf{c}}(\cdot)$ using the Fourier data $\mathbf{X}_g$. Therefore, contrary to the training and validation losses in Table 2, the *serial* process gives fewer test MSEs on exponents than the *parallel* process (see Figure 14) because the exponent detector in the *serial* process is trained with detected locations and more suitable for them. Nonetheless, the exponent detector through the *parallel* process still performs better than the model `detecting together`, providing efficiency in training time. Therefore, we conclude that the detection

models with the `splitting strategy` detect multiple singularities more effectively and accurately, and the *parallel learning* process makes the training procedure in the `splitting strategy` efficient.

5. Conclusion

Our work presents a novel application of neural networks for the automated detection and characterization of singularities in functions. Through extensive numerical experiments, we validate the accuracy and robustness of our approach in identifying both single and multiple singularities, encompassing their locations and exponents. By harnessing the capabilities of machine learning, we offer a valuable tool for researchers and practitioners across diverse fields, facilitating efficient analysis and interpretation of functions exhibiting multiple singularities.

However, detecting multiple singularities still presents a significant challenge in our research. While our current methodology focuses on functions with two singularities for testing purposes, addressing functions with an arbitrary number of singularities requires the development of a novel neural network (NN) detector. A potential approach involves refining the splitting strategy, where the first NN detects potential singularity locations across the entire domain. Subsequently, a second NN zooms in on specific intervals, such as $[a, b]$, surrounding potential singularities to pinpoint their accurate locations and types. However, implementing this strategy poses challenges, particularly in incorporating additional information into the input data. Specifically, for accurate detection, we need to augment the input discrete Fourier transform (DFT) data, originally generated from the interval $[0, 1]$, with crucial points such as $\{0, 1, a, b\}$. This augmentation ensures that our NN detector receives comprehensive input data necessary for precise singularity detection across various intervals.

In future work, we will explore more efficient methods for utilizing training datasets within the splitting strategy, specifically focusing on how the location and exponent detectors can better communicate and share detected information to enhance each minimization process. This would involve transitioning from the current one-directional splitting approach, $\mathbf{D}_2^{\mathbf{c}}(\cdot) \to \mathbf{D}_2^{\mathbf{s}}(\cdot, \cdot)$, to an alternating update pattern, $\mathbf{D}_2^{\mathbf{c}}(\cdot, \cdot) \to \mathbf{D}_2^{\mathbf{s}}(\cdot, \cdot) \to \mathbf{D}_2^{\mathbf{c}}(\cdot, \cdot) \to \cdots$. We anticipate that this iterative pattern will help the detectors more effectively identify the correlation between locations and exponents. Additionally, we plan to extend our analysis to functions with different types of singularities and apply our methodology to real-world datasets, thereby enhancing the applicability and robustness of our approach.

References

[1] S Abarbanel, E Tadmor, S Gottlieb, J S Hesthaven, David Gottlieb, and Anne Gelb. The resolution of the Gibbs phenomenon for "spliced" functions in one and two dimensions. *Computers & Mathematics with Applications*, 33(11):35–58, June 1997.

[2] Rick Archibald, Kewei Chen, Anne Gelb, and Rosemary Renaut. Improving tissue segmentation of human brain MRI through preprocessing by the Gegenbauer reconstruction method. *NeuroImage*, 20(1):489–502, September 2003.

[3] Rick Archibald and Anne Gelb. A method to reduce the gibbs ringing artifact in mri scans while keeping tissue boundary integrity. *IEEE Transactions on Medical Imaging*, 21(4):305–319, 2002.

[4] Rick Archibald and Anne Gelb. Reducing the effects of noise in image reconstruction. *Journal of Scientific Computing*, 17(1-4):167–180, 2002.

[5] Rick Archibald, Anne Gelb, Sigal Gottlieb, and Jennifer Ryan. One-sided post-processing for the discontinuous galerkin method using eno type stencil choosing and the local edge detection method. *Journal of Scientific Computing*, 28(2-3):167–190, 2006.

[6] Rick Archibald, Anne Gelb, Rishu Saxena, and Dongbin Xiu. Discontinuity detection in multivariate space for stochastic simulations. *Journal of Computational Physics*, 228(7):2676–2689, 2009.

[7] Rick Archibald, Anne Gelb, and Jungho Yoon. Polynomial fitting for edge detection in irregularly sampled signals and images. *SIAM journal on numerical analysis*, 43(1):259–279, 2005.

