# Causal Deep Operator Networks for Data-Driven Modeling of Dynamical Systems

Truong X. Nghiem[1], Thang Nguyen[2], Binh T. Nguyen[2], Linh Nguyen[3]
[1] *School of Informatics, Computing, and Cyber Systems, Northern Arizona University*, Flagstaff, AZ, USA
[2] *Department of Engineering, Texas A&M University–Corpus Christi*, Corpus Christi, TX, USA
[3] *Institute of Innovation, Science and Sustainability, Federation University Australia*, Churchill, VIC, Australia

***Abstract*—The deep operator network (DeepONet) architecture is a promising approach for learning functional operators, that can represent dynamical systems described by ordinary or partial differential equations. However, it has two major limitations, namely its failures to account for initial conditions and to guarantee the temporal causality – a fundamental property of dynamical systems. This paper proposes a novel causal deep operator network (Causal-DeepONet) architecture for incorporating both the initial condition and the temporal causality into data-driven learning of dynamical systems, overcoming the limitations of the original DeepONet approach. This is achieved by adding an independent root network for the initial condition and independent branch networks conditioned, or switched on/off, by time-shifted step functions or sigmoid functions for expressing the temporal causality. The proposed architecture was evaluated and compared with two baseline deep neural network methods and the original DeepONet method on learning the thermal dynamics of a room in a building using real data. It was shown to not only achieve the best overall prediction accuracy but also enhance substantially the accuracy consistency in multistep predictions, which is crucial for predictive control.**

***Index Terms*—Data-driven modeling, causality, neural networks, deep operator networks, multistep predictions.**

## I. Introduction

Neural networks (NNs) are known to be universal approximators of continuous functions, with many applications, such as in computer vision [1], energy systems [2], and dynamical systems modeling [3]. In recent years, there has been a growing interest in applying deep learning to learn functional operators, which map input functions into output functions. As these operators can implicitly represent solutions of ordinary differential equations (ODEs) and partial differential equations (PDEs), often used to describe dynamical systems, the ability to learn (or discover) the operators of complex dynamical systems from data has important implications in data-driven nonlinear dynamics and controls. Different NN architectures have been employed in the literature for learning dynamical system models [4], e.g., recurrent neural networks (RNNs) and neural ODEs, although many are focused on predicting the state evolution of autonomous systems without external inputs.

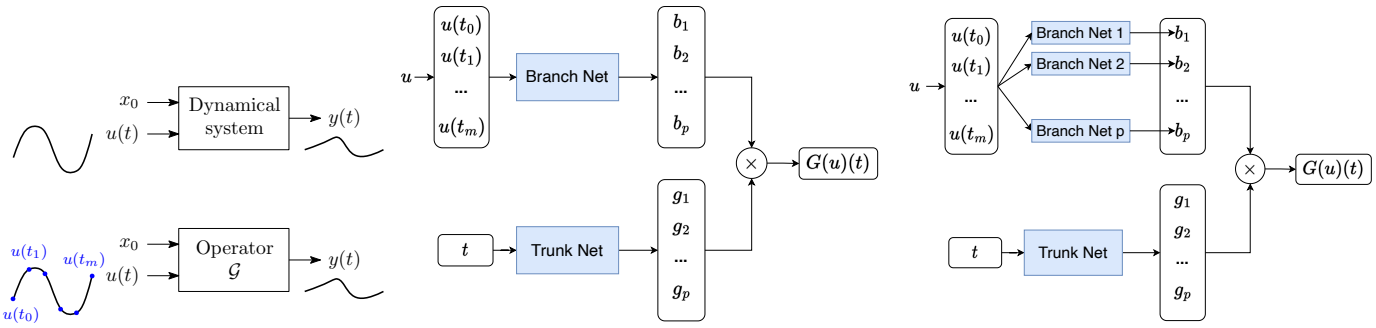
Learning the operator of a dynamical system for predicting its behavior under unseen input signals is crucial for control applications. A promising approach is the deep operator network (DeepONet) architecture, proposed recently by Lu et al. [5]. It has been applied for learning operators in various applications, such as predicting linear instability waves in high-speed boundary layers [6], predicting the power grid's post-fault trajectories [7], learning nonlinear operators in oscillatory function spaces for building seismic wave responses [8], and modeling inelastic scattering probabilities in air mixtures [9].

