LEARNING FINE SCALE DYNAMICS FROM COARSE OBSERVATIONS VIA INNER RECURRENCE

Victor Churchill* & Dongbin Xiu

Department of Mathematics, The Ohio State University, Columbus, Ohio 43210, USA

*Address all correspondence to: Victor Churchill, Department of Mathematics, The Ohio State University, Columbus, Ohio 43210, USA; Tel.: +1 614 292 4975; Fax: +1 614 292 1479, E-mail: churchill.77@osu

Original Manuscript Submitted: 6/3/2022; Final Draft Received: 7/31/2022

Recent work has focused on data-driven learning of the evolution of unknown systems via deep neural networks (DNNs), with the goal of conducting long-term prediction of the dynamics of the unknown system. In many real-world applications, data from time-dependent systems are often collected on a time scale that is coarser than desired, due to various restrictions during the data acquisition process. Consequently, the observed dynamics can be severely undersampled and do not reflect the true dynamics of the underlying system. This paper presents a computational technique to learn the fine-scale dynamics from such coarsely observed data. The method employs inner recurrence of a DNN to recover the fine-scale evolution operator of the underlying system. In addition to mathematical justification, several challenging numerical examples, including unknown systems of both ordinary and partial differential equations, are presented to demonstrate the effectiveness of the proposed method.

KEY WORDS: deep neural networks, flow map approximation, coarse time sampling

1. INTRODUCTION

Data-driven modeling and learning of unknown dynamical systems, including both ordinary and partial differential equations, has been a prominent area of research over the past few years. One approach to this problem, known as governing equation discovery, constructs a mapping from the state variables of the unknown system to their time derivatives. The equation itself is modeled using sparse approximation from a large dictionary of potential right-hand-side functions. In certain circumstances, exact equation recovery is possible. See, for example Brunton et al. (2016) and the related work in recovering both ODEs (Kang et al., 2019; Schaeffer and McCalla, 2017; Schaeffer et al., 2018; Tran and Ward, 2017) and PDEs (Rudy et al., 2017; Schaeffer, 2017).

More recently, deep neural networks (DNNs) have also been used to construct this mapping. See, for example Lu et al. (2021b), Qin et al. (2019), Raissi et al. (2018), and Rudy et al. (2019) for ODE modeling and Long et al. (2018b), Lu et al. (2021a), Raissi et al. (2017a,b), Raissi (2018), and Sun et al. (2019) for PDE modeling.

An alternative approach, known as flow map or evolution operator discovery, seeks to model the unknown system by constructing an approximation to its flow map over a short time (Qin
Although the method does not directly recover the unknown governing equations, it produces an accurate predictive model, once an accurate flow map approximation is constructed, so that any new initial condition can be accurately marched forward in time for long-term prediction. One advantage of flow map based learning is that it does not require approximations of temporal derivatives, which are prone to larger numerical errors. This approach uses DNNs, in particular, residual networks [ResNet (He et al., 2016)], as nonlinear approximators to model the flow map from data. Introduced to model autonomous systems in Qin et al. (2019), this general framework of flow map based DNN learning has been extended to model nonautonomous systems (Qin et al., 2021), parametric dynamical systems (Qin et al., 2020), partially observed dynamical systems (Fu et al., 2020), chaotic systems such as Lorenz 63 and 96 (Churchill and Xiu, 2022), and PDEs in modal space (Wu and Xiu, 2020), and nodal space (Chen et al., 2022). Adjustments have also been proposed to improve robustness through ensemble averaged learning (Churchill et al., 2022). Relevant components of this framework are reviewed below.

As the field of unknown system learning matures, and examples start to be drawn from real-world applications rather than synthetic benchmarking problems, the methodology needs to keep pace with circumstances and limitations of data collection. One such scenario is that data are frequently collected on a coarser timescale than desired, which makes them appear discontinuous or discrete and obfuscates the underlying smooth fine time scale dynamics of the system. For example, storage or equipment limitations can restrict collection of certain climate data to daily or hourly intervals, while in reality the underlying continuous dynamics would be better understood by collecting over much shorter intervals, e.g., every minute or even more frequently. It is then natural to wonder if it is possible to construct an accurate model for the smooth unknown dynamics over a much shorter timescale using the coarsely observed data. In this paper, we answer this question affirmatively. Hence, the chief contribution of this paper is a systematic and rigorous examination of data-driven flow map approximation of unknown systems which are coarsely observed in time using an approachable and mathematically grounded DNN framework, a topic that (to the best of our knowledge) has not been explored in the literature. In particular, we propose using the concept of inner recurrence to require the network approximation of the flow map to operate on a finer time scale than the observations. This work builds upon the existing DNN framework for evolution discovery and opens up the possibility of addressing a more complex and broader class of problems. After presenting the DNN structure and training specifics, along with its mathematical justification, we present six challenging ODE and PDE learning problems to demonstrate the effectiveness of the method.