[8] Rick Archibald, Jiuxiang Hu, Anne Gelb, and Gerald Farin. Improving the accuracy of volumetric segmentation using pre-processing boundary detection and image reconstruction. *IEEE Transactions on Image Processing*, 13(4):459–466, 2004.

[9] Emmanuel Bacry, Jean-François Muzy, and Alain Arneodo. Singularity spectrum of fractal signals from wavelet analysis: Exact results. *Journal of statistical physics*, 70(3):635–674, 1993.

[10] William L Briggs and Van Emden Henson. *The DFT: an owner's manual for the discrete Fourier transform*. SIAM, 1995.

[11] Zheng Chen and Chi-Wang Shu. Recovering exponential accuracy from collocation point values of smooth functions with end-point singularities . *Journal of Computational and Applied Mathematics*, 265:83 – 95, 2014.

[12] Zheng Chen and Chi-Wang Shu. Recovering Exponential Accuracy in Fourier Spectral Methods Involving Piecewise Smooth Functions with Unbounded Derivative Singularities. *Journal of Scientific Computing*, 65(3):1145–1165, 2015.

[13] Rene Y Choi, Aaron S Coyner, Jayashree Kalpathy-Cramer, Michael F Chiang, and J Peter Campbell. Introduction to machine learning, neural networks, and deep learning. *Translational vision science & technology*, 9(2):14–14, 2020.

[14] Doug Cochran, Anne Gelb, and Yang Wang. Edge detection from truncated fourier data using spectral mollifiers. *Advances in computational mathematics*, 38(4):737–762, 2013.

[15] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[16] Dennis Denker and Anne Gelb. Edge detection of piecewise smooth functions from undersampled fourier data using variance signatures. *SIAM Journal on Scientific Computing*, 39(2):A559–A592, 2017.

[17] Anne Gelb. Parameter Optimization and Reduction of Round Off Error for the Gegenbauer Reconstruction Method. *Journal of Scientific Computing*, 20(3):433–459, 2004.

[18] Anne Gelb and Dennis Cates. Detection of edges in spectral data iii—refinement of the concentration method. *Journal of Scientific Computing*, 36(1):1–43, 2008.

[19] Anne Gelb and Taylor Hines. Detection of edges from nonuniform fourier data. *Journal of Fourier Analysis and Applications*, 17(6):1152–1179, 2011.

[20] Anne Gelb, Guohui Song, Aditya Viswanathan, and Yang Wang. Edge detection from two-dimensional fourier data using gaussian mollifiers. *MICHIGAN STATE UNIV EAST LANSING*, 2016.

[21] Anne Gelb and Eitan Tadmor. Detection of edges in spectral data. *Applied and computational harmonic analysis*, 7(1):101–135, 1999.

[22] Anne Gelb and Eitan Tadmor. Detection of edges in spectral data. II. Nonlinear enhancement. *SIAM Journal on Numerical Analysis*, 38(4):1389–1408, 2000.

[23] David Gottlieb and Sigal Gottlieb. Spectral methods for compressible reactive flows. *Comptes Rendus Mécanique*, 333(1):3–16, 2005.

[24] David Gottlieb and Chi-Wang Shu. Resolution properties of the Fourier method for discontinuous waves. *Computer Methods in Applied Mechanics and Engineering*, 116(1-4):27–37, 1994.

[25] David Gottlieb and Chi-Wang Shu. On the Gibbs Phenomenon IV: Recovering Exponential Accuracy in a Subinterval from a Gegenbauer Partial Sum of a Piecewise Analytic Function. *Mathematics of Computation*, 64(211):1081, July 1995.

[26] David Gottlieb and Chi-Wang Shu. On the Gibbs phenomenon V: recovering exponential accuracy from collocation point values of a piecewise analytic function. *Numerische Mathematik*, 71(4):511–526, 1995.

[27] David Gottlieb and Chi-Wang Shu. On the Gibbs Phenomenon III: Recovering Exponential Accuracy in a Sub-Interval From a Spectral Partial Sum of a Pecewise Analytic Function. *SIAM Journal on Numerical Analysis*, 33(1):280–290, 1996.

[28] David Gottlieb, Chi-Wang Shu, Alex Solomonoff, and Hervé Vandeven. On the Gibbs phenomenon I: recovering exponential accuracy from the Fourier partial sum of a nonperiodic analytic function. *Journal of Computational and Applied Mathematics*, 43(1):81–98, November 1992.

