Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2023-0078

Vol. **35**, No. 5, pp. 1194-1228
May 2024

A Causality-DeepONet for Causal Responses of Linear Dynamical Systems

Lizuo Liu[1], Kamaljyoti Nath[2] and Wei Cai[1,*]

[1] *Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA.*
[2] *Division of Applied Mathematics, Brown University, Providence, RI 02912, USA.*

Received 23 March 2023; Accepted (in revised version) 26 December 2023

Abstract. In this paper, we propose a DeepONet structure with causality to represent causal linear operators between Banach spaces of time-dependent signals. The theorem of universal approximations to nonlinear operators proposed in [5] is extended to operators with causalities, and the proposed Causality-DeepONet implements the physical causality in its framework. The proposed Causality-DeepONet considers causality (the state of the system at the current time is not affected by that of the future, but only by its current state and past history) and uses a convolution-type weight in its design. To demonstrate its effectiveness in handling the causal response of a physical system, the Causality-DeepONet is applied to learn the operator representing the response of a building due to earthquake ground accelerations. Extensive numerical tests and comparisons with some existing variants of DeepONet are carried out, and the Causality-DeepONet clearly shows its unique capability to learn the retarded dynamic responses of the seismic response operator with good accuracy.

AMS subject classifications: 65R20, 65Z05, 78M25

Key words: Neural network, universal approximation theory of nonlinear operator, DeepONet, Causality-DeepONet.

1 Introduction

Computing operators between physical quantities defined in function spaces have many applications in forward and inverse problems in scientific and engineering computations. For example, in wave scattering in inhomogeneous or random media, the mapping between the media physical properties, which can be modelled as a random field, and the

*Corresponding author. *Email addresses:* lizuol@smu.edu (L. Liu), kamaljyoti_nath@brown.edu (K. Nath), cai@smu.edu (W. Cai)

wave field is a nonlinear operator, which represents some of the most challenging computational tasks in medical imaging, geophysical and seismic problems. A specific example comes from earthquake safety studies of buildings and structures, the response of structures to seismic ground accelerations gives rise to a causal operator between spaces of highly oscillatory temporal signals.

Structural dynamic analysis has always been one of the crucial problems in the civil engineering field. Traditionally, researchers in this field analyzing structural dynamic response focus on constructing proper mathematical models like ordinary or partial differential equations and utilizing grid-based numerical methods to solve them. The finite element method [50] is one of the popular methods considered for the solutions along with an appropriate time integration scheme like Newmark's-Beta method [6, 36]. Alternatively, system identification-based methods, as an attempt to construct a surrogate model by mapping the input signals to the output responses directly, have shown their superior capability in accelerating the computations. A comprehensive review of this approach was provided in [42, 19]. Meanwhile, recently learning time sequential response operator between input and output signals [23] has been studied using recurrent neural network (RNN) [11,24], long short-term memory neural network (LSTM) [17], WaveNet [37], the one-step ResNet approximation [40] and the multi-step recurrent ResNet approximation [40]. The RNN and its variant LSTM are ubiquitous network structures for predicting time series in financial engineering, machine translation, and sentiment analysis and so on in the natural language processing field. In particular, LSTM has been shown to have the potential to predict building responses excited by seismic ground accelerations [20, 46]. The one-step ResNet approximation and the multi-step recurrent ResNet approximation, provide the approximation to the integral form of the dynamical system and have been demonstrated effective equation recovery for linear and nonlinear dynamical systems [12, 40].

Deep neural networks (DNNs), as one of the most intuitive frameworks for model reductions with its superior ability to approximate general high dimensional functions [7], have been considered recently in learning mappings whose closed forms are not known. So far, DNNs have shown much promise in solving problems from scientific and engineering computing, including initial and boundary value problems of ODEs and PDEs [4, 10, 15, 18, 30, 41, 47, 28]. Soon after universal approximation theorems to functions by neural networks was proposed [7], Chen & Chen [5] proved that there also exists a framework that could give universal approximations to nonlinear operators between Banach spaces. Based on this theory, the DeepONet [32] was constructed for learning operators where trunk net functions are used as basis and the branch net functions as mappings from the input functions to some hidden manifolds. The DeepONet replaced the one-hidden layer networks in the original proposal in Chen & Chen's paper [5] by two deep neural networks, which has been shown to have the potential to break the curse of dimensionality from the input space. In the meantime, other approach for learning operators based on a graph kernel network [26] for PDEs has also been proposed. The nonlin-

ear operator is decomposed by composing nonlinear activation functions with a class of integral operators with a trainable kernel. The Fourier neural operator has been proposed [25] by replacing the integral operator with the Fourier transform and a trainable mask in frequency domain. Both the graph kernel neural network and the Fourier neural operator show the capability to approximate specific operators with very good accuracy and efficiency.

In this paper, we will study the DeepONet specifically for time-dependent operators from a physical system such as those encountered in a building seismic wave response problems. The Causality-DeepONet will be proposed to ensure the causality of the retarded Green's function of the underlying differential equation between the input seismic ground accelerations and the output building responses. In addition to the causality consideration, the time homogeneity of a dynamic system will also be used in the design of the neural network by encoding the convolutional nature of the retarded Green's function in the choice of the network weights. The proposed DeepONet with built-in causality allows us to learn, accurately and with minimum requirement of training data, the mapping between the ground accelerations and the corresponding displacements of the building at the roof level excited by the seismic ground accelerations.

It could be noted that the term causality is also used in fields such as causal inference [31, 34] and causal interpretability of neural network [35], or applying the neural network to solve problem like causal reasoning [14]. Those works are referring to the logical cause-effect relationships between data. However, in our setting we focus on the physical concept of temporal causality, i.e., the state of the system at the current time is not affected by that of the future, but only by its current state and past history. For this reason, we name our framework Causality-DeepONet. In a study of crack propagation [13], past histories of tensile energies were provided as input to the branch net of a variational energy-based DeepONet with convolution to predict the growth of fracture under quasi-static loading conditions. This approach is similar to implicit time discretization of nonlinear time evolution equation commonly used for time dependent Navier-Stokes equations where the state(s) of the system at previous time step(s) act as a forcing term for the linear system to be solved for the state of the current time.

The rest of the paper is organized as follows. In Section 2, we state the problem considered in the present study. In Section 3, we give a short review of the universal approximation theory of nonlinear operator by neural networks, and its recent development DeepONet. Further, we provide a review of a multi-scale neural network introduced to handle high frequency functions and the POD-DeepONet for efficient basis functions in the trunk net of DeepONet. Fourier neural operator is also reviewed in this section. In Section 4, we propose two extensions of the DeepONet, one is the multi-scale DeepONet, the other is the Causality-DeepONet. In Section 5, a comparison of the results with all the mentioned frameworks will be carried out. The conclusion and future works are included in the Section 6.

2 Problem statement: Calculation of building response due to seismic load

The problem under study is the prediction of the dynamic response of a multi-story building due to seismic loading. The equation of motion, after a finite element type discretization, for the building due to ground motions during an earthquake could be written as a dynamic system of differential equations [6],

$$\boldsymbol{M}\ddot{\boldsymbol{x}}+\boldsymbol{C}\dot{\boldsymbol{x}}+\boldsymbol{K}\boldsymbol{x}=\boldsymbol{f}(t), \tag{2.1}$$

where $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$ are the mass, damping and stiffness matrices of the system from the finite element discretization. $\boldsymbol{f}(t)$ the applied force, for our case, is due to ground motions during an earthquake and could be written as

$$\boldsymbol{f}(t)=\boldsymbol{M}\boldsymbol{\iota}\ddot{u}_g, \tag{2.2}$$

where $\ddot{u}_g$ is the ground acceleration due to the earthquake and $\boldsymbol{\iota}$ is the influence vector. The interested reader may refer [6] for more details on formulation and solution methods.

Ground accelerations due to earthquakes are recorded at different recording stations. In the present study, we consider ground accelerations due to earthquakes for different earthquakes recorded at different stations and taken from the database of the Pacific Earthquake Engineering Research Center (`https://ngawest2.berkeley.edu/`) (`https://peer.berkeley.edu/`) [†]. One of the typical records of ground accelerations is shown in Fig. 1. The earthquake record at different stations may be recorded at different sampling rates (different δt). Earthquake records recorded at finer $\delta t<0.02$ sec are filtered using a Butterworth filter with frequency (0.1-24.9) Hz then re-sampled to $\delta t=0.02$ sec and after that amplified to match with original PGA level. Figures before and after processing for the mentioned earthquake record in Fig. 1 are shown in Appendix C.1. The building is considered at rest initially with the initial condition

$$\begin{cases}\boldsymbol{x}(0)=\boldsymbol{0},\\ \dot{\boldsymbol{x}}(0)=\boldsymbol{0}.\end{cases} \tag{2.3}$$

Our objective is to evaluate an operator operating on the ground acceleration and predict the response of the building. Thus, it is a mapping from ground acceleration to the response of the top floor of the building.

$$\mathcal{R}:\ddot{u}_g(t)\longrightarrow x_1(t). \tag{2.4}$$

Detailed numerical study is carried out for a six-storied reinforced cement concrete (RCC) building. A 3D model of the building is generated in openseespy [48]. Apart from the

[†]The earthquake ground acceleration considered are taken from the Pacific Earthquake Engineering Research Center (PEER: `https://ngawest2.berkeley.edu/`, `https://peer.berkeley.edu/`)

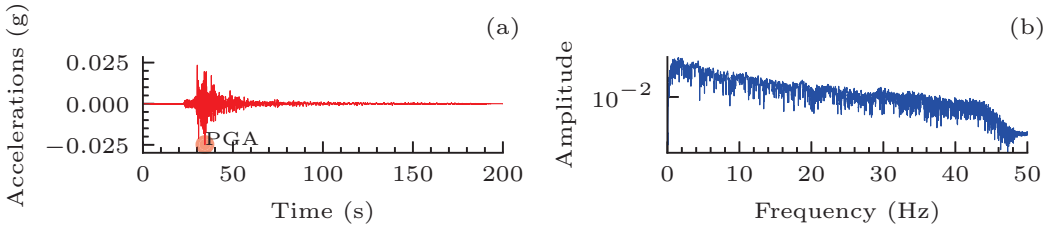

Figure 1: **A typical ground acceleration of due to earthquake:** The ground acceleration is due to 14383980 earthquake recorded at station North Hollywood, 2008 (a) time history of the acceleration (b) frequency spectrum. The absolute maximum acceleration is indicated which is also known as Peak Ground Acceleration (PGA) of the earthquake.

dead load (beam, column, slab, wall etc.), the live load is also considered on each floor and roof. Generally, in seismic analysis of building, a full dead load and a reduced live load are considered. In the present study, the lumped mass is calculated based on a full dead load and 50% of the live load at floor level and 25% at roof level. The damping matrix of the building is calculated using model damping for 5% model damping for all the modes. The matrix size of $\boldsymbol{M},\boldsymbol{C},\boldsymbol{K}$ is 504×504 since there is 504 degree of freedom of the building model after applying support (boundary conditions) at the foundation level. Ground acceleration is applied in the major direction. The records obtained from PEER contain 3 different directions. The vertical ground accelerations are not considered. The other two horizontal ground accelerations are considered one at a time and applied only in the major direction.

In the case of a classical damped system [6], the displacement $\boldsymbol{x}(t)$ may be decomposed as the superposition of the modal contributions of undamped system $\boldsymbol{x}(t)=\boldsymbol{\Phi q}$, $\boldsymbol{\Phi}=[\phi_1,\phi_2,\cdots,\phi_n]$, $\boldsymbol{q}=[q_1,q_2,\cdots,q_n]^T$. q_l and ϕ_l are the modal coordinates and the corresponding modes, respectively, for natural frequency ω_l. The responses (displacement) $\boldsymbol{x}(t)$ of the system for ground accelerations $\ddot{u}_g(t)$ may be represented as

$$\boldsymbol{x}(t)=\int_0^t \ddot{u}_g(\tau)\boldsymbol{h}(t-\tau)d\tau, \tag{2.5}$$

where $\boldsymbol{h}(t)=-\sum_{\ell=1}^{n}\phi_\ell\frac{\Gamma_\ell}{\omega_{\ell D}}e^{-\xi_\ell\omega_\ell t}\sin\omega_{\ell D}t$ is the unit impulse response, also known as, the Green's function and fundamental solution, $\Gamma_\ell=\frac{\phi_\ell^T\boldsymbol{M\iota}}{\phi_\ell^T\boldsymbol{M}\phi_\ell}$ and $\boldsymbol{\iota}$ is the influence vector, $\omega_{\ell D}=\omega_\ell\sqrt{1-\xi_\ell^2}$ with ξ_ℓ as the damping ratio.

In the case of a non-classical damped system [6], the responses (displacement) $\boldsymbol{x}(t)$ of the system due to ground accelerations $\ddot{u}_g(t)$ may be represented as

$$\boldsymbol{x}(t)=-\sum_{\ell=1}^{n}\left[\tilde{\gamma}_\ell^\delta\omega_\ell^{\mathcal{N}}D_\ell(t)+\alpha_\ell^\delta\dot{D}_\ell(t)\right], \tag{2.6}$$

where

$$D_\ell(t)=\int_0^t \ddot{u}_g(\tau)H_\ell(t-\tau)d\tau,\quad \dot{D}_\ell(t)=\int_0^t \ddot{u}_g(\tau)\dot{H}_\ell(t-\tau)d\tau. \tag{2.7}$$

$H_\ell(t)=-\frac{1}{\omega_{\ell D}^{\mathcal{N}}}e^{-\xi_\ell^{\mathcal{N}}\omega_\ell^{\mathcal{N}}t}\sin\omega_{\ell D}^{\mathcal{N}}t$ is the Green's function of the non-classical damped system, with $\omega_{\ell D}^{\mathcal{N}}=\omega_\ell^{\mathcal{N}}\sqrt{1-(\xi_\ell^{\mathcal{N}})^2}$ and $\omega_\ell^{\mathcal{N}}=|\lambda_\ell^{\mathcal{N}}|$, $\xi_\ell^{\mathcal{N}}=-\frac{\mathrm{Re}(\lambda_\ell^{\mathcal{N}})}{|\lambda_\ell^{\mathcal{N}}|}$ where $\lambda_\ell^{\mathcal{N}}$ is the eigenvalues of the system of first-order differential equations reduced from Eq. (2.1). $\omega_\ell^{\mathcal{N}}$ is a function of the amount of system damping.