2. FLOW MAP MODELING OF UNKNOWN SYSTEMS

We are interested in constructing accurate approximate models for the evolution laws behind dynamical data. Throughout this paper our discussion will be on unknown systems observed at discrete time instances with a constant time step \( \Delta \),

\[
\begin{align*}
t_0 < t_1 < \cdots, & \quad t_{n+1} - t_n = \Delta, \\
\forall n. & 
\end{align*}
\]

(1)

Generality is not lost with the constant time step assumption. We will also use a subscript to denote the time variable of a function, e.g., \( x_n = x(t_n) \). Some parts of this review closely follow that of Churchill et al. (2022).

Consider an unknown autonomous system

\[
\frac{dx}{dt} = f(x), \quad x \in \mathbb{R}^d,
\]

(2)
where $f: \mathbb{R}^d \rightarrow \mathbb{R}^d$ is unknown. Because the system is autonomous, its flow map depends only on the time difference but not the actual time, i.e., $x_n = \Phi_{t_n-t_s}(x_s)$. Thus, the solution over one time step satisfies

$$x_{n+1} = \Phi_{\Delta}(x_n) = x_n + \Psi_{\Delta}(x_n),$$

(3)

where $\Psi_{\Delta} = \Phi_{\Delta} - I$, with $I$ as the identity operator.

When data for the state variables $x$ over the time stencil (1) are available, they can be grouped into pairs separated by one time step,

$$\{x^{(m)}(0), x^{(m)}(\Delta)\}, \quad m = 1, \ldots, M,$$

where $M$ is the total number of such data pairs. This is the training data set. One can define a residual network [ResNet He et al. (2016)] in the form of

$$y^{out} = [I + N_{\Delta}](y^{in}),$$

(4)

where $N_{\Delta}: \mathbb{R}^d \rightarrow \mathbb{R}^d$ stands for the mapping operator of a standard feed-forward fully connected neural network. An illustration of this structure is shown in Fig. 1.

The network is then trained by using the training data set and minimizing the mean squared loss function,

$$\frac{1}{M} \sum_{m=1}^{M} \left\| x^{(m)}(\Delta) - [I + N_{\Delta}](x^{(m)}(0); \Theta) \right\|^2,$$

(5)

with respect to the hyperparameters of the network (weights and biases) $\Theta$. If the network is well-trained then $N_{\Delta} \approx \Psi_{\Delta}$.

For predictions, numerical stability is essential. Although theoretical analysis is not available at the moment, extensive numerical experiments have shown that numerical stability can be significantly enhanced by using a loss function with outer recurrence.

That is, a sequence of $R_{out}$ snapshots

$$\{x^{(m)}(0), x^{(m)}(\Delta), \ldots, x^{(m)}(R_{out} \Delta)\}, \quad m = 1, \ldots, M,$$

(6)

FIG. 1: ResNet DNN structure
are collected as training data, and the recurrent loss function

\[
\frac{1}{M} \sum_{m=1}^{M} \sum_{r=1}^{R_{\text{out}}} \left\| x^{(m)}(r\Delta) - [I + N_{\Delta}]^r(x^{(m)}(0); \Theta) \right\|^2,
\]

is used instead, where \([I + N_{\Delta}]^r\) indicates composition of the network function with itself \(r\) times.

The trained network thus accomplishes

\[
x^{(m)}(r\Delta) \approx [I + N_{\Delta}]^r(x^{(m)}(0)), \quad \forall m = 1, \ldots, M, \, r = 1, \ldots, R_{\text{out}}.
\]

Once the network is trained to satisfactory accuracy, it can be used as a predictive model on the \(\Delta\) timescale

\[
x_{n+1} = x_n + N_{\Delta}(x_n), \quad n = 0, 1, \ldots
\]

for any initial condition \(x(t_0)\). This framework was proposed in Qin et al. (2019), with extensions to parametric systems and time-dependent (nonautonomous) systems (Qin et al., 2020, 2021).