The original DeepONet architecture, however, has two major limitations when used to model dynamical systems represented by ODEs. Firstly, it does not directly account for the initial conditions of the system, instead often assuming zero initial conditions. This drawback restricts its use in predictive control applications, where the dynamical system model is used to predict the future system behavior starting from varying initial states. Secondly, it does not enforce the temporal causality of the dynamical system, which refers to the property that the state and output of the system at any given time depend on its input only up to that time, and not on any future input. By not guaranteeing the causality, a learned DeepONet model might be temporally acausal, defying the physics of the system and potentially causing inaccuracy and physical inconsistency of the model, which are crucial in some applications such as controls. Some recent works have attempted to address these limitations. For instance, Tan et al. in [10] proposed a modified DeepONet approach that incorporates an independent branch network to account for the initial condition. Nonetheless, their evaluation of the proposed architecture was limited to a small set of regular PDEs and did not consider the case where the initial condition changes. In [11], Causality-DeepONet was proposed to capture the temporal causality of a dynamical system. However, the approach is limited to linear systems and, instead of designing a new architecture, it replaces the input signals of the branch network by a zero-padding signals with a shifting window to express the causality.

We propose a *causal deep operator network* (Causal-DeepONet) architecture for data-driven learning of dynamical systems. Causal-DeepONet overcomes the limitations of the original DeepONet by directly accounting for the initial condition and flexibly enforcing the temporal causality through a novel network architecture design, which is our **main contribution** and is illustrated later in Fig. 2. The architecture incorporates an independent root network for the initial condition and independent branch networks conditioned,

This material is based upon work supported by the National Science Foundation under Grant No. 2138388 and Grant No. 2238296.

The authors would like to thank Tung Nguyen at Northern Arizona University for his help in a part of the experiment.




(a) Operator for a dynamical system. (b) Unstacked DeepONet architecture. (c) Stacked DeepONet architecture.

Fig. 1. The deep operator network (DeepONet) approach: (a) A dynamical system is an operator that maps an input function (discretized in the case of DeepONet) to an output function; (b) and (c) are the unstacked and stacked DeepONet architectures.

or switched on/off, by time-shifted step functions or sigmoid functions to express the temporal causality. We evaluated the effectiveness and demonstrated the benefits of Causal-DeepONet, compared with two baseline NN methods and the original DeepONet method, in learning the thermal dynamics of a room in a real-life building, using real data. It was shown to not only achieve the best overall prediction accuracy but also enhance substantially the accuracy consistency in multistep predictions, which is crucial for predictive control. Our approach belongs to the class of physics-informed machine learning (PIML) methods, that seamlessly integrate known physical properties of a system (the temporal causality in our case) into data-driven modeling of the system for enhancing the model's fidelity, data efficiency, interpretability, robustness, and reliability [12]. PIML has been demonstrated in the literature to improve significantly data-driven modeling of complex dynamical systems, including heating, ventilation, and air conditioning (HVAC) systems [13], [14].

The rest of the paper is organized as follows. Section II briefly reviews the original DeepONet architecture. Section III describes in detail our proposed Causal-DeepONet architecture. Section IV presents and discusses our experiments for evaluating and demonstrating the effectiveness of Causal-DeepONet. Finally, Section V summarizes the conclusions of this work and outlines potential future work.

## II. Deep Operator Networks (DeepONets)

This section provides a brief overview of deep operator networks (DeepONets), an approach for operator learning using deep neural networks (DNNs). While the DeepONet approach is broadly applicable to learning any nonlinear operators, we will focus on learning dynamical systems from data. More details can be found in [5].

Consider a dynamical system represented by the ODE

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \quad x(t_0) = x_0 \\ y(t) &= g(x(t), u(t)) \end{aligned} \tag{1}$$

with independent time variable $t \geq t_0$, input variable $u \in \mathbb{R}^{n_u}$, state variable $x \in \mathbb{R}^{n_x}$, and output variable $y \in \mathbb{R}^{n_y}$. Here, $\dot{x}(t)$ denotes the time derivative of $x(t)$, i.e., $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$. Given an initial state $x_0$ and an input function (or signal) $u(t)$, the dynamical system (1) produces an output response function (or signal) $y(t)$ that satisfies the above ODE. Therefore, the dynamical system defines a nonlinear operator $\mathcal{G}$ that maps an input function $u(t)$ to an output function $y(t)$, given an initial state $x_0$, i.e., $\mathcal{G} : (x_0, u(t)) \mapsto y(t)$.