Further,

$$\tilde{\gamma}_\ell^\delta=\left(\xi_\ell^{\mathcal{N}}\alpha_\ell^\delta-\sqrt{1-(\xi_\ell^{\mathcal{N}})^2}\gamma_\ell^\delta\right),\quad \alpha_\ell^\delta=\mathrm{Re}(2\beta_\ell^\delta\psi_\ell),$$
$$\gamma_\ell^\delta=\mathrm{Im}(2\beta_\ell^\delta\psi_\ell),\quad \beta_\ell^\delta=\frac{-\psi_\ell^T\boldsymbol{M}\boldsymbol{\iota}}{2\lambda_\ell^{\mathcal{N}}\psi_\ell^T\boldsymbol{M}\psi_\ell+\psi_\ell^T\boldsymbol{C}\psi_\ell},$$

where ψ_ℓ is the corresponding eigenvector of λ_ℓ.

The discussion above of Eqs. (2.5) and (2.6) infers and inspires us to consider two phenomena in formulating an operator, the first one is that the responses at the present state is not influenced by the future ground acceleration, we understand it as causality of the system, meaning the state of system at current time should not be affected by the future, but only by its past history of the ground acceleration. The second one is the convolution nature of the Green's function kernel. As shown in many works with convolutional neural networks [1,38,44,49], the convolution function as a specific domain knowledge for neural network to learn about the target operator. With these two insights we will construct an operator in the DeepONet framework to address both causality and convolution kernel in Section 4.2, and name it as Causality-DeepONet.

3 Background / preliminary

In this section, first we will discuss the universal approximation theorem for operators by neural networks. Then in Section 3.2 we will review the DeepONet. Multi-scale deep neural networks and POD-DeepONet will be described in Sections 3.3 and 3.4, respectively. Furthermore, we discuss Fourier Neural Operator (FNO) in Section 3.5

3.1 Universal approximation theory

The universal approximation theory of neural networks is one of the mathematical basis for the broad applications of neural network, which justifies approximating functions or operators by weighted compositions of some functions, whose inputs are also weighted. The parameters, i.e., the weights and bias, could be obtained by minimizing a loss function with optimization algorithms such as stochastic gradient descent and its variants. Thus, neural network learning turns an approximation problem to one of optimization

with respect to the parameters. In this section, we focus on the approximation of operators.

The work of [5] gives a constructive procedure for approximating nonlinear operator $\mathcal{G}$ between continuous functions $\mathcal{G}(f)(x)$ in a compact subset of $C(\mathcal{X})$ with $\mathcal{X}\subseteq\mathbb{R}^d$ and continuous functions $f(x)$ in a compact subset of $C(\mathcal{F})$ with $\mathcal{F}\subseteq\mathbb{R}^d$

$$\mathcal{G}: f(x)\in C(\mathcal{F})\to\mathcal{G}(f)(x)\in C(\mathcal{X}), \tag{3.1}$$

where $C(\mathcal{X})$ and $C(\mathcal{F})$ are the continuous function spaces over $\mathcal{X}$ and $\mathcal{F}$, respectively, and $\mathcal{X}$ and $\mathcal{F}$ are compact subsets of $\mathbb{R}^d$, the Euclidean space of dimension d. The universal approximations with respect to operators are based on the two following results:

- **Universal Approximation of Functions [5]:** Given any $\varepsilon_1>0$, there exists a positive integer N, $\{\boldsymbol{w}_k\}_{k=1}^N\in\mathbb{R}^d$,$\{b_k\}_{k=1}^N\in\mathbb{R}$, such that functions $\mathcal{G}(f)(x)$ selected from a compact subset $\mathcal{U}$ of $C(\mathcal{X})$ could be uniformly approximated by a one-hidden-layer neural network with any Tauber-Wiener (TW) activation function σ_t

$$\left|\mathcal{G}(f)(x)-\sum_{k=1}^{N}\mathcal{J}_k(\mathcal{G}(f))\sigma_t(\boldsymbol{w}_k\cdot x+b_k)\right|<\varepsilon_1,\quad\forall\, x\in\mathcal{X}, \tag{3.2}$$

  where $\mathcal{J}_k(\mathcal{G}(f))$ is a linear continuous functional defined on $\mathcal{V}$ (a compact subset of $C(\mathcal{F})$), and all $\boldsymbol{w}_k$,b_k are independent of x and $f(x)$. A Tauber-Wiener activation function is defined as follows.

  Definition 3.1. Assume $\mathbb{R}$ is the set of real numbers. $\sigma:\mathbb{R}\to\mathbb{R}$ is called a Tauber-Wiener (TW) function if all the linear combinations $g(x)=\sum_{i=1}^{I}c_i\sigma(w_ix+b_i)$ are dense in every $C[a,b]$, where $\{w_i\}_{i=1}^I$, $\{b_i\}_{i=1}^I$, $\{c_i\}_{i=1}^I\in\mathbb{R}$ are real constants.

- **Universal Approximation of Functionals [5]:** Given any $\varepsilon_2>0$, there exists a positive integer M, m points $\{x_j\}_{j=1}^m\in\mathcal{F}$ with real constants c_i^k,W_{ij}^k,$B_i^k\in\mathbb{R}$, $i=1,\cdots,M$, $j=1,\cdots,m$, such that a functional $\mathcal{J}_k(\mathcal{G}(f))$ could be approximated by a one-hidden-layer neural network with any TW activation function σ_b

$$\left|\mathcal{J}_k(\mathcal{G}(f))-\sum_{i=1}^{M}c_i^k\sigma_b\left(\sum_{j=1}^{m}W_{ij}^k f\left(x_j\right)+B_i^k\right)\right|<\varepsilon_2,\quad\forall f\in\mathcal{V}, \tag{3.3}$$

  where the coefficients c_i^k,W_{ij}^k,B_i^k and nodes $\{x_j\}_{j=1}^m$ and m,M are all independent of $f(x)$.

Combining these two universal approximations, the authors of [5] proposed the universal approximations of nonlinear operators by neural networks when restricted to the compact subset $\mathcal{V}$ of the continuous function space $C(\mathcal{F})$ defined on a compact domain

$\mathcal{F}$ in $\mathbb{R}^d$. Namely, given any $\varepsilon>0$, there exists positive integers M,N, m points $\{x_j\}_{j=1}^m \in \mathcal{F}\subseteq\mathbb{R}^d$ with real constants $c_i^k, W_{ij}^k, B_i^k \in\mathbb{R}$, $i=1,\cdots,M, j=1,\cdots,m$, $\{\boldsymbol{w}_k\}_{k=1}^N\in\mathbb{R}^d$, $\{b_k\}_{k=1}^N\in\mathbb{R}$ that are all independent of continuous functions $f\in\mathcal{V}\subseteq C(\mathcal{F})$ and $x\in\mathcal{X}\subseteq\mathbb{R}^d$ such that

$$\left|\mathcal{G}(f)(x)-\sum_{k=1}^{N}\sum_{i=1}^{M}c_i^k\sigma_b\left(\sum_{j=1}^{m}W_{ij}^k f\left(x_j\right)+B_i^k\right)\sigma_t(\boldsymbol{w}_k\cdot x+b_k)\right|<\varepsilon. \tag{3.4}$$

3.2 DeepONet

Based on the universal approximation of nonlinear operator, Lu et al. [32] proposed the Deep Operator Network (known as DeepONet) by replacing the two one-hidden-layer neural networks in Eq. (3.4) with two deep neural networks. For a general operator $\mathcal{G}(f)(x)$, DeepONet has form

$$\mathcal{G}(f)(x)\sim\sum_{k=1}^{N}\underbrace{\sigma_{Br,k}\left(\left\{f\left(x_j\right)\right\}_{j=1}^m\right)}_{Br_k}\underbrace{\sigma_{T,k}(x)}_{T_k}, \tag{3.5}$$

where $\sigma_{Br}(\cdot)$ with a signal $\left\{f(x_j)\right\}_{j=1}^m$ as input is a deep neural network with N outputs, named as the branch net, $\sigma_T(\cdot)$ with input x is also a deep neural network with N outputs, called the trunk net. The branch net takes the input function as input and the trunk net takes the coordinates as input at which the output function need to be approximated. The schematics are shown in Fig. 2. The DeepONet has already been shown it is able to learn not only explicit mathematical operators like integration and fractional derivatives, but also PDE operators [3,8,9,27,32].

3.3 Multi-scale Deep Neural Network (MscaleDNN)

Multi-scale Deep Neural Network (MscaleDNN) [30] is a specific framework for problem whose output function is highly oscillatory. Since the highly oscillating feature of the responses of buildings excited by ground accelerations due to earthquakes, we also propose the multi-scale DeepONet that incorporates the multi-scale neural network into the DeepONet. The general fully connected neural network could learn the low frequency content of the data quickly, but the learning process will be stalled when higher frequency components are involved in the data. This frequency bias phenomenon is considered as the Frequency Principle studied in [45]. To remedy the frequency bias in terms of learning convergence, Liu et al. [30] introduced a MscaleDNN to accelerate the convergence of neural network for fitting problems of highly oscillating data. The MscaleDNN contains several sub-neural networks, for each of which a different frequency scaling is introduced by scaling the inputs accordingly, to arrive at the following form for the MscaleDNN,

$$f_{\boldsymbol{\theta}}(x)\sim\sum_{i=1}^{S}w_i f_{\boldsymbol{\theta}_i}(\mathcal{S}_i x), \tag{3.6}$$

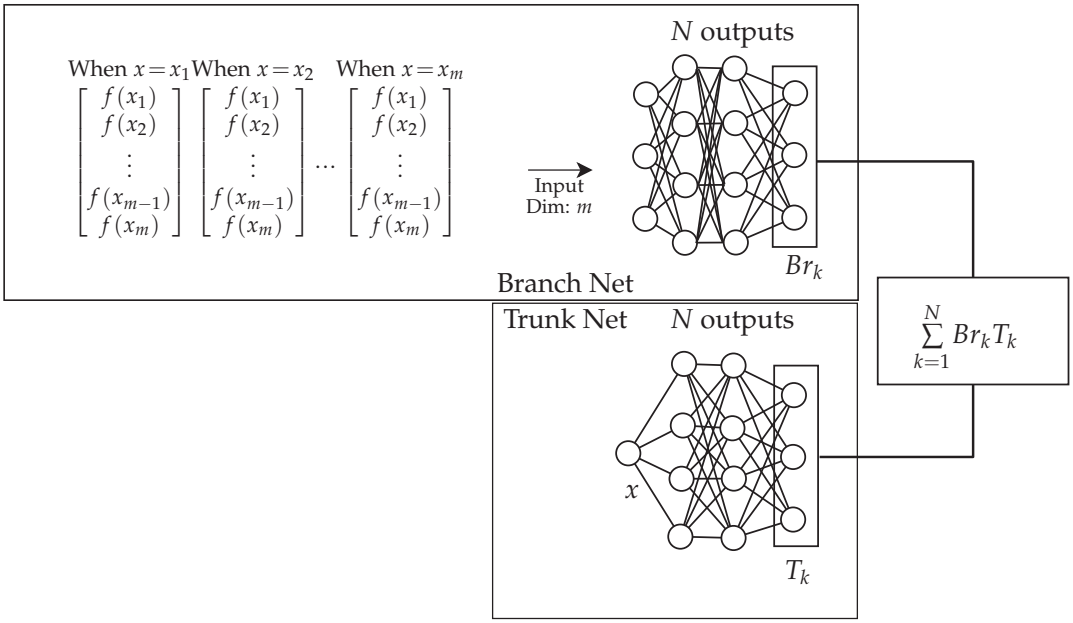

Figure 2: **Schematic Diagram of the DeepONet:** A schematic diagram of DeepONet showing branch and the trunk net along with the input data and output. The number of the input neurons of the branch net is equal to the number of sensor points in the input signals. The trunk net takes the input point x at where the output function need to be evaluated. Thus, the number of input neurons of trunk net is equal to the dimension of the problem. In the present study, it is the time point $x=t$ at which the output needs to be evaluated. Note for different time point t, the corresponding input of branch net is the same.

where $\mathcal{S}_i$ are the custom scales and $\{f_{\boldsymbol{\theta}_i}(\cdot)\}_{i=1}^{S}$ are the distinct sub-fully connected neural networks. Eq. (3.6) shows a multi-scale neural network with S scales. The final output of multi-scale deep neural network is the weighted sum of the outputs of the sub-neural networks with trainable weights w_i. The MscaleDNN has already shown its power for solving fitting problems and PDEs with high frequencies in [29,30,43].