[29] Sigal Gottlieb, David Gottlieb, and Chi-Wang Shu. Recovering high-order accuracy in weno computations of steady-state hyperbolic systems. *Journal of Scientific Computing*, 28(2-3):307–318, 2006.

[30] Gottlieb, David and Shu, Chi-Wang. On the Gibbs Phenomenon and Its Resolution. *SIAM review*, 39(4):644–668, August 2006.

[31] Ankur Handa, Michael Bloesch, Viorica Pătrăucean, Simon Stent, John McCormac, and Andrew Davison. gvnn: Neural network library for geometric computer vision. In *European Conference on Computer Vision*, pages 67–82. Springer, 2016.

[32] Felix J Herrmann. Singularity characterization by monoscale analysis: Application to seismic imaging. *Applied and Computational Harmonic Analysis*, 11(1):64–88, 2001.

[33] Tai-Chiu Hsung, DP-K Lun, and Wan-Chi Siu. Denoising by singularity detection. *IEEE Transactions on Signal Processing*, 47(11):3139–3144, 1999.

[34] Jae-Hun Jung, Sigal Gottlieb, Saeja Oh Kim, Chris L Bresten, and Daniel Higgs. Recovery of high order accuracy in radial basis function approximations of discontinuous problems. *Journal of Scientific Computing*, 45(1-3):359–381, 2010.

[35] Qin Li, Jianfeng Lu, and Weiran Sun. A convergent method for linear half-space kinetic equations. *ESAIM: M2AN*, 51(5):1583–1615, 2017.

[36] Uzu Lim, Harald Oberhauser, and Vidit Nanda. Hades: Fast singularity detection with local measure comparison. *arXiv preprint arXiv:2311.04171*, 2023.

[37] Yaron Lipman and David Levin. Approximating piecewise-smooth functions. *IMA journal of numerical analysis*, 30(4):1159–1183, 2009.

[38] Stephane Mallat and Wen Liang Hwang. Singularity detection and processing with wavelets. *IEEE transactions on information theory*, 38(2):617–643, 1992.

[39] Adam Martinez, Anne Gelb, and Alexander Gutierrez. Edge detection from non-uniform fourier data using the convolutional gridding algorithm. *Journal of Scientific Computing*, 61(3):490–512, 2014.

[40] Yohhan Pao. Adaptive pattern recognition and neural networks. 1989.

[41] Alex Petersen, Anne Gelb, and Randall Eubank. Hypothesis testing for fourier based edge detection methods. *Journal of Scientific Computing*, 51(3):608–630, 2012.

[42] Chi-Wang Shu and Peter S Wong. A note on the accuracy of spectral method applied to nonlinear conservation laws. *Journal of scientific computing*, 10(3):357–369, 1995.

[43] Wolfgang Stefan, Adityavikram Viswanathan, Anne Gelb, and Rosemary Renaut. Sparsity enforcing edge detection method for blurred and noisy fourier data. *Journal of Scientific Computing*, 50(3):536–556, 2012.

[44] Zbigniew R Struzik. Determining local singularity strengths and their spectra with the wavelet transform. *Fractals*, 8(02):163–179, 2000.

[45] Adityavikram Viswanathan, Douglas Cochran, Anne Gelb, and Dennis M Cates. Detection of signal discontinuities from noisy fourier data. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1705–1708. IEEE, 2008.

[46] Adityavikram Viswanathan, Anne Gelb, and Douglas Cochran. Iterative design of concentration factors for jump detection. *Journal of Scientific Computing*, 51(3):631–649, 2012.

[47] Julius Von Rohrscheidt and Bastian Rieck. Topological singularity detection at multiple scales. In *International Conference on Machine Learning*, pages 35175–35197. PMLR, 2023.

[48] Hao Wang, Xilin Yang, Zeqi Liu, Jing Pan, Yuan Meng, Zijian Shi, Zhensong Wan, Hengkang Zhang, Yijie Shen, Xing Fu, et al. Deep-learning-based recognition of multi-singularity structured light. *Nanophotonics*, 11(4):779–786, 2022.

[49] Junmei Zhong, Ruola Ning, and David Conover. Image denoising based on multiscale singularity detection for cone beam ct breast imaging. *IEEE transactions on medical imaging*, 23(6):696–703, 2004.