The dynamical system (1) exhibits two important properties. Firstly, its behavior depends on the initial state $x_0$. Secondly, there is an inherent *temporal causality* from its input to its output, which refers to the property that the state and output of the system at any given time depend on its input only up to that time, and not on any future input. Mathematically, $x(t)$ and $y(t)$ at any time $t \geq t_0$ depend on $x_0$ and $u(\tau)$ for $\tau \in [t_0, t]$ but not for $\tau > t$.

The operator $\mathcal{G}$ can be learned from data by a machine learning (ML) model. In [5], Lu et al. proposed a DNN architecture called DeepONet for learning nonlinear operators like $\mathcal{G}$. A DeepONet will learn the dynamical system (1) as an operator $G$ that maps an input function $u(t)$ to an output function $G(u)(t)$ in a time duration $t_0 \leq t \leq T$, for a finite horizon $T$. Note that the initial state $x(t_0)$ is not explicitly taken into account by the DeepONet operator $G$, unlike the dynamical system operator $\mathcal{G}$. Because a DNN cannot directly take a continuous-time function as an input, the input function $u(t)$ is discretized at a finite set of $m+1$ discrete time instants $t_0 < t_1 < \cdots < t_m = T$ (these time instants are referred to as "sensors" in [5]). Thus, the function $u(t)$ is represented by a finite number of inputs to the DeepONet model as $u(t_0), u(t_1), \ldots, u(t_m)$, where each $u(t_i)$ is the system input vector at time $t_i$. This core idea is illustrated in Figure 1a. The DeepONet takes another input $t$ and produces the predicted system output at time $t$ as $\tilde{y}(t) = G(u)(t)$, where $\tilde{y}$ is the predicted value. For simplicity, it is assumed that the system output is scalar, i.e., $n_y = 1$, although the DeepONet approach can be extended to a vector system output.

Two DeepONet architectures are proposed in [5]. The unstacked architecture, shown in Figure 1b, uses a single NN called the *branch network* to transform the inputs $u(t_0), u(t_1), \ldots, u(t_m)$ into a vector $b \in \mathbb{R}^p$, which captures the effect of the input function $u(t)$ on the output. Another

NN called the *trunk network* takes the scalar time value $t \in \mathbb{R}$ to generate a vector $g \in \mathbb{R}^p$. The predicted output at time $t$ is then calculated by merging the two vectors as

$$G(u)(t) = \tilde{y}(t) = \sum_{k=1}^{p} \underbrace{b_k(u(t_0), u(t_1), \dots, u(t_m))}_{\text{branch}} \underbrace{g_k(t)}_{\text{trunk}}.$$

The stacked architecture, shown in Figure 1c, is a variation of the unstacked architecture, where $p$ independent branch networks generate the $p$ elements of $b$. The branch and trunk networks can be any types of NNs, such as fully connected neural networks and convolutional neural networks. It was proved in [5] that the proposed DeepONets are universal approximators of operators by the Generalized Universal Approximation Theorem for Operator.

While the DeepONet approach has been demonstrated to be a powerful and promising tool for data-driven scientific discovery and for modeling complex systems as operators, it has **two major limitations** when used to model dynamical systems such as (1). Firstly, it does not directly take the initial state $x_0$ of the system into account, whereas the evolution of system (1) depends on $x_0$. More importantly, the DeepONet architecture does not guarantee the temporal causality of the dynamical system, causing a possibility that the learned model might be temporally acausal, which defies the physics of the system. This can be seen by noticing that the input $u(t_i)$ at any time instant $t_i$ can affect the vector $b$ independently of the time $t$, hence a future input value $u(t_i)$ at $t_i > t$ can potentially affect the output $y(t)$ at time $t$, violating the system's temporal causality. These limitations may affect the model accuracy and physical consistency, which are crucial in some applications such as control systems. To overcome these limitations, in the next section, we propose a novel approach based on DeepONet for learning dynamical systems from data.

## III. Causal Deep Operator Networks (Causal-DeepONets)

We propose a novel *causal deep operator network* (Causal-DeepONet) architecture for data-driven learning of dynamical systems to address the two limitations of the vanilla DeepONet approach. The architecture of the proposed Causal-DeepONet is illustrated in Figure 2 and described in this section.