3.4 POD-DeepONet

Instead of modeling the basis of output data by training the trunk net, Lu et al. [33] proposed the POD-DeepONet based on the work of Bhattacharya et al. [2]. The trunk net in the vanilla DeepONet is replaced by the basis obtained from proper orthogonal decomposition (POD) of the outputs of training data after subtracting the mean. Thus, the outputs of branch net are the coefficients of the precomputed basis vectors

$$\mathcal{G}(f)\sim\sum_{k=1}^{p}\sigma_{Br,k}\left(\{f(x_j)\}_{j=1}^{m}\right)\mathcal{B}_k+\mathcal{B}_0, \tag{3.7}$$

where $\mathcal{B}_0$ is the mean of the output of training data, $\{\mathcal{B}_k\}_{k=1}^{p}$ are the selected p basis vectors obtained by SVD or POD of the zero-mean outputs of training data, and $\sigma_{Br}(\{f(x_j)\}_{j=1}^{m})$ is the branch network with p outputs whose k^{th} output corresponds to the k^{th} singular value. In [33], it was shown that POD-DeepONet is more effective than the vanilla DeepONet.

3.5 Fourier Neural Operator (FNO)

Li et al. [25] proposed the framework Fourier neural operator where the integral kernels are parameterized in Fourier space directly. The input function is first lifted to a higher dimension, generally using a linear transformation. Then the iterative update to the representation from the input of l-th layer v_l to the output of l-th layer v_{l+1} is considered as,

$$v_{l+1}(x) := \sigma(Wv_l(x) + (\mathcal{K}(a;\phi)v_l)(x)) \tag{3.8}$$

given input function $a(x)$, where $W:\mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$ is a linear transformation, and $\sigma:\mathbb{R} \to \mathbb{R}$ is a non-linear activation function which is defined component-wise.

The kernel integral transformation is parameterized in the Fourier space

$$(\mathcal{K}(\phi)v_t)(x) = \mathcal{F}^{-1}(R_\theta \cdot \mathcal{F}(v_t))(x), \tag{3.9}$$

with $(\mathcal{F}f)_j(k) = \int_D f_j(x)e^{-2i\pi\langle x,k\rangle}dx$ and $(\mathcal{F}^{-1}f)_j(x) = \int_D f_j(k)e^{-2i\pi\langle x,k\rangle}dk$ the Fourier transform and inverse Fourier transform respectively. R_θ is the trainable parameter. For each kernel integral transformation, only specific m modes will be kept. Zero-padding after the kept modes is required to recover the same size as input for the output of each layer. The output is projected to the target dimension using a neural network. From the implementation level, the Fourier transform and inverse Fourier transform are realized by fast Fourier transform and inverse fast Fourier transform. The FNO is discretization-invariant and has shown the efficiency for learning PDE operators and so on, as shown in the paper [25].

4 Methodologies

4.1 Multi-scale DeepONet

As shown in Fig. 1, the spectrum of a typical earthquake signals contains not only low frequency components, but also high frequency components, therefore, we introduce the multi-scale DeepONet to handle the oscillatory information. Since the oscillatory features are function of time t, it is natural that we replace the fully connected trunk net by the

MscaleDNN with S scales.

$$\begin{aligned}\mathcal{G}(f)(t) &\sim \sum_{k=1}^{N} \sigma_{Br,k}\left(\{f(t_j)\}_{j=1}^{m}\right)\sigma_{T,k}(t),\\ \sigma_T(t) &= \sum_{i=1}^{S} w_i \sigma_{\boldsymbol{\theta}_i}(\mathcal{S}_i t),\end{aligned} \tag{4.1}$$

where $\{\sigma_{\boldsymbol{\theta}_i}(\cdot)\}_{i=1}^{S}$ are S distinct fully connected neural networks with N outputs.

4.2 Causality-DeepONet

The DeepONet proposed in [32] is based on a proven theorem of universal approximation for nonlinear operators, as introduced in the Section 3.1. We will apply the universal approximation theory to a nested subspaces of continuous functions indexed by the output time, which will provide a heuristic argument for the form of the Causality-DeepONet to be proposed. Rigorous mathematical justification though is still to be derived.

For a ground acceleration dynamic excitation $\ddot{u}_g(s)$, the response function $\mathcal{R}(\ddot{u}_g)(t)$ experiences a retardation effect due to the causality of the physical process. Therefore, in order to apply the universal approximation of function (3.2) to input function space $C[0,t]$, we view it as a subspace of $C[0,T]$ through an embedding by extending a function in $C[0,t]$ from interval $[0,t]$ to $[0,T]$ such that the extended function is zero in $[t+\delta,T]$ and a linear interpolation of its value at t and zero at $t+\delta$, δ is selected small enough to maintain the causality of the information up to t to some accuracy, i.e.,

$$C[0,t] \hookrightarrow C[0,T], \quad \forall t \in [0,T]. \tag{4.2}$$

We have

$$\left|\mathcal{R}(\ddot{u}_g)(t') - \sum_{k=1}^{N} \mathcal{J}_k\left(\mathcal{R}(\ddot{u}_g),t\right)\sigma_t\left(\boldsymbol{w}_k t' + b_k\right)\right| < \varepsilon_1, \quad \forall\, t' \in [0,t] \subseteq [0,T]. \tag{4.3}$$

Comparing with the original universal approximation theorem of functions Eq. (3.2), we require the dependence of t for the parameters $\mathcal{J}_k\left(\mathcal{R}(\ddot{u}_g),t\right)$, in which we may introduce the *Causality* and *Convolution*. It can be assumed that for every given interval $[0,t]$, the approximation Eq. (4.3) is valid within the interval based on the proof in [5]. In principle, the integer N, the parameters $\{\boldsymbol{w}_k\}_{k=1}^{N}$, $\{b_k\}_{k=1}^{N} \in \mathbb{R}$, should also have a t-dependence, however, due to the nested property in (4.2), for practical implementations they are taken as global parameters to be trained for all $t \in [0,T]$.

Following the discussion in Section 2, we extend the universal approximation of functionals Eq. (3.3) to approximate the functionals with causality. The functional $\mathcal{J}_k\left(\mathcal{R}(\ddot{u}_g),t\right)$ (defined on a compact subset of $C[0,t]$) could be rewritten as

$$\mathcal{J}_k\left(\mathcal{R}(\ddot{u}_g),t\right) = \mathcal{J}_k\left(\mathcal{R}(\ddot{u}_g\chi_{[0,t]})\right) \in \mathbb{R}, \quad \ddot{u}_g \in C[0,t], \;\; t \in [0,T], \tag{4.4}$$

where $\chi_{[0,t]}(s)$ is the characteristic function, such that

$$\chi_{[0,t]}(s)=\begin{cases}1, & s\in[0,t],\\ 0, & \text{otherwise.}\end{cases} \tag{4.5}$$

Then, following the similar approach as universal approximation of functionals Eq. (3.3), we may state that given any $\varepsilon_2>0$, there exists a positive integer M, $\hat{m}$ equal-spaced points $\{s_j\}_{j=1}^{\hat{m}}\in[0,t]$ with real constants $c_i^k,W_{ij}^k,B_i^k\in\mathbb{R}$, $i=1,\cdots,M$, $j=1,\cdots,\hat{m}$, such that $\mathcal{J}_k\left(\mathcal{R}(\ddot{u}_g),t\right)$ could be approximated by a one-hidden-layer neural network with any TW activation function σ_b

$$\left|\mathcal{J}_k(\mathcal{R}(\ddot{u}_g),t)-\sum_{i=1}^{M}c_i^k\sigma_b\left(\sum_{j=1}^{\lfloor\frac{t}{h}\rfloor}W_{i,\hat{m}-\lfloor\frac{t}{h}\rfloor+j}^k\ddot{u}_g\left(s_j\right)+B_i^k\right)\right|<\varepsilon_2,\quad\forall\ddot{u}_g\in C[0,t], \tag{4.6}$$

where the h is the step size and $\hat{m}=\lfloor\frac{t}{h}\rfloor$. For any given interval $[0,t]$, based on the results of [5], there exists $\hat{m}\in\mathbb{N}$ such that there are $\hat{m}$ points $\{s_j\}_{j=1}^{\hat{m}}$ that could be applied to construct the functional approximation Eq. (3.3). We further assume those points are equal-spaced. The equal-spaced sampling could be obtained by applying some appropriate smoothing kernel to input and output functions for the problems that are not equal-spaced. And likewise, the coefficients c_i^k,W_{ij}^k,B_i^k and nodes $\{s_j\}_{j=1}^{\hat{m}}$ and $\hat{m},M$ are all independent of $\ddot{u}_g(s)$, however, t-dependent.

The indicator function $\chi_{[0,t]}(s)$ Eq. (4.5), implemented by the inner upper summation limit $\lfloor\frac{t}{h}\rfloor$ in Eq. (4.6), as a discontinuous function does not belong to the continuous function space, which could be replaced by a smoothed version with a short transition at $s=t$ while still keeping the causality.

Causality-DeepONet: Combining these two desired universal approximations, we can heuristically consider the following DNN representation of an operator for retarded response for $t\in[0,T]\subset\mathbb{R}$. The basic idea is that we could find m points $\{s_i\}_{i=1}^{m}\in[0,T]$ to approximate the functional $\mathcal{J}_k(\mathcal{R}(\ddot{u}_g),T)$ based on the universal approximation of functionals with causality Eq. (4.6). The information to approximate the functional $\mathcal{J}_k(\mathcal{R}(\ddot{u}_g),t)$ where $[0,t]\subseteq[0,T]$ is offered by the value of $\{\ddot{u}_g(s_i)\}_{i=1}^{\lfloor\frac{t}{h}\rfloor}$ only. To keep the input signals of the same length at different time point, we consider zero-padding $\{\ddot{u}_g(s_i)\}_{i=1}^{\lfloor\frac{t}{h}\rfloor}$ as shown in Fig. 3. Thus the coefficients c_i^k,W_{ij}^k,B_i^k and nodes $\{s_j\}_{j=1}^{m}$ are t-independent. In addition, the convolution with respect to the input signals could be implemented by shifting the signals, as shown in Fig. 3.

Namely, we could find positive integers M,N, m equal-spaced points $\{s_j\}_{j=1}^{m}\in[0,T]$ with real constants $c_i^k,W_{ij}^k,B_i^k\in\mathbb{R}$, $i=1,\cdots,M$, $j=1,\cdots,m$, $\{w_k\}_{k=1}^{N}\in\mathbb{R}$, $\{b_k\}_{k=1}^{N}\in\mathbb{R}$ that are

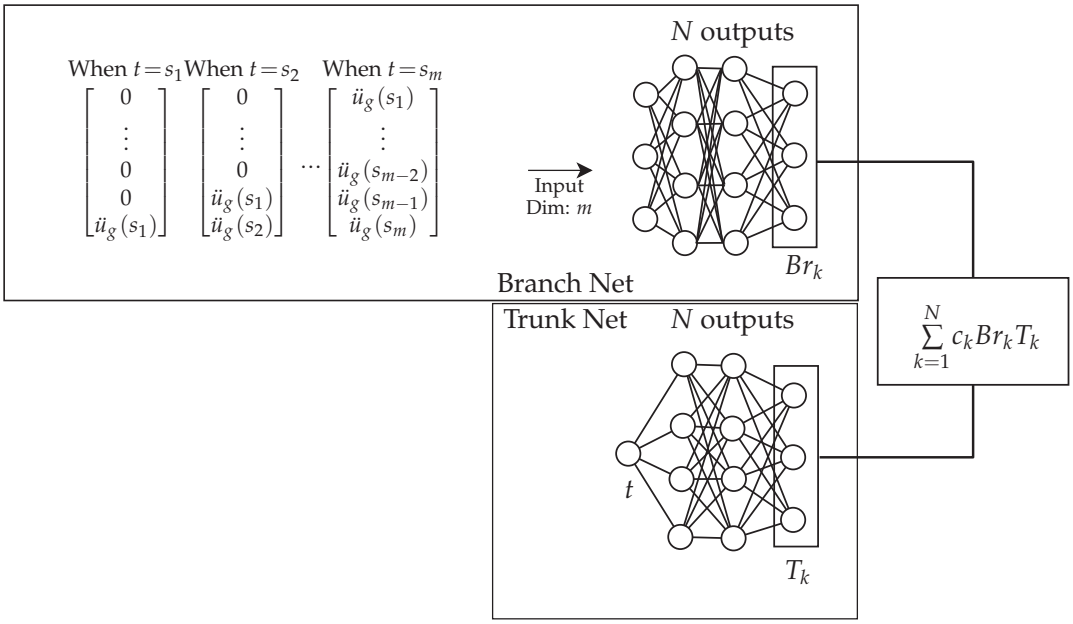

Figure 3: **Schematic Diagram of the Causality-DeepONet:** A schematic of the Causality-DeepONet showing branch and the trunk net along with the input data and output. Similar to DeepONet, the number of input neurons of branch of Causality-DeepONet is equal to the number of sensor points in the input signals. The input signals of branch, however, will be replaced by a zero-padding signals with a shifting window to express the causality and the convolution.

all independent to continuous functions $\ddot{u}_g \in C[0,T]$ and t, such that

$$\mathcal{R}(\ddot{u}_g)(t) \sim \sum_{k=1}^{N} c_k \underbrace{\sum_{i=1}^{M} \sigma_b \left(\sum_{j=1}^{\lfloor \frac{t}{h} \rfloor} W^k_{i,\lfloor \frac{T}{h} \rfloor - \lfloor \frac{t}{h} \rfloor + j} \ddot{u}_g(s_j) + \sum_{j=\lfloor \frac{t}{h} \rfloor}^{\lfloor \frac{T}{h} \rfloor - 1} W^k_{i,\lfloor \frac{T}{h} \rfloor - j} 0 + B^k_i \right)}_{Br_k} \underbrace{\sigma_t(\boldsymbol{w}_k t + b_k)}_{T_k}. \tag{4.7}$$

4.3 Loss function and error calculation

In the present study, we consider two loss functions depending on the method considered. Assuming $\hat{\boldsymbol{x}}_\ell(\boldsymbol{\theta})$ are the predicted response by neural network for the ℓ^{th} earthquake ground acceleration $\ddot{\boldsymbol{u}}_g^{(\ell)}$ with the corresponding true value $\boldsymbol{x}_\ell$, where $\boldsymbol{\theta}$ are trainable variables of the neural networks, including the weights and bias. The first loss func-

tion considered is the MSE loss function

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{n}\sum_{\ell=1}^{n}\|\boldsymbol{x}_\ell - \hat{\boldsymbol{x}}_\ell(\boldsymbol{\theta})\|^2 \\ &= \frac{1}{n}\sum_{\ell=1}^{n}\left[\frac{1}{m}\sum_{i=1}^{m}(x_\ell^{(i)} - \hat{x}_\ell^{(i)}(\boldsymbol{\theta}))^2\right],\end{aligned} \tag{4.8}$$

where n is the number of samples (different earthquake accelerations) considered and $\|\cdot\|^2$ is the MSE error for one sample, m is the number of points in each of the earthquake acceleration.