To account for the initial state $x_0$, we introduce a new network called the *root network*, which takes $x_0$ as the input and produces a vector $h \in \mathbb{R}^p$. The values of $h_i(x_0)$, $i = 1, \dots, p$, are used in the output expression to capture the effect of the initial state on the operator's output.

To enforce the temporal causality, the proposed Causal-DeepONet architecture in Figure 2 employs two improvements.

- Firstly, we use individual branch networks `Branch net` $i$, for $i = 0, \dots, m$, corresponding to each input $u(t_i)$, to generate individual vectors $c_i \in \mathbb{R}^q$ of dimension $q$, so that each $c_i$ only depends on $u(t_i)$. This is to ensure that the vectors $c_i$ follow the same temporal order as the discrete-time inputs $u(t_i)$. The concatenated vector $c$ of all $c_i$ is a vector of dimension $p$ as in the original DeepONet, hence $p$ and $q$ must be selected so that $p = (m+1)q$.
- Secondly, to ensure the temporal causality of the effect of the vectors $c_i(u(t_i))$ on the output, we design the Causal-DeepONet to "turn off" all $c_i$ for $t_i$ in the future of the time $t$. This is achieved by multiplying each $c_i$ with a function $\sigma_i(t) \in [0, 1]$, which has the property that $\sigma_i(t) = 0$ when $t < t_i$ and $\sigma_i(t) = 1$ when $t \geq t_i$. In other words, $\sigma_i(t)$ is a step function shifted by the time duration $t_i$. With this definition, the product $c_i(u(t_i))\sigma_i(t)$ satisfies

$$c_i(u(t_i))\sigma_i(t) = \begin{cases} 0 & \text{if } t < t_i \\ c_i(u(t_i)) & \text{if } t \geq t_i \end{cases}.$$

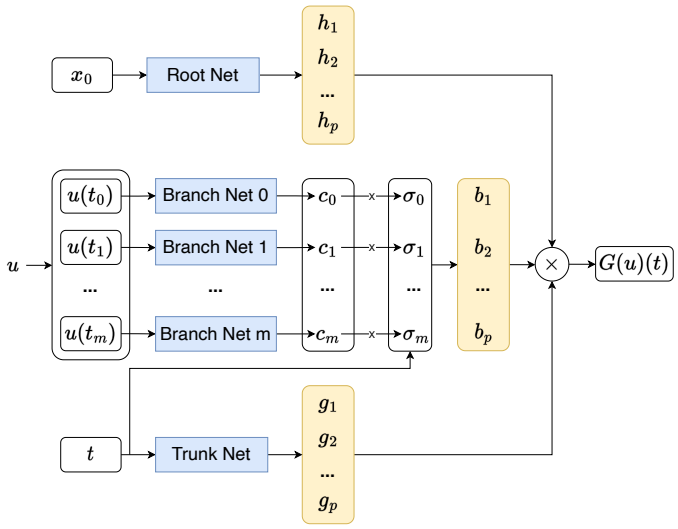



Fig. 2. The proposed Causal-DeepONet architecture for modeling a dynamical system. It takes into account the initial state $x_0$ by using the root network and captures the temporal causality of the system by using independent branch networks and time-shifted functions $\sigma_i(t)$.

The products $c_i(u(t_i))\sigma_i(t)$ are concatenated to produce a vector $b \in \mathbb{R}^p$ of the same dimension $p$ as the vectors $h$ and $g$. The sum of the elements of the element-wise products of $h$, $b$, and $g$ is the predicted output $\tilde{y}(t) = G(u)(t)$:

$$\tilde{y}(t) = \sum_{k=1}^{p} h_k(x_0) b_k(t, u(t_0), u(t_1), \dots, u(t_m)) g_k(t). \quad (2)$$

It is important to note here that because the term $b_k(t, u(t_0), u(t_1), \dots, u(t_m))$, by construction, adheres to the temporal causality of the dynamical system, the predicted output of the Causal-DeepONet given by (2) also satisfies the temporal causality property.

Choosing the time-shifted step functions for $\sigma_i(t)$, as described above, can represent strictly the temporal causality from the input to the output. However, if smooth functions are preferred for $\sigma_i(t)$, the requirement of $\sigma_i(t)$ can be relaxed as $\sigma_i(t) \approx 0$ when $t < t_i$ and $\sigma_i(t) \approx 1$ when $t \geq t_i$, and time-shifted sigmoid functions defined below can be used instead

$$\sigma_i(t) = \text{sigmoid}(\alpha(t - t_i)) \quad (3)$$

where $\alpha$ is a positive constant. The larger $\alpha$ is, the more strictly $\sigma_i(t)$ enforces the temporal causality.

A Causal-DeepONet model can be trained from time-series data of a dynamical system, obtained from experiments or simulations of the system. The learned model can be used to predict the output signal of the system at multiple future time instants at once from a known initial state $x_0$ and an input signal given by discrete-time values $u(t_0), u(t_1), \ldots, u(t_m)$. The multistep prediction capability of Causal-DeepONet is particularly useful for control techniques such as the receding horizon control (RHC) scheme for model predictive control (MPC) [15]. This scheme requires making predictions of the state or output of a dynamical system over multiple future time steps from the current (measured or observed) state.

## IV. Experiments and Discussion

To evaluate the proposed Causal-DeepONet approach and demonstrate its benefits in learning dynamical systems from data, we conducted experiments on a real-world heating, ventilation, and air conditioning (HVAC) system. This section will detail the experiments and discuss the results.

### A. Experimental System and Data

We used actual data from the HVAC system of the building of the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. Data from a laboratory room were collected during the heating season in November 2022 for the study. This room is at a corner of the building's first floor, with two large adjacent walls and windows to the outside, hence the outside weather can directly impact the room's environment.

The goal of the experiment was to develop a data-driven model of the room's temperature dynamics from the HVAC data. Following the convention of modeling HVAC systems [16], we assume that the room's air is fully mixed and define $T_z$ as the room's air temperature. Let $m_{sa}$ and $T_{sa}$ be respectively the mass flow rate and temperature of the supply air into the room, and $T_{oa}$ be the outside air temperature, which is a disturbance to the room's thermal dynamics. This laboratory often has low occupancy and low usage, thus the internal heat disturbance has negligible impact on the room's temperature. The dynamical model of the room's temperature can be expressed as $\dot{x}(t) = f(x(t), u(t))$ and $y(t) = x(t)$, with state $x = T_z$ and input $u = [m_{sa}, T_{sa}, T_{oa}]^\top$. The function $f(x, u)$ can be developed using the physical principles of the thermal dynamics of the room [16] or learned directly from data using ML techniques. Our study was focused on the latter approach using NNs.

To obtain data for model training and validation, an experiment was conducted for 11 days by uniformly and randomly changing the thermostat set-point of the room between $68\,^\circ$F to $74\,^\circ$F every 30 minutes. The set-points were changed programmatically using Python and a BACnet (building automation and control network) interface. The acquired dataset for each room was a set of time-series data, which included the room temperature $T_z$, supply air temperature $T_{sa}$, supply airflow rate $m_{sa}$, and outside temperature $T_{oa}$, measured at 5-minute intervals. Each dataset contained 3121 data points of the form $\{T_z(t_k), T_{sa}(t_k), m_{sa}(t_k), T_{oa}(t_k)\}_{k=1}^{3121}$ at consecutive sampling time steps $t_k$.

### B. Modeling Methods

For comparison purposes, several NN-based methods were used to learn data-driven models of the room's temperature dynamics, in addition to the proposed Causal-DeepONet method. Inspired by our interest to predict the room's temperature over several future time steps, e.g., to be used in MPC, each model was developed to give a prediction function of the general form

$$x(t_{k+\tau}) = \mathcal{M}(x(t_k), u(t_k), u(t_{k+1}), \ldots u(t_{k+m}), t_{k+\tau})$$

for any $\tau \in \{1, \ldots, m\}$. Here, $t_{k+i}$ is the $i$-th sampling time instant after $t_k$, i.e., $t_{k+i} = t_k + \delta i$, for a given sampling time step $\delta$ ($\delta = 5$ minutes in our study). Given the measured temperature $x(t_k)$ (i.e., the initial state) and the sequence of current and future inputs $\{u(t_k), u(t_{k+1}), \ldots u(t_{k+m})\}$ (i.e., the discretized input signal over the duration $[t_k, t_{k+m}]$), the function $\mathcal{M}$ can predict the temperature $x(t_{k+\tau})$ at any future time instant $t_{k+\tau}$ within that duration.

Our study evaluated two baseline NN methods, the original DeepONet method, and our proposed Causal-DeepONet method on the same modeling task, as described below.