The second loss function considered is a weighted MSE loss function and defined as,

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{n}\sum_{\ell=1}^{n}\frac{1}{\max|\boldsymbol{x}_\ell|^2}\|\boldsymbol{x}_\ell - \hat{\boldsymbol{x}}_\ell(\boldsymbol{\theta})\|^2 \\ &= \frac{1}{n}\sum_{\ell=1}^{n}\frac{1}{\max|\boldsymbol{x}_\ell|^2}\left[\frac{1}{m}\sum_{i=1}^{m}(x_\ell^{(i)} - \hat{x}_\ell^{(i)}(\boldsymbol{\theta}))^2\right].\end{aligned} \tag{4.9}$$

The penalty is set to be the reciprocal of the square of maximum of the absolute value of the response. This act as a normalization factor when the responses have different magnitude for different earthquakes. The larger penalty is considered to the responses whose magnitude is smaller, thus it is expected that the neural network could predict the response whose magnitude is smaller accurately.

In order to check the accuracy of the predicted results we consider relative L_2 error,

$$\begin{aligned}\text{Relative } L_2 \text{ Error} &= \frac{1}{n}\sum_{\ell=1}^{n}\frac{\|\boldsymbol{x}_\ell - \hat{\boldsymbol{x}}_\ell(\boldsymbol{\theta})\|}{\|\boldsymbol{x}_\ell\|} \\ &= \frac{1}{n}\sum_{\ell=1}^{n}\sqrt{\frac{\sum_{i=1}^{m}(x_\ell^{(i)} - \hat{x}_\ell^{(i)}(\boldsymbol{\theta}))^2}{\sum_{i=1}^{m}(x_\ell^{(i)})^2}},\end{aligned} \tag{4.10}$$

and for the error with respect to the ℓ^{th} case, we consider the relative error,

$$\text{Err} = \frac{\max_i\left|x_\ell^{(i)} - \hat{x}_\ell^{(i)}(\boldsymbol{\theta})\right|}{\max_i\left|x_\ell^{(i)}\right|}. \tag{4.11}$$

The parameters $\boldsymbol{\theta}$ which include both weights and biases are optimised using the Adam optimizer [21] in a Pytorch [39] environment,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}). \tag{4.12}$$

Once the optimized parameters of the networks (weights and biases) are obtained, these may be used for the prediction of the response of the system for an unknown input signal (earthquake ground acceleration).

5 Numerical results and discussion

In this section, we will present the numerical results of multi-scale DeepONet (MS-DeepONet) and Causality-DeepONet for the prediction of response of the multistoried building discussed in Section 2. We will also have a comparison study of the results with a few other DeepONet methods and Fourier neural operator. First, we will study the prediction of the response with different DeepONet methods and the results of multi-scale DeepONet. Then, we will present predicted responses with Fourier neural operator, and Causality-DeepONet. We also study the different methods with different network sizes and training samples, which are discussed in subsequent sections. To avoid overfitting, dropout [16] is considered during training for a few of the cases but is disabled during evaluation. Further, we consider a 0 displacement at 0 time instant to enforce the solution satisfying the initial conditions for all the numerical examples in this section.

The testing dataset consists of different earthquakes which are not included in the training dataset. The test dataset is considered from 22 different earthquakes. One of them is recorded at three different stations. Two of them are recorded at the same station. Thus, the testing dataset consists of 44 ground accelerations (2 horizontal directions) from different earthquake recording stations. The training dataset consists of different earthquakes not considered in the testing dataset. The training dataset may be from the same earthquake recorded at different stations or different earthquakes. The record series number (RSN) for the training and testing dataset are listed in Appendix A. Details about the training and testing dataset are shown in Table 5 in Appendix B.

5.1 DeepONet, POD-DeepONet, and Multi-scale DeepONet

First, we will present the predicted results with the DeepONet, POD-DeepONet, and Multi-scale DeepONet method. We use the notation $[N_1]$-$[N_2]\times L$-$[N_3]$ to represent a neural network with the input size of N_1, L hidden layers with N_2 neurons in each layer, and the output dimension of N_3 neurons. The notation $[N_1]$-$S\times\{[N_2]\times L\}$-$[N_3]$ represents a multi-scale neural network with the input size of N_1, and S sub-neural networks that contain L hidden layers with N_2 hidden neurons in each layer. The output dimension is N_3.

The network structures considered is [4000]-[200]×3-[200] for branch and [1]-[200]×3-[200] for trunk for the DeepONet case. As for the POD-DeepONet, since there is no trunk net, the network structure considered is [4000]-[200]×3-[100]. The reason that POD-DeepONet has 100 output neurons is the basis we considered contains 100 basis vector. For the multi-scale DeepONet case, we consider the branch net size [4000]-[200]×3-[200] and trunk net size [1]-20×{[10]×3}-[200]. To keep the number of neurons as the same as previous cases, the number of hidden neurons for each subnet at each layer are divided by 20, the number of scales. In the present study we consider an MscaleDNN in the trunk with 20 equally spaced scales $[1,1+40\pi,\cdots,1+40n\pi,\cdots,1+760\pi]$. In the meantime, the time t is scaled to $t\in[0,1]$ for multi-scale DeepONet case. All the neural networks in this

Table 1: Relative L_2 error for training and testing dataset using DeepONet(DON), POD-DeepONet(PDON), and Multi-scale DeepONet(MSDON).

Case	Branch	Trunk	Loss Eq. (4.8)		Loss Eq. (4.9)	
			Train	Test	Train	Test
DON	[4000]-[200]×3-[200]	[1]-[200]×3-[200]	1.0	0.999	1.0	0.999
PDON	[4000]-[200]×3-[100]	None	0.448	1.213	0.035	1.003
MSDON	[4000]-[200]×3-[200]	[1]-20×{[10]×3}-[200]	0.13	0.87	0.10	0.83

section are trained up to 20000 epochs with Adam optimizer.

The activation considered for three network structures are slightly different. For the DeepONet and POD-DeepONet case, we consider ReLU(x) activation function. For the multi-scale DeepONet case, we consider $\sin(x)$ activation function, according to the results in [29, 43].

The training of DeepONet is considered with a learning rate of 10^{-4} in the first 1000 epochs, then 10^{-5} in the 1000 to 3000 epochs, and 10^{-6} for the remaining epochs. In order to avoid overfitting, we consider using dropout with a rate of 0.01 for the branch net and L_2 weight regularization with 3×10^{-5} coefficient for weights of the branch net as well.

Instead, for the multi-scale DeepONet case, we assign the dropout rate of 0.10 for the trunk net. The learning rate considered is 3×10^{-4} in the first 1000 epochs, then 1.5×10^{-4} in the 1000 to 2500 epochs, and 7.5×10^{-5} for the rest of the epochs up to 5000.

To avoid the overfitting of the training of POD-DeepONet, we consider L_2 weight regularization with coefficient 10^{-6}. The learning rate considered is 10^{-3} in the first 5000 epochs, then 10^{-4} in the 5000 to 10000 epochs, and 10^{-5} for the rest of the epochs up to 20000.

As discussed in Section 4.3, we consider two different loss functions given by Eqs. (4.8) and (4.9) for all the cases. The training samples for all 3 models are 100 samples. The training and testing results are shown in Table 1. It could be observed that the errors in predicted responses are high for both training and testing dataset for DeepONet(DON) from Table 1. Though the POD-DeepONet (PDON) and the multi-scale DeepONet (MSDON) have some improvements for the training data, the errors in the testing dataset are still large. A further observation is that the training relative L_2 errors with Loss Eq. (4.9) are smaller comparing with the training relative L_2 error with loss Eq. (4.8).

The relative L_2 errors for the POD-DeepONet case and multi-scale DeepONet case are smaller compared to DeepONet for the training dataset. However, the performance of the network is poor in the case of the testing dataset shows that 100 training samples could not offer enough information for the target response space. The predicted response for the best training and testing cases are shown in Appendix C.2, Appendix C.3, and Appendix C.4, respectively. The performance of POD-DeepONet highly relies on the quality of the training data. If the training dataset covers a large enough region of the space of interest, then the POD-DeepONet could have very good performance. It could

be also observed the proposed multi-scale DeepONet is not desired based on the results of testing cases, though it accelerated the convergence for the training process.

5.2 Fourier Neural Operator(FNO)

In the previous section, we discussed the results of DeepONet, POD-DeepONet and multi-scale DeepONet. We observed that the results were not satisfactory. In this section, we will present and discuss the results of the Fourier neural operator(FNO). The FNO considered in this example is implemented using the python package available at `https://github.com/neuraloperator/neuraloperator`. As introduced in Section 3.5, we consider 4 Fourier integral operator layer and 1000 modes will be kept. The activation function, the number of lifting channel, the number of projection channel and the skip connection considered are GeLU(x), 256, 256 and soft-gating, respectively, all of which are the default settings of the neuraloperator package.

We only consider the loss function given by Eq. (4.9) since in the previous cases, the loss function Eq. (4.9) offers better performance in the training cases, comparing with loss function Eq. (4.8). Like the previous studies, we provide the initial conditions as additional data point $\{0,x(t_1),x(t_2),\cdots,x(t_m)\}$ for every response in the training dataset to force the FNO satisfy the initial conditions. Further, to ensure the prediction of FNO satisfy another condition, $\mathcal{R}(\mathbf{0})(t)=0,\forall t$, i.e., the response will be 0 if the input signals are $\mathbf{0}$, we add the zero signal pair $(\mathbf{0},\mathbf{0})$ to the training dataset. It is observed that this additional zero pair data improves the accuracy of the results for the FNO case.

Fig. 4 shows the evolution of relative L_2 error for the training and testing dataset for a Fourier Neural Operator trained up to 20000 epochs. The final relative L_2 error is 3.4% for testing cases. The training dataset size considered is 10. Fig. 5 and Fig. 6 show the worst and the best predictions, respectively, using FNO for the test dataset. To avoid overfitting, we consider L_2 weight regularization with a coefficient 1×10^{-4}. The learning rate considered is 10^{-3} in the first 1000 epochs, then 10^{-4} in the 1000 to 3000 epochs, and 10^{-5} for the rest of the training up to 20000 epochs. It could be observed that FNO can predict the responses with good accuracy, though the error for the worst case is large.

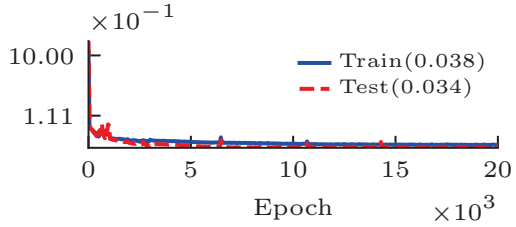

Figure 4: **Relative L_2 Error for Training and Testing Dataset when Using Fourier Neural Operator:** The plot shows the relative L_2 error for training and testing dataset when using Fourier neural operator.

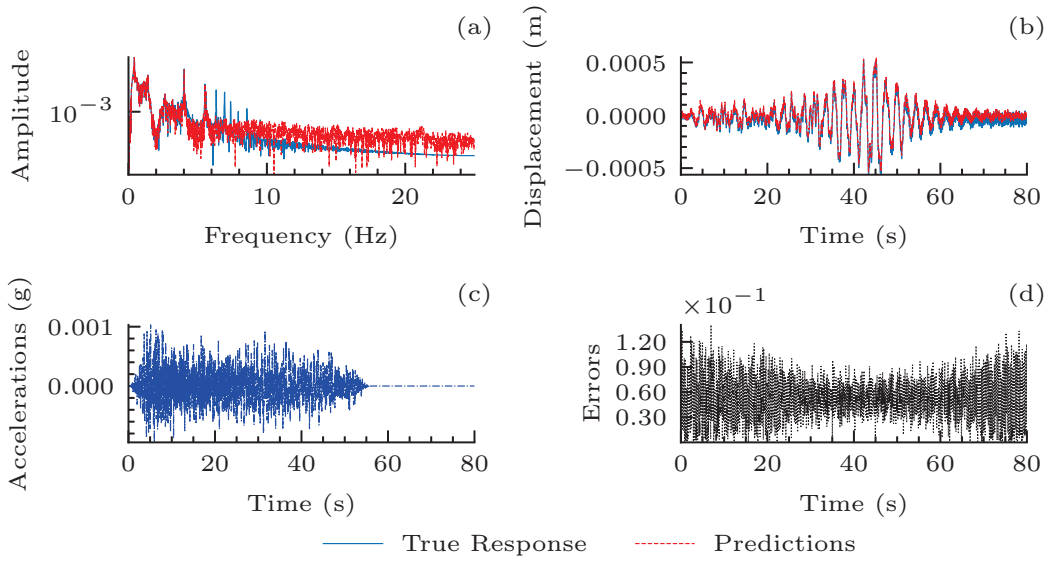

Figure 5: **The Worst Case of Predictions of FNO(Relative L_2 Error: 0.224):** The worst predictions in testing dataset for the FNO. (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

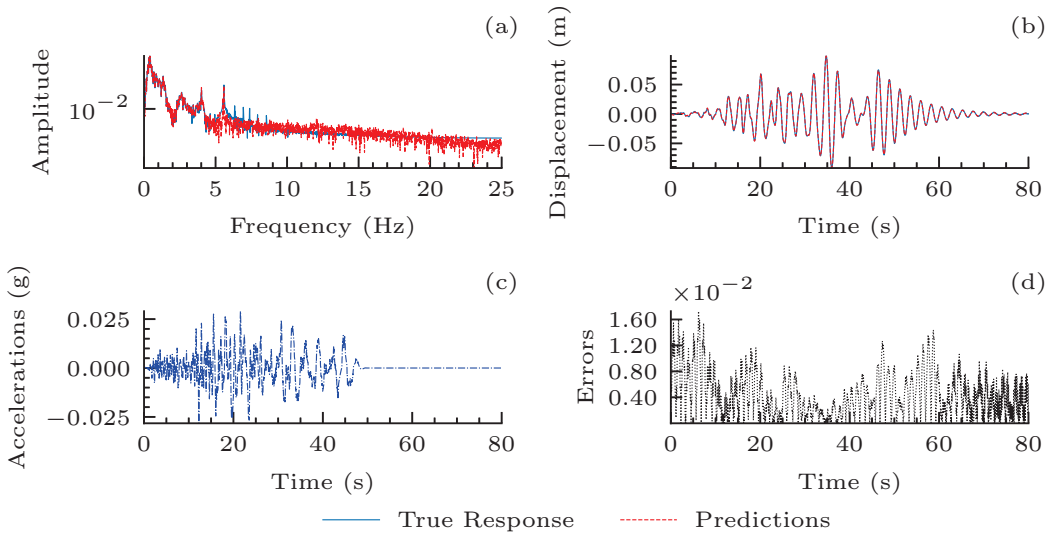

Figure 6: **The Best Case of Predictions of FNO(Relative L_2 Error: 0.021)** The best predictions in testing dataset for the FNO. (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

Table 2: Relative L_2 Error for training and testing dataset when using different sizes of Causality-DeepONet.

Case	Branch	Trunk	Sample	Loss Eq. (4.8)		Loss Eq. (4.9)	
				Train	Test	Train	Test
1	[4000]-[30]×2-[30]	[1]-[30]×2-[30]	10	0.055	0.089	0.017	0.018
2	[4000]-[60]×2-[60]	[1]-[60]×2-[60]	10	0.023	0.040	0.014	0.015
3	[4000]-[90]×2-[90]	[1]-[90]×2-[90]	10	0.030	0.051	0.015	0.016
4	[4000]-[120]×2-[120]	[1]-[120]×2-[120]	10	0.018	0.033	0.016	0.017

[1] The notation [N_1]-[N_2]×2-[N_3] represents a neural network with the input size of N_1, 2 hidden layers with N_2 neurons in each layer, and the output dimension of N_3 neurons.

5.3 Causality-DeepONet

As discussed in Section 2 and Section 4.2, both convolution and causality are considered in the formulation of Causality-DeepONet. In this section, we will present the numerical results and a comprehensive discussion about Causality-DeepONet for the prediction of the responses of the problem discussed in Section 2.

Similar to the previous numerical examples, in this study as well, we consider the two loss functions given by Eqs. (4.8) and (4.9) with different network sizes. We also study the effect of the number of training samples on the accuracy of test results. Further, given the fact that there are multiple choices of activation functions, we test a few of the popular activation functions with the same training dataset and the same network sizes. As shown later, Causality-DeepONet with standard sigmoid activation functions converges much slower. By defining a custom sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}} - \frac{1}{2}$, the performance is improved. Furthermore, to study the effect of only causality without convolution and only convolution but without causality, we do numerical studies with causality only and convolution only, observing that both causality and convolution are indispensable components of the proposed Causality-DeepONet. Like the previous studies, we provide the initial conditions as additional data pairs $\{\ddot{u}_g : [0,0,\cdots,0], x(0) : 0\}$ in the training dataset to enforce the Causality-DeepONet satisfies the initial conditions.

Fig. 7 and Fig. 8 show the worst and the best predictions using Causality-DeepONet for the test dataset. The network is trained using 100 training samples. The network size considered is [4000]-[120]×2-[120] for branch and [1]-[120]×2-[120] for trunk. The activation function considered is $\tanh(x)$. To avoid overfitting, we consider L_2 weight regularization with a coefficient 1×10^{-4} for the branch net. The learning rate considered is 10^{-3} in the first 2000 epochs, then 10^{-4} in the 2000 to 10000 epochs, and 10^{-5} for the rest of the training up to 20000 epochs. The loss function considered for this case is Loss Eq. (4.9). It could be observed that Causality-DeepONet can predict the responses with good accuracy for all the cases, as the error for the worst case also is within the satisfactory limit.

To study the effect of different network size and loss function (Eqs. (4.8) and (4.9)),

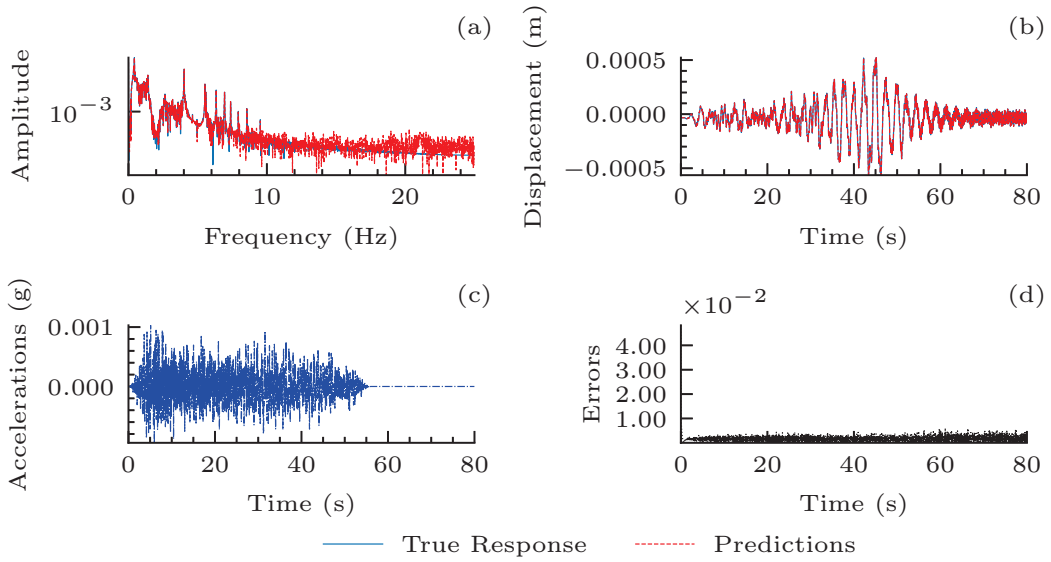

Figure 7: **The Worst Case of Predictions of Causality-DeepONet(Relative L_2 Error: 0.008):** The worst predictions in testing dataset for the Causality-DeepONet. (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

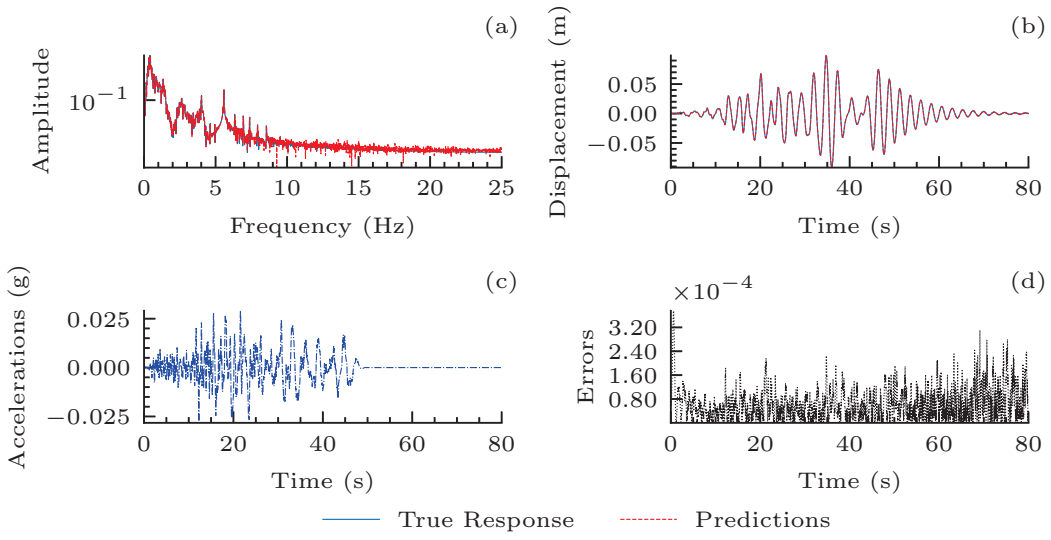

Figure 8: **The Best Case of Predictions of Causality-DeepONet(Relative L_2 Error: 0.00030)** The best predictions in testing dataset for the Causality-DeepONet. (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

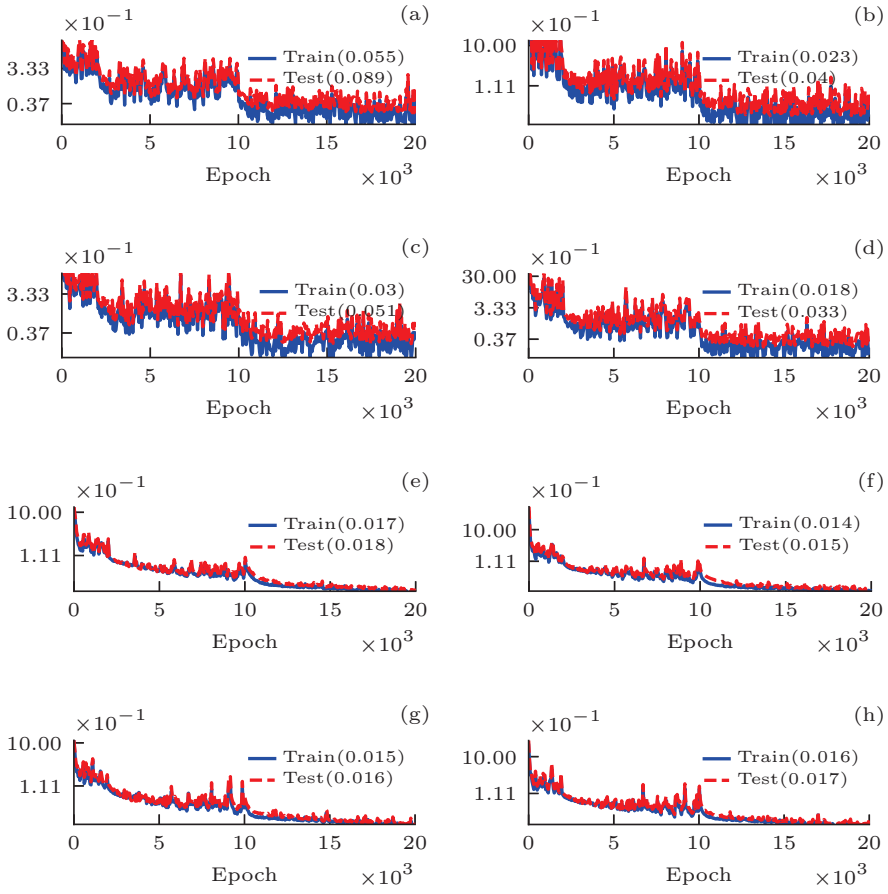

Figure 9: **Relative L_2 Error for Training and Testing Dataset when Using Causality-DeepONet with Different Branch and Trunk Sizes:** The plots (a), (b), (c), (d) are the cases corresponding to Case-1 to Case-4 of Table 2, respectively, with Loss function Eq. (4.8). Similarly, the plots (e), (f), (g), (h) are the cases corresponding to Case-1 to Case-4 of Table 2, respectively, with Loss function Eq. (4.9).

we consider different network sizes with the same number of training dataset. The activation function considered for all the cases is $\tanh(x)$. To avoid overfitting, we consider L_2 weight regularization for the branch net with a coefficient 10^{-4}. The learning rate considered is 10^{-3} in the first 2000 epochs, then 10^{-4} in the 2000 to 10000 epochs, and 10^{-5} for the rest of the epochs up to 20000 epochs. The relative L_2 errors for both training and testing dataset after completion of training is shown in Table 2 and the corresponding relative L_2 error with epoch is shown in Fig. 9. It could be observed that the proposed Causality-DeepONet shows a good accuracy in both the case of loss functions. The MSE loss function given by Eq. (4.8) is more sensitive to the network size as relative L_2 errors

Table 3: Relative L_2 Error for training and testing dataset when using Causality-DeepONet with different activation functions.