- **Multi-output DNN** (baseline, non-causal): This method uses a conventional DNN that takes $\{x(t_k), u(t_k), u(t_{k+1}), \ldots u(t_{k+m})\}$ as its inputs and produces $m$ outputs, which are the predictions $\{x(t_{k+1}), \ldots x(t_{k+m})\}$ at all the future sampling time instants in one forward computation pass. The model is temporally non-causal because it has no mechanism to enforce the temporal causality from the input values to the outputs at different times. DNN models were trained with various numbers of hidden layers (from 1 to 3) and different numbers of neurons (4, 8, 12, or 16 neurons for each hidden layer), and the most accurate model on the test dataset was selected.
- **Recurrent neural network (RNN)** (baseline, causal): This method trains a RNN that takes the current state $x(t_i)$ and input $u(t_i)$ to predict the next state $x(t_{i+1})$, recursively rolled out over $m$ sampling time steps. At the current time step $t_k$, the model uses $x(t_k)$ and $u(t_k)$ to predict $x(t_{k+1})$. At $t_{k+1}$, it uses the predicted state $x(t_{k+1})$ and the input $u(t_{k+1})$ to predict $x(t_{k+2})$. This continues until the predicted $x(t_{k+m})$ is obtained. The model is temporally causal because, by construction, the predicted output at a time step is not affected by future inputs. It is expected that the prediction accuracy of a RNN model will deteriorate further into the future due to the propagation of prediction uncertainty. RNN models were trained with various numbers of hidden layers (from 1 to 3) and different numbers of neurons (4, 8, 12, or 16 neurons for each hidden layer), and the most accurate model on the test dataset was selected.
- **DeepONet** (non-causal): The original DeepONet method, as proposed in [5], was used to train DeepONet models that take $\{u(t_k), u(t_{k+1}), \ldots u(t_{k+m}), t\}$ to produce the

TABLE I
$R^2$ SCORES FOR EACH MODEL AT 4 FUTURE TIME STEPS.

| | $R_1^2$ | $R_2^2$ | $R_3^2$ | $R_4^2$ |
|---|---|---|---|---|
| **Multi-output DNN** | 0.948 | 0.944 | 0.940 | 0.941 |
| **RNN** | 0.974 | 0.946 | 0.913 | 0.893 |
| **DeepONet** | 0.969 | 0.966 | 0.947 | 0.942 |
| **Causal-DeepONet** | **0.992** | **0.987** | **0.982** | **0.978** |

TABLE II
MSEs FOR EACH MODEL AT 4 FUTURE TIME STEPS.

| | $\text{MSE}_1$ | $\text{MSE}_2$ | $\text{MSE}_3$ | $\text{MSE}_4$ |
|---|---|---|---|---|
| **Multi-output DNN** | 0.161 | 0.175 | 0.189 | 0.181 |
| **RNN** | 0.081 | 0.169 | 0.271 | 0.332 |
| **DeepONet** | 0.097 | 0.102 | 0.158 | 0.183 |
| **Causal-DeepONet** | **0.021** | **0.039** | **0.056** | **0.069** |

predicted state $x(t)$, for $t_k \leq t \leq t_{k+m}$. It is important to note that the original DeepONet architectures do not take the initial state $x(t_k)$ into account. The model is temporally non-causal, as explained in Section II. Multiple DeepONet models were trained, whose numbers of hidden layers and neurons per hidden layer for the branch and trunk networks were varied similarly to the multi-output DNN and RNN methods. The most accurate DeepONet model on the test dataset was selected.

- **Causal-DeepONet** (causal): The proposed Causal-DeepONet method was used to train models that take $\{x(t_k), u(t_k), u(t_{k+1}), \dots u(t_{k+m}), t\}$ (note the inclusion of the initial state $x(t_k)$) to produce the predicted state $x(t)$, for $t_k \leq t \leq t_{k+m}$. The model is temporally causal by design. Again, multiple Causal-DeepONet models were trained with varying hyperparameters, and the most accurate model on the test dataset was selected.

We remark that, while the two baseline methods can predict the output at only discrete sampling time steps $t_{k+i}$, the two operator-based methods, in principle, can predict the output at any continuous time $t$ in the horizon $[t_k, t_{k+m}]$. However, in this study, we evaluated all the models at only the sampling time steps $t_{k+1}, \dots, t_{k+m}$.