Case	Activation	Sample	Relative L_2 Error Eq. (4.10)	
			Train	Test
1	$\tanh(x)$	10	0.016	0.017
2	$\sin(x)$	10	0.011	0.016
3	Sigmoid(x)	10	0.142	0.111
4	ReLU(x)	10	0.014	0.028
5	Custom Sigmoid (Eq. (5.1))	10	0.02	0.024

for both training and testing are reduced with an increase in network sizes. The weighted loss function given by Eq. (4.9) is less sensitive to the network sizes for this numerical study. Further, it is also observed that the relative L_2 error in the case of loss function Eq. (4.9) is less than that of relative L_2 error in the case of loss function Eq. (4.8). Thus, we conclude that the additional penalty terms in the loss function remove the bias from the magnitude of output functions/data in the present study. For the further numerical studies conducted, we consider with loss function given by Eq. (4.9) only.

As discussed earlier, there are multiple choices of activation functions, and we test the performance of Causality-DeepONet with a few popular activation functions. For this purpose, we consider the same network sizes and training dataset and loss functions (Eq. (4.9) for all the cases of activation function considered. The network sizes considered are [4000]-[120]$\times$2-[120] for branch and [1]-[120]$\times$2-[120] for trunk. To avoid overfitting, we consider L_2 weight regularization for the parameters in branch net with a coefficient 10^{-4} for case 1,2,4 and without L^2 regularization for branch and trunk net for case 3 and 5 in Table 3. The learning rate considered is 10^{-3} in the first 2000 epochs, then 10^{-4} in the 2000 to 10000 epochs, and 10^{-5} for the rest of the epoch up to 20000 epochs. As shown in Table 3 and Fig. 10, It could be concluded that the Causality-DeepONet with $\tanh(x)$, $\sin(x)$ and ReLU(x) as activation functions obtains excellent predictions given limited training samples. However, the Causality-DeepONet with sigmoid as activation functions is not convergent as expected. By shifting the sigmoid function, we define a custom sigmoid function

$$\sigma(x) = \frac{1}{1+e^{-x}} - \frac{1}{2} \tag{5.1}$$

thus improve the results, as shown in Fig. 10(e) and case 5 in Table 3.

From the above discussion it could be observed that the proposed Causality-DeepONet is able to predict the response of the problem considered with a good accuracy. We also study the effect of training samples in the accuracy of predicted response. For this purpose we consider different samples with same network size and other hyperparameters. The statistical properties of the different training samples are shown in Table 5 in Appendix B. It could be noted that the training samples in datasets Train-I, Train-II,

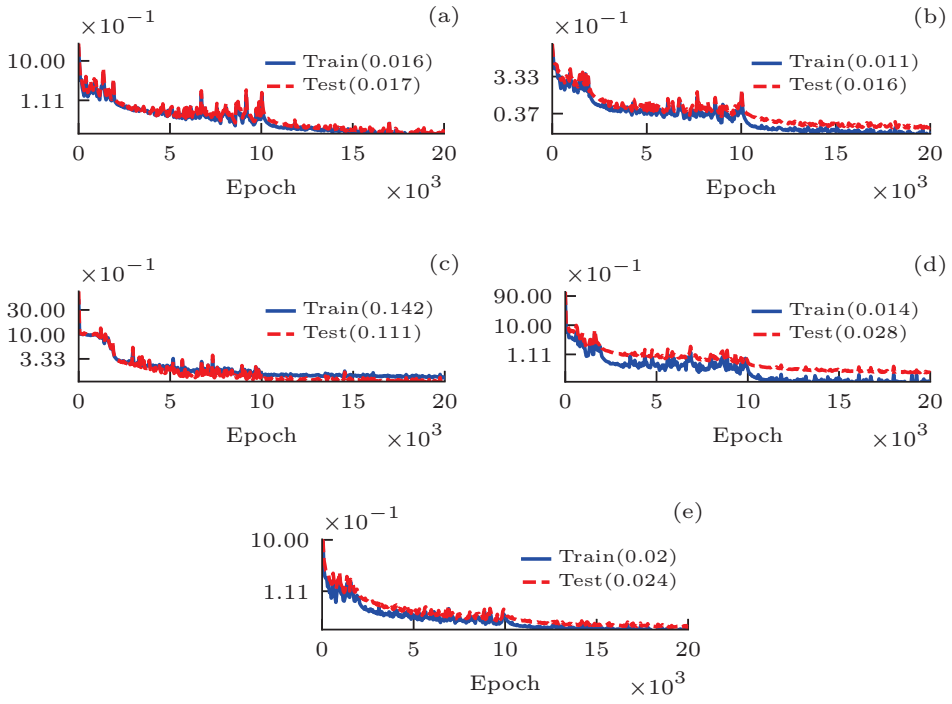

Figure 10: **Relative L_2 Error for Training and Testing Dataset when Using Causality-DeepONet With Different Activation Functions:** The plots (a)-(e) correspond to cases Case-1 to Case-5 of Table 3, respectively, with Loss function Eq. (4.9).

Train-III are exclusively different. The training samples in datasets Train-II and Train-III are included in the dataset Train-IV. The training samples in dataset Train-I and Train-IV are included in dataset Train-V. The training samples in dataset Train-V are included in dataset Train-VI.

We consider a network size of [4000]-[120]×2-[120] for branch and [1]-[120]×2-[120] for trunk. The activation function considered is $\tanh(x)$. To avoid overfitting, we consider L_2 weight regularization with a coefficient 1×10^{-4} for the branch net. The learning rate considered is 10^{-3} in the first 2000 epochs, then 10^{-4} in the 2000 to 10000 epochs, and 10^{-5} for the 10000 to 20000 epochs. The relative L_2 errors of different cases with different sample in training are shown in Table 4 and Fig. 11. It could be observed that the L_2 error is small even with smaller number of training set and the accuracy increases with the increase in number of training set, though the improvements in accuracy is limited with increase in number of samples. On the other hand, it could also be observed that the performance of Causality-DeepONet of Case-3 in Table 4 with 10 training samples that have larger deviation is poor comparing with Case-4 to Case-6 in Table 4. The further

Table 4: Relative L_2 error for training and testing dataset when using Causality-DeepONet with different numbers of training samples.

Case	Sample	Relative L_2 Error Eq. (4.10)		Train data set
		Train	Test	
1	7	0.006	0.01	Train-I
2	8	0.004	0.008	Train-II
3	10	0.016	0.017	Train-III
4	20	0.008	0.005	Train-IV
5	50	0.003	0.003	Train-V
6	100	0.002	0.001	Train-VI

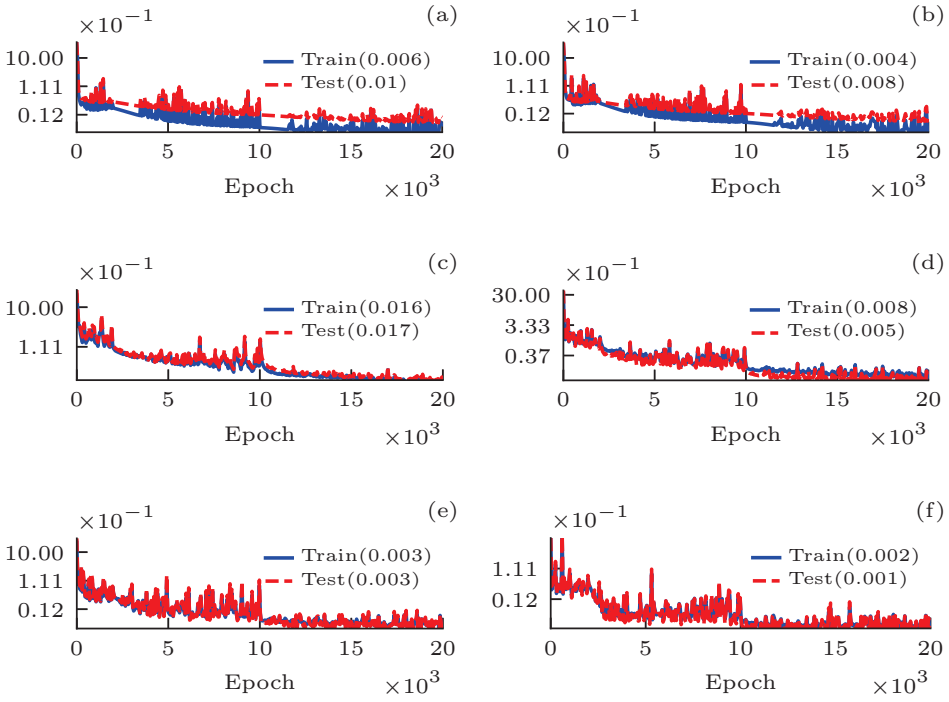

Figure 11: **Relative L_2 Error for Training and Testing Dataset when Using Causality-DeepONet with Different Number of Training Samples:** The plots (a)-(e) correspond to the cases Case-1 to Case-5 in Table 4, respectively, with Loss function Eq. (4.9).

observation from Table 4 is that the training relative L_2 error is greater than the testing relative L_2 error in Case-4 of Table 4, given the fact that dataset Train-IV contains Train-II and Train-III.

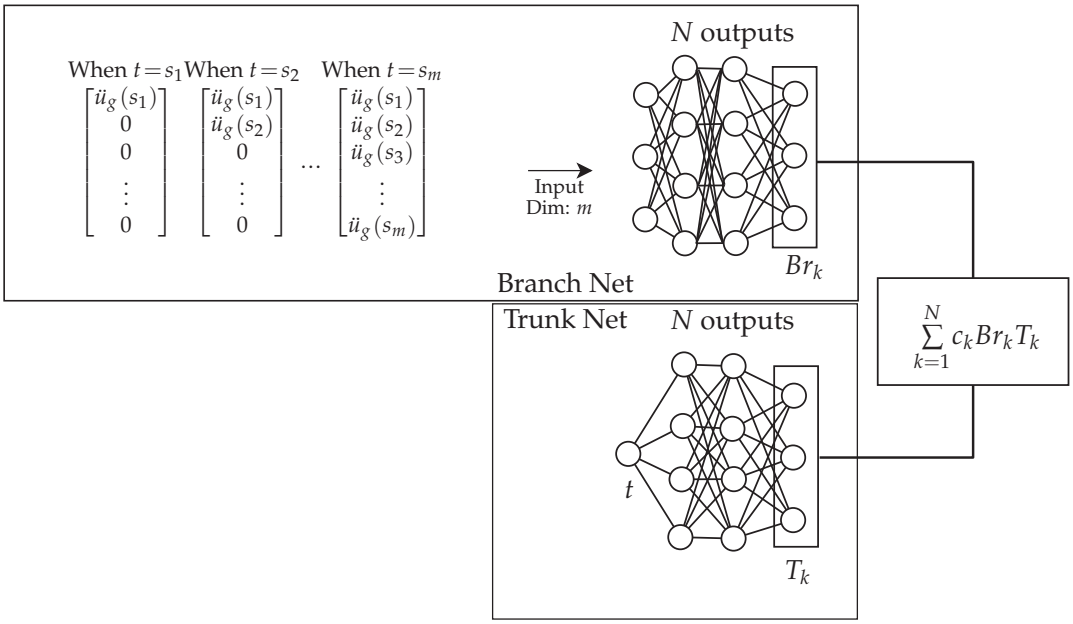

Figure 12: **Schematic Diagram of Causality-DeepONet without Convolution:** A schematic of the Causality-DeepONet without convolution showing branch and the trunk net along with the input data and output. The number of input neurons for the branch is equal to the number of sensor points in the input signals. The input signals for the branch, however, are replaced by a zero-padding for the future information.

As discussed in Section 2, the proposed Causality-DeepONet involves both the phenomenon of causality and convolution. To evaluate the importance of convolution on the accuracy of prediction, we study the method only with causality but without convolution. The neural network considered for this purpose has form

$$\mathcal{R}_c(\ddot{u}_g)(t) \sim \sum_{k=1}^{N} c_k \underbrace{\sum_{i=1}^{M} \sigma_b \left(\sum_{j=1}^{\lfloor \frac{t}{h} \rfloor} W_{i,j}^k \ddot{u}_g(s_j) + \sum_{j=\lfloor \frac{t}{h} \rfloor + 1}^{\lfloor \frac{T}{h} \rfloor} W_{i,j}^k 0 + B_i^k \right)}_{B_k} \underbrace{\sigma_t(\boldsymbol{w}_k t + b_k)}_{T_k}. \tag{5.2}$$

The difference between the formulation given by Eq. (5.2) and the proposed Causality-DeepONet (Eq. (4.7)) is in the difference in the weights of the branch. In the case of the formulation without convolution, the weights are $W_{i,j}^k$, whereas in the case of the proposed Causality-DeepONet (with convolution) the weights are $W_{i,\lfloor \frac{T}{h} \rfloor - \lfloor \frac{t}{h} \rfloor + j}^k$.

To study the effect of convolution, we compare the results of the problem with Causality-DeepONet and Causality-DeepONet without convolution.