### C. Results and Discussion

In this study, the horizon length $m$ was set to 4, i.e., the models predict 4 time steps into the future. Each data sample for training the models, therefore, consists of $m+1$ consecutive state and input values in addition to the initial state, that is $\{x(t_k), x(t_{k+1}), \dots, x(t_{k+m}), u(t_k), u(t_{k+1}), \dots, u(t_{k+m})\}$. To demonstrate the data efficiency and effectiveness of different modeling methods, we chose a relatively small training dataset of 128 samples from the beginning of the experimental dataset and the test dataset of the remaining samples. All models were trained and evaluated on the same training and test datasets. To evaluate the performance of a

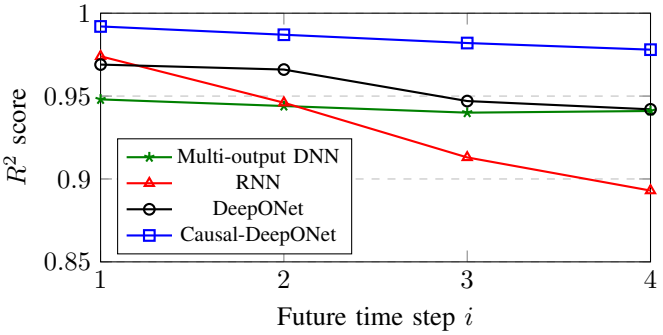



Fig. 3. Change of $R^2$ score over future steps for each model. Higher is better.

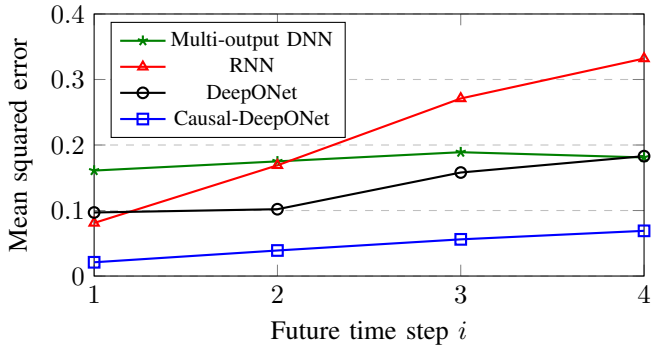



Fig. 4. Change of MSE over future steps for each model. Lower is better.

model, the coefficient of determination ($R^2$ score) and the mean squared error (MSE) were calculated for each prediction of $x(t_{k+i})$, with $i$ ranging from 1 to $m$. The step $i$ is indicated by the subscript of each metric value, e.g., $R_1^2$ and $\text{MSE}_1$ are the $R^2$ score and MSE of $x(t_{k+1})$.

The results of the best model for each method are presented in Table I and Table II, which compare the $R^2$ scores and MSEs, respectively, of the different modeling methods. The best accuracy results are emphasized in bold. Figures 3 and 4 illustrate the changes of the $R^2$ scores and MSEs for each model over the future time steps $t_{k+i}$.

The following observations can be made of the results.

- The baseline multi-output DNN (dark green lines) expectedly achieved relatively consistent prediction accuracy over the future time steps, due to its ability to learn and predict all the future outputs in one forward computation pass. However, it is the least accurate model in the first two time steps and the second least accurate model in the last two time steps. This can be explained by its lack of temporal causality and its low data efficiency, which is impacted by the number of outputs (requiring a larger network size) and the small training dataset.
- The baseline RNN (red lines) achieved high accuracy at the first time step, but its accuracy deteriorated quickly over time, becoming the least accurate model in the last two time steps. This is expected due to prediction uncertainty propagation, which was explained above.

TABLE III
RELATIVE QUALITIES OF THE METHODS.

| | Overall Accuracy | Accuracy Consistency |
|---|---|---|
| **Multi-output DNN** | Low | High |
| **RNN** | Medium | Low |
| **DeepONet** | Medium | Medium |
| **Causal-DeepONet** | **High** | **High** |

- The original DeepONet (black lines) achieved good prediction accuracy at all time steps and was the second most accurate model overall. Its accuracy, however, decayed relatively quickly over time; for example, while it started out noticeably more accurate than the multi-output DNN model ($0.969$ versus $0.948$ in $R^2$ score), it became just as accurate as that model at the last time step ($0.942$ versus $0.941$ in $R^2$ score).
- The proposed Causal-DeepONet (blue lines) achieved the best accuracy at every time step and was substantially more accurate than all the other models. It also maintained its accuracy consistently over time; in fact, its accuracy at the last time step ($R^2 = 0.978$) was still better than the best accuracy at the first time step of all the other models ($R^2 = 0.974$).