A network size of $[4000]$-$[120]\times 2$-$[120]$ for branch and [1]-$[120]\times 2$-[120] for the trunk are considered in both the cases. The activation function considered for both cases is

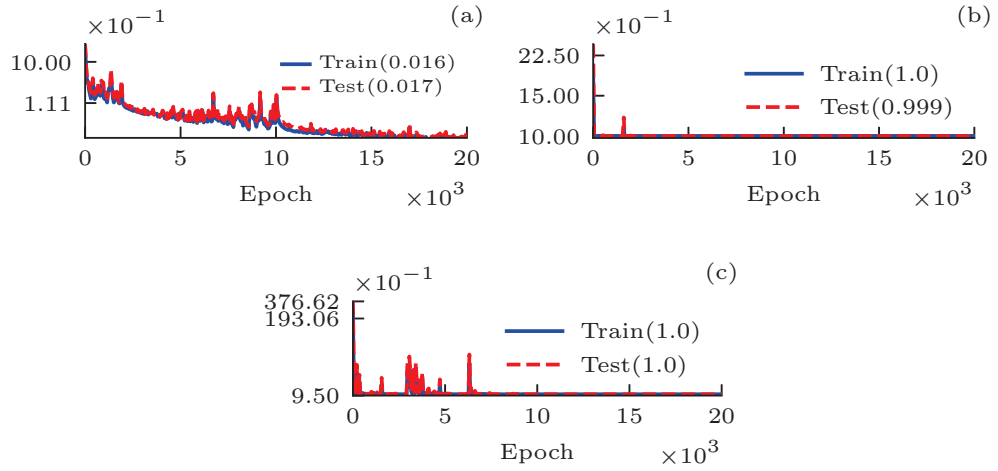

Figure 13: **Relative L_2 Error for Training and Testing Dataset when Using Causality-DeepONet with or without Convolutions:** (a) Causality-DeepONet with convolution with loss function Eq. (4.9), 10 training samples. (b) Proposed Net with causality only with loss function Eq. (4.9), 100 training samples. (c) Convolutional DeepONet with loss function Eq. (4.9), 100 training samples.

$\tanh(x)$. To avoid overfitting, we consider L_2 weight regularization for the branch net with a coefficient 10^{-4}. The learning rate considered is 10^{-3} in the first 2000 epochs, then 10^{-4} in the 2000 to 10000 epochs, and 10^{-5} for the rest of the epoch up to 20000 epochs. The case without convolution is considered to be trained with 100 training samples. However, the case with convolution is trained with 10 training samples only. Fig. 13(a) and (b) show the relative L_2 error for both cases. It can be observed that the variant without convolution is not able to provide satisfactory accuracy.

As further comparison, we apply the convolutional DeepONet (CNNDeepONet) but without causality incorporated to the response problems, to show that the proposed Causality-DeepONet requires both convolution and causality. Convolution neural network (CNN) in DeepONet is considered in [22], where the authors considered CNN along with DNN in the branch network. The convolutional neural network considered in the present study has 3 convolutional layer with kernel size 10 with output channel size of 100 for each layer. A maxpooling layer with kernel size 3 is set between the convolutional layer. The DeepONet components contain a branch net with size $[4000]$-$[120]\times 2$-$[120]$ and a trunk net with size [1]-$[120]\times 2$-[120]. The activation considered is $\mathrm{ReLU}(x)$. To train the CNNDeepONet, the training data size considered is 100, as the same as the case with causality only. The learning rate considered is 10^{-4} for the first 1000 epochs, 5×10^{-5} for the 1000 to 3000 epochs, and 2.5×10^{-5} up to 20000 epochs. To avoid overfitting, the L_2 weight regularization with a coefficient 3×10^{-5} is considered for the parameters in branch net. Fig. 13(c) shows the evolution of relative L_2 error for training and testing

dataset when using convolutional DeepONet. It could be concluded that for the problem we considered, DeepONet only with convolution neural network along with DNN in the branch net does not provide satisfactory results, compared with the proposed causality DeepONet Fig. 13(a).

6 Conclusion and future works

In this paper, we have studied how to improve the accuracy of DeepONet for causal oscillatory linear dynamical systems. Two new variants of DeepONet, the multi-scale DeepONet and the Causality-DeepONet are proposed. As an application, we considered the problem of learning the mapping between earthquake ground accelerations and building's causal responses, which are both highly oscillatory. In the multi-scale DeepONet, multi-scale neural networks are used in the trunk net. Meanwhile, the Causality-DeepONet includes both causality and convolution as specific domain knowledge in its design. Though the multi-scale DeepONet improved the training of the seismic response operator, it failed to give satisfactory prediction results in the test cases. However, the Causality-DeepONet is able to provide accurate predictions in the test cases as well. We have also studied the effect of the size of networks, the number of training samples, and the type of activation functions on the accuracy of prediction of responses using Causality-DeepONet. It can be found that the proposed Causality-DeepONet can provide good accuracy in the prediction of response of the problem considered, comparing with various existing neural network methods.

For future work, the Causality-DeepONet for nonlinear problems such as nonlinear dynamics, nonlinear electrical circuits etc, may be considered. Also future studies should include establishing a solid mathematical foundation for the Causality-DeepONet by extending the work of [5] to the proposed framework of the Causality-DeepONet.

Acknowledgments

The authors like to thank Prof. George Em Karniadakis for suggesting this research project. The work of W. Cai is supported by the US National Science Foundation grant DMS-2207449. The work of K. Nath is supported by OSD/AFOSR MURI grant FA9550-20-1-0358.

Appendices

A Dataset

In this section, we list the record series number(RSN) of the earthquake considered for testing and training dataset from the website `https://ngawest2.berkeley.edu/site`.

The interested readers are welcomed to register and download the data. The RSN for the training dataset is 1147, 1149, 1153, 1154, 1155, 1157, 1160, 1164, 1177, 1187, 1194, 1196, 1198, 1199, 1203, 1204, 1205, 1206, 1207, 1209, 1210, 1211, 1212, 1214, 1215, 1218, 1220, 1226, 1227,1228, 1230, 1232, 1234, 1235, 1236, 1237, 1244, 1245, 1246, 1248, 1249, 1260, 1261, 1263, 1265, 1273, 1277, 1285, 1286, 1287, 2200, 3000, 1762, 1773, 3800, 3900, 4000, 4200, 4400, 4500, 4900, 5900, 6200, 8100, 8400, 8700, 8800, 9500.

The RSN for the testing dataset is 1127, 1145, 11700, 12, 14, 142, 15, 1620, 2108, 24, 293, 323, 35, 38, 446, 53, 582, 65, 721, 9, 913, 958.

B Additional tables

In Table 5, we show the statistical properties of the training and testing dataset. The total number of testing dataset considered is 44 earthquake ground accelerations. These test data are considered for all the numerical examples. It is also important to note that in the

Table 5: Properties of the earthquake records considered for training and testing of neural networks.

Case		Test	Train-I	Train-II	Train-III	Train-IV	Train-V	Train-VI
Samples		44	7	8	10	20	50	100
Mag.	Min	4.30	6.30	6.30	4.92	4.90	4.90	4.70
	Max	7.90	7.62	7.62	7.62	7.62	7.62	7.62
	Mean	6.58	7.27	7.21	7.07	6.85	7.01	7.10
	SD	0.82	0.52	0.51	0.83	0.94	0.83	0.82
PGA $\max\|\ddot{u}_g\|$	Min	0.00103	0.00119	0.00119	0.00245	0.00245	0.00119	0.00119
	Max	0.35726	0.11473	0.11473	0.31313	0.31313	0.31313	0.31313
	Mean	0.05298	0.06558	0.05768	0.10973	0.08027	0.06972	0.07277
	SD	0.06428	0.03412	0.03816	0.11792	0.09269	0.07151	0.06338
Max $\ddot{u}_g$	Min	0.00103	0.00118	0.00118	0.00245	0.00245	0.00118	0.00118
	Max	0.35726	0.09732	0.09732	0.30114	0.30114	0.30114	0.30114
	Mean	0.04883	0.05824	0.05121	0.10440	0.07439	0.06440	0.06661
	SD	0.06199	0.03005	0.03370	0.11204	0.08818	0.06795	0.06008
Min $\ddot{u}_g$	Min	-0.27870	-0.11473	-0.11473	-0.31313	-0.31313	-0.31313	-0.31313
	Max	-0.00095	-0.00119	-0.00119	-0.00233	-0.00233	-0.00119	-0.00112
	Mean	-0.04820	-0.06006	-0.05285	-0.09849	-0.07365	-0.06312	-0.06791
	SD	0.05502	0.03382	0.03695	0.10559	0.08314	0.06391	0.05899
Energy $\|\ddot{u}_g\|^2$	Min	0.00020	0.00016	0.00016	0.00043	0.00043	0.00016	0.00016
	Max	3.22599	1.03466	1.03466	7.31542	7.31542	7.31542	7.31542
	Mean	0.38912	0.43726	0.38280	1.66238	1.03450	0.77436	0.82185
	SD	0.71029	0.34597	0.35425	2.40233	1.86043	1.34532	1.22943

case of training, dataset Train-I, Train-II and Train-III are completely different dataset. Training dataset Train-I is not included in training dataset Train-II, similarly, training dataset Train-I and Train-II are not included in training dataset Train-III. However, training dataset Train-IV includes training dataset Train-III and training dataset Train-V includes training dataset Train-IV as well. The training dataset Train-VI includes all the training data I to V. The training dataset Train-VI is used for the training of DeepONet, DeepONet with Gaussian Normalization, POD-DeepONet, MSDeepONet. The training dataset Train-I to Train-VI are the data used for the discussion in Section 4.2.

C Additional figures

C.1 A typical ground acceleration due to earthquake before and after processing

Fig. 14(a)-(b) show ground acceleration due to earthquake 14383980 before and after processing. The ground acceleration record are recorded with $\delta t = 0.005$ sec. The signal is passed through a butterworth filter. It is also important to note that the length of the signal is 200 second which is much higher than the other signal considered. Thus we have not considered initial 23.56 second acceleration and we have not considered the earthquake after 103.56 sec. The acceleration removed accounts for 0.7% of total energy.

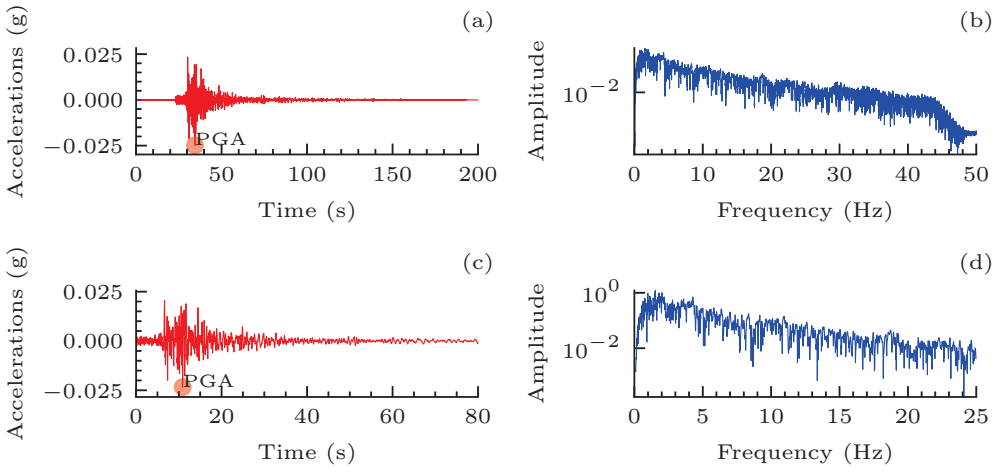

Figure 14: **Ground Acceleration:** The ground acceleration due to 14383980 earthquake recorded at station North Hollywood, 2008 (a) Time history of the acceleration. (b) Frequency spectrum. (c) The resampled acceleration. (d) The frequency spectrum of resampled acceleration.

C.2 Additional results for numerical study for DeepONet

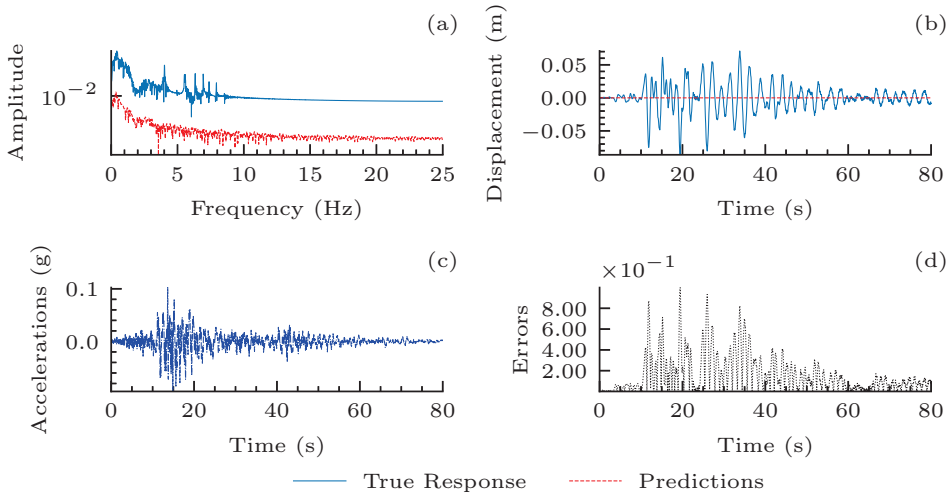

Figure 15: **One of the Training Samples with Predictions of DeepONet:** One of the training samples of DeepONet with Loss Eq. (4.9). (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

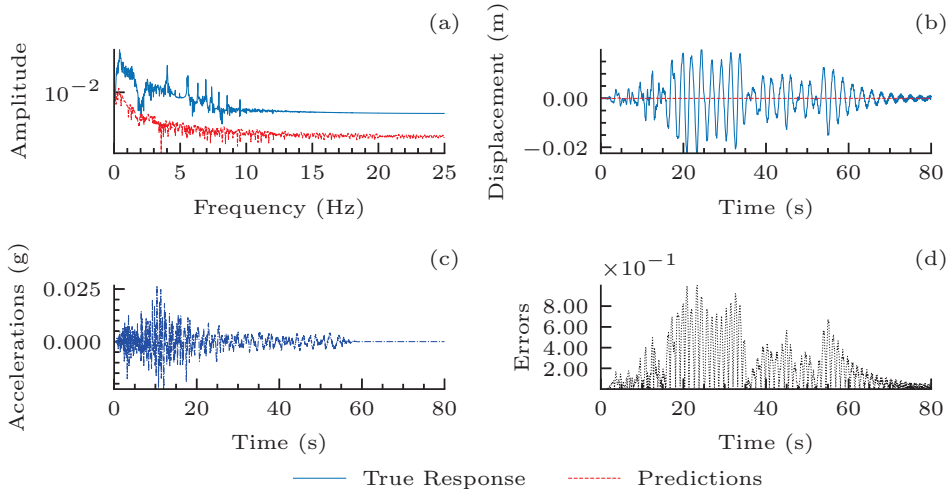

Figure 16: **The Best Case of the Predictions of DeepONet:** The best predictions for the DeepONet for testing cases with Loss Eq. (4.9). (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