It is evident from the results that **our proposed Causal-DeepONet method outperformed the baseline methods and the original DeepONet method** for modeling dynamical systems from data by a large margin. Its superior performance can be explained by its integration of the initial state and especially the temporal causality of the system. Table III summarizes the relative qualities of the methods in terms of the overall accuracy and the consistency of accuracy over time.

## V. CONCLUSION

We proposed a novel deep operator network (DeepONet) architecture for learning data-driven models of dynamical systems. The method, called Causal-DeepONet, improves upon the original DeepONet architecture by taking into account the initial state and enforcing the temporal causality of the system. Our method was evaluated against a baseline multi-output DNN method, a baseline RNN method, and the original DeepONet method on learning the thermal dynamics of a room in a building using real HVAC data. The experimental results showed that the proposed Causal-DeepONet method not only improves model accuracy significantly but also achieves highly consistent accuracy in multi-step prediction tasks. The latter quality will enable the use of Causal-DeepONet in applications that require accurate predictions in multiple future time steps, such as predictive control of complex dynamical systems. By incorporating the inherent temporal causality property, our method provides a foundation for the development of more accurate and robust data-driven models of dynamical systems.

The findings of our study have important implications for future research that we plan to pursue, including investigating the application of Causal-DeepONet to other complex physical systems and exploring the use of Causal-DeepONet models in predictive control applications.

## REFERENCES


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[2] Z. Wang, T. Hong, and M. A. Piette, "Building thermal load prediction through shallow machine learning and deep learning," *Applied Energy*, vol. 263, p. 114683, Apr. 2020.

[3] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Neural Networks with Physics-Informed Architectures and Constraints for Dynamical Systems Modeling," in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*. PMLR, May 2022, pp. 263–277.

[4] C. Legaard, T. Schranz, G. Schweiger, J. Drgoňa, B. Falay, C. Gomes, A. Iosifidis, M. Abkar, and P. Larsen, "Constructing Neural Network Based Models for Simulating Dynamical Systems," *ACM Computing Surveys*, vol. 55, no. 11, pp. 236:1–236:34, Feb. 2023.

[5] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, Mar. 2021.

[6] P. C. D. Leoni, L. Lu, C. Meneveau, G. E. Karniadakis, and T. A. Zaki, "Neural operator prediction of linear instability waves in high-speed boundary layers," *Journal of Computational Physics*, vol. 474, p. 111793, Feb. 2023.

[7] C. Yue, K. Zhang, D. Wang, C. Kang, X. Ma, Y. Yu, and R. Wang, "Deeponet-grid-uq: A trustworthy deep operator framework for predicting the power grid's post-fault trajectories," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 3615–3628, 2021.

[8] X. Zhao, C. Kang, X. Li, and X. Ma, "Multiscale deeponet for nonlinear operators in oscillatory function spaces for building seismic wave responses," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 9, pp. 3589–3600, 2021.

[9] M. S. Priyadarshini, S. Venturi, and M. Panesi, "Application of DeepOnet to model inelastic scattering probabilities in air mixtures," in *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021.

[10] X. Yin, X. Li, C. Kang, and X. Ma, "Enhanced deeponet for modeling partial differential operators considering multiple input functions," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4248–4259, 2021.

[11] L. Liu, K. Nath, and W. Cai, "A Causality-DeepONet for Causal Responses of Linear Dynamical Systems," Sept. 2022.

[12] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, May 2021.

[13] J. Drgoňa, A. R. Tuor, V. Chandan, and D. L. Vrabie, "Physics-constrained deep learning of multi-zone building thermal dynamics," *Energy and Buildings*, vol. 243, July 2021.

[14] T. L. Nguyen and T. X. Nghiem, "A Comparative Study of Physics-Informed Machine Learning Methods for Modeling HVAC Systems," in *Under review at IEEE Conference on Control Technology and Applications (CCTA) 2023*, 2023.

[15] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Madison, Wisconsin: Nob Hill Publishing, LLC, Oct. 2017.

[16] Z. Afroz, GM. Shafiullah, T. Urmee, and G. Higgins, "Modeling techniques used in building HVAC control systems: A review," *Renewable and Sustainable Energy Reviews*, vol. 83, pp. 64–84, Mar. 2018.