C.3 Additional results for numerical study for POD-DeepONet

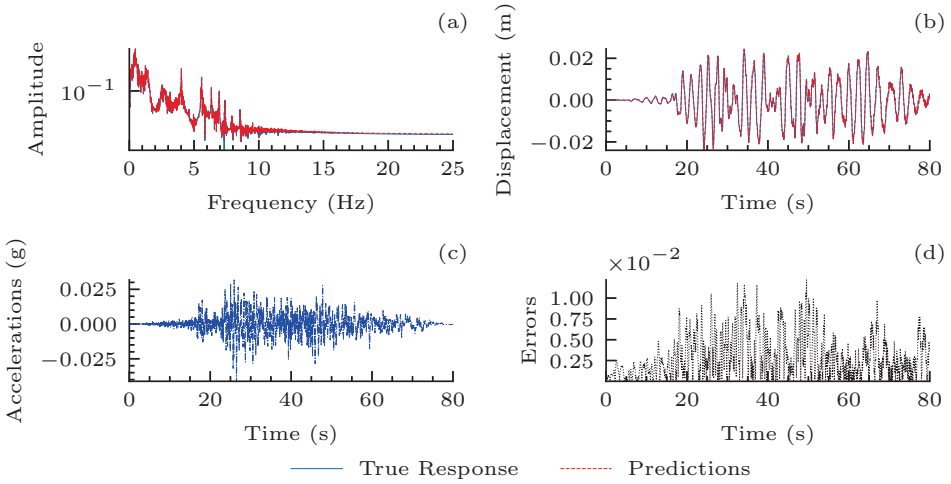

Figure 17: **One of the Training Samples with Predictions of POD-DeepONet:** One of the training samples of POD-DeepONet with Loss Eq. (4.9). (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

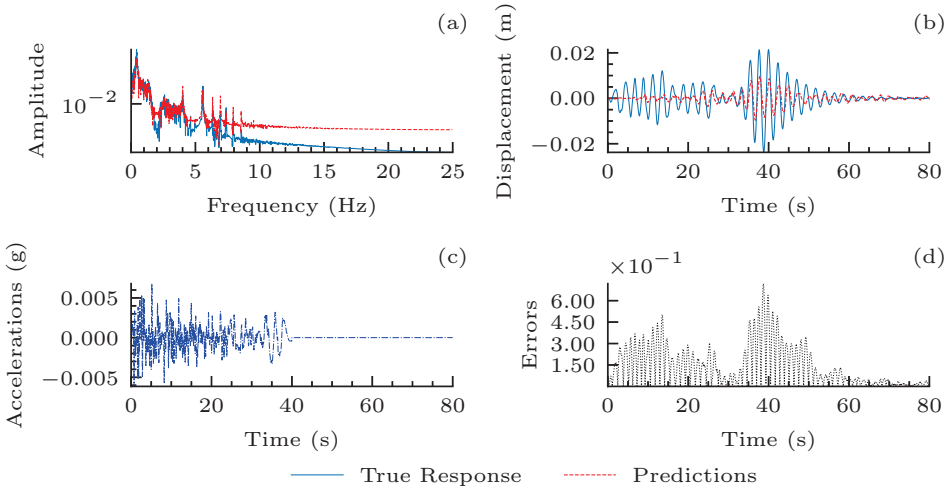

Figure 18: **The Best Case of the Predictions of POD-DeepONet(Relative L_2 Error: 0.76):** The best predictions for the POD-DeepONet for testing cases with Loss Eq. (4.9). (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

C.4 Additional results for numerical study for Multi-scale DeepONet

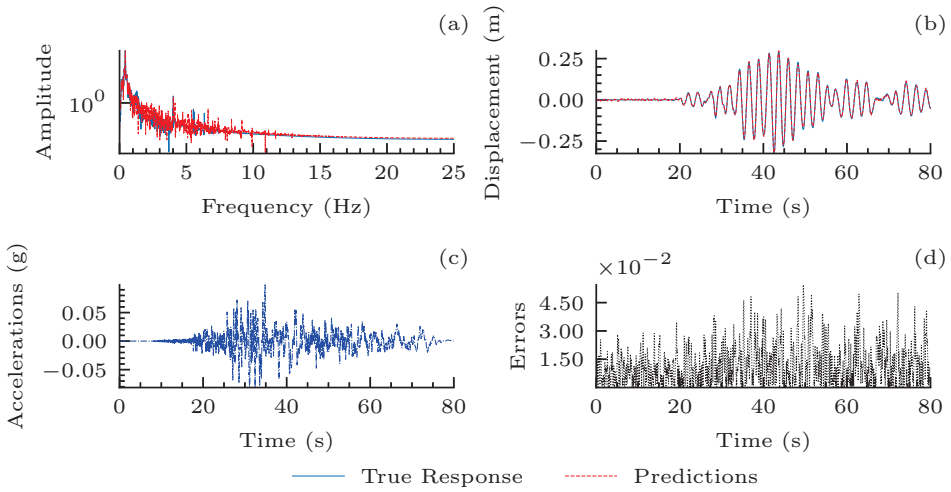

Figure 19: **One of the Training Samples with Predictions of MS-DeepONet:** One of the training samples of MS-DeepONet with Loss Eq. (4.9). (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

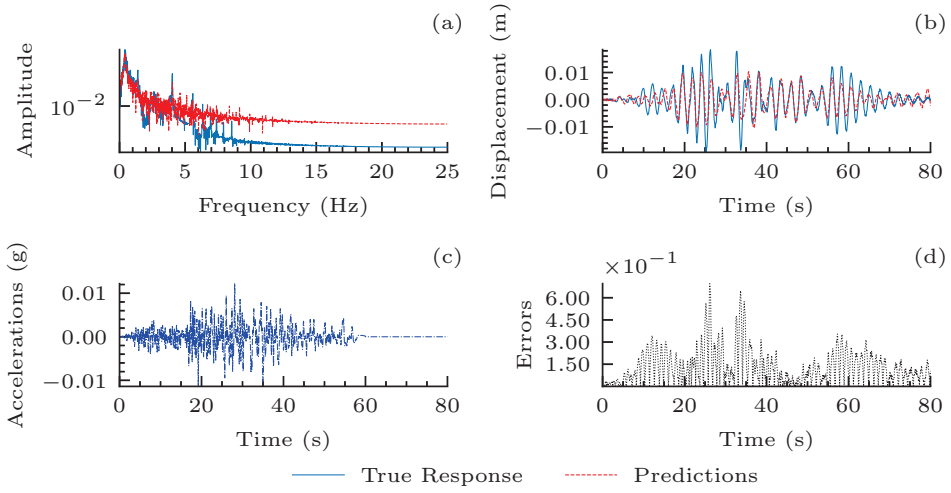

Figure 20: **The Best Case of the Predictions of MS-DeepONet(Relative L_2 Error: 0.59):** The best predictions for the MS-DeepONet for testing cases with Loss Eq. (4.9). (a) The Amplitude of the prediction and true response in Fourier Domain, (b) The prediction and true response, (c) The corresponding input signals(ground acceleration), (d) The relative error Eq. (4.11).

References

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, Aug 2017.

[2] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI Journal Of Computational Mathematics*, 7:121–157, 2021.

[3] Shengze Cai, Zhicheng Wang, Lu Lu, Tamer A. Zaki, and George Em Karniadakis. DeepM&Mnet: Inferring the Electroconvection Multiphysics Fields Based on Operator Approximation by Neural Networks. *J. Comput. Phys.*, 436(C), jul 2021.

[4] Wei Cai, Xiaoguang Li, and Lizuo Liu. A Phase Shift Deep Neural Network for High Frequency Approximation and Wave Problems. *SIAM Journal on Scientific Computing*, 42(5):A3285–A3312, 2020.

[5] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, July 1995.

[6] Anil K. Chopra. *Dynamics of Structures*. Pearson, 4th edition, 2011.

[7] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[8] Beichuan Deng, Yeonjong Shin, Lu Lu, Zhongqiang Zhang, and George Em Karniadakis. Approximation rates of DeepONets for learning operators arising from advection–diffusion equations. *Neural Networks*, 153:411–426, 2022.

[9] P Clark Di Leoni, Lu Lu, Charles Meneveau, George Karniadakis, and Tamer A Zaki. DeepONet prediction of linear instability waves in high-speed boundary layers. *arXiv preprint arXiv:2105.08697*, 2021.

[10] Weinan E and Bing Yu. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Communications in Mathematics and Statistics*, 6:1–12, 2017.

[11] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[12] Xiaohan Fu, Lo-Bin Chang, and Dongbin Xiu. Learning reduced systems via deep neural networks with memory. *Journal of Machine Learning for Modeling and Computing*, 1(2):97–118, 2020.

[13] S. Goswami, M. Yin, Y. Yu, GE., Karniadakis. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*. 2022 Mar 1;391:114587.

[14] Abbavaram Gowtham Reddy. Causality in Neural Networks - An Extended Abstract. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 271–272, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[16] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[18] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.
[19] Gaëtan Kerschen, Keith Worden, Alexander F. Vakakis, and Jean-Claude Golinval. Past, present and future of nonlinear system identification in structural dynamics. *Mechanical Systems and Signal Processing*, 20(3):505–592, 2006.
[20] Hyun-Su Kim. Development of seismic response simulation model for building structures with semi-active control devices using recurrent neural network. *Applied Sciences*, 10(11):3915, 2020.
[21] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2015.
[22] Katiana Kontolati, Somdatta Goswami, Michael D. Shields, George Em Karniadakis, On the influence of over-parameterization in manifold based surrogates and deep neural operators. *Journal of Computational Physics*. 479, 2023, 112008.
[23] Qianxiao Li and Weinan E. Machine learning and dynamical systems. *SIAM News*, 11 2021.
[24] Zhong Li, Jiequn Han, Weinan E, and Qianxiao Li. Approximation and optimization theory for linear continuous-time recurrent neural networks. *Journal of Machine Learning Research*, 23(42):1–85, 2022.
[25] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
[26] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.
[27] Chensen Lin, Zhen Li, Lu Lu, Shengze Cai, Martin Maxey, and George Em Karniadakis. Operator learning for predicting multiscale bubble growth dynamics. *The Journal of Chemical Physics*, 154(10):104118, 2021.
[28] Guochang Lin, Fukai Chen, Pipi Hu, Xiang Chen, Junqing Chen, Jun Wang and Zuoqiang Shi. BI-GreenNet: Learning Green's Functions by Boundary Integral Network. *Commun. Math. Stat.* 11, 103–129 (2023).
[29] Lizuo Liu, Bo Wang, and Wei Cai. Linearized Learning Methods with Multiscale Deep Neural Networks for Stationary Navier-Stokes Equations with Oscillatory Solutions, 2021.
[30] Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-Scale Deep Neural Network (MscaleDNN) for Solving Poisson-Boltzmann Equation in Complex Domains. *Communications in Computational Physics*, 28(5):1970–2001, 2020.
[31] Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal Effect Inference with Deep Latent-Variable Models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6449–6459, Red Hook, NY, USA, 2017. Curran Associates Inc.
[32] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.
[33] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.
[34] Yunan Luo, Jian Peng, and Jianzhu Ma. When causal inference meets deep learning. *Nature*

Machine Intelligence, 2(8):426–427, 2020.

[35] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. Causal Interpretability for Machine Learning - Problems, Methods and Evaluation. *SIGKDD Explor. Newsl.*, 22(1):18–33, may 2020.

[36] Nathan M. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85(3):67–94, 1959.

[37] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio, 2016.

[38] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[40] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[41] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[42] Hoon Sohn, Charles R Farrar, Francois M Hemez, Devin D Shunk, Daniel W Stinemates, Brett R Nadler, and Jerry J Czarnecki. A Review of Structural Health Review of Structural Health Monitoring Literature 1996-2001. 1 2002.

[43] Bo Wang, Wenzhong Zhang, and Wei Cai. Multi-Scale Deep Neural Network (MscaleDNN) Methods for Oscillatory Stokes Flows in Complex Domains. *Communications in Computational Physics*, 28(5):2139–2157, 2020.

[44] Nick Winovich, Karthik Ramani, and Guang Lin. ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. *Journal of Computational Physics*, 394:263–279, 2019.

[45] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *Communications in Computational Physics*, 28(5):1746–1767, 2020.

[46] Ruiyang Zhang, Zhao Chen, Su Chen, Jingwei Zheng, Oral Büyüköztürk, and Hao Sun. Deep long short-term memory networks for nonlinear structural seismic response prediction. *Computers & Structures*, 220:55–68, 2019.

[47] Wenzhong Zhang and Wei Cai. FBSDE based neural network algorithms for high-dimensional quasilinear parabolic PDEs. *Journal of Computational Physics*, 470:111557, 2022.

[48] Minjie Zhu. OpenSeesPy. `https://openseespydoc.readthedocs.io/en/latest/#`, 2015. [Online].

[49] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

[50] Olek C Zienkiewicz, Robert Leroy Taylor, and Jian Z Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005.