### 3. LEARNING FINE-SCALE DYNAMICS VIA INNER RECURRENCE

We now present the modeling of unknown systems on a finer timescale. Assume, as in Eq. (6), that the training data set is again made up of observations of the state variables \(x\) of an unknown dynamical system at \(t = 0, \Delta, \ldots, R_{\text{out}}\Delta\). Unlike in the previous section, however, the time scale \(\Delta\) is not sufficient (i.e., too large) to describe and properly determine the dynamics of the underlying system. This could occur for many reasons, including, e.g., limited storage resources, that make \(\Delta\) insufficient to represent the smooth dynamics of the system. Hence we wish to learn the missing fine-scale time dynamics.

#### 3.1 Inner Recurrence for ODE Modeling

Therefore, we introduce the concept of inner recurrence to learn the dynamics on a finer time scale \(\delta < \Delta\). Inner recurrence dictates the time scale of the neural network function itself, that is, the length of time marched forward by the approximate flow map. No loss is measured at the inner recurrence steps, as there are no data for this fine time scale. However, we enforce that the network operator must compose \(R_{\text{in}}\) times between observations such that \(N_{\delta} \approx \Psi_{\delta}\) (as opposed to \(\Psi_{\Delta}\) as above). This is achieved by recurring the network function \(R_{\text{in}}\) times (on the fine time scale \(\delta\)) in between when the loss is measured on the coarse timescale \(\Delta\) for \(r = 1, \ldots, R_{\text{out}}\). In particular, the loss is now given by

\[
\frac{1}{M} \sum_{m=1}^{M} \sum_{r=1}^{R_{\text{out}}} \left\| x^{(m)}(r\Delta) - [I + N_{\delta}]^{r-R_{\text{in}}}(x^{(m)}(0); \Theta) \right\|^2,
\]

where \(\delta\) is chosen such that \(R_{\text{in}} = \Delta/\delta\) is an integer, and \([I + N_{\delta}]^{r-R_{\text{in}}}\) represents the composition of the neural network function \(r \cdot R_{\text{in}}\) times. In practice, the application and variation of the underlying function will dictate the appropriate size of \(\delta\) and \(R_{\text{in}}\) depending on the problem and the observation time scale \(\Delta\). As illustrated in Fig. 2, if inner recurrence \(R_{\text{in}} = 3\) and outer recurrence \(R_{\text{out}} = 4\) then data are observed on the time scale \(\Delta\) but the network learns on the
FIG. 2: Time sampling diagram for $R_{in} = 3$ and $R_{out} = 4$. The squares represent the coarse observed time scale, while the circles represent the fine learning time scale.

time scale $\delta$. In training, the loss is minimized with respect to the network hyperparameters $\Theta$ via Adam (Kingma and Ba, 2014). Another diagram visualizes the loss computation in Fig. 3.

Applying the learning of fine-scale time dynamics to the modeling of unknown systems is straightforward. In particular, the implementation only requires composing the network with itself $R_{in}$ times in between when the loss is measured, which effectively enforces the approximation of $\Psi_\delta$ as opposed to $\Psi_\Delta$. An important assumption in the approximation of a flow map $\Psi_\delta$ is that the time step $\Delta$ is small enough that there exists a unique flow map $\Psi_\Delta$. Later, we show an example where the observation time scale $\Delta$ is too large such that $\Psi_\Delta$ is not unique and therefore the network (depending on random initialization and stochastic optimization) finds either the appropriate map or another solution which goes through the same observation points. Therefore, we use the rule of thumb that if $\Delta$ is small enough (which is problem-dependent) then we can proceed as above.

3.2 Inner Recurrence for PDE Modeling in Nodal Space

The flow map learning approach has also been applied in modeling PDEs. When the solution of the PDE can be expressed using a fixed basis, the learning can be conducted in modal space. In this case, the ResNet approach can be adopted. See Wu and Xiu (2020) for details. When data of the PDE solutions are available as nodal values over a set of grids in physical space, its DNN learning is more involved. In this case, a DNN structure was developed in Chen et al. (2022), which can accommodate the situation when the data are on unstructured grids. It is based on a numerical scheme for solving the PDE, and consists of a set of specialized layers including disassembly layers and an assembly layer, which are used to model the potential differential operators involved in the unknown PDE. The proposed DNN model defines the following mapping,

$$w_{n+1} = w_n + A(F_1(w_n), \ldots, F_J(w_n)), \quad (10)$$

where $F_1, \ldots, F_J$ are the NN operators for the disassembly layers and $A$ is the NN operator for the assembly layer which operates componentwise on the grid elements. An illustration of this structure is shown in Fig. 4. The DNN modeling approach was shown to be highly flexible and accurate to learn a variety of PDEs in Chen et al. (2022). See Chen et al. (2022) for more details on this structure and its mathematical properties. Since this configuration can also be viewed as
a residual network application, i.e. \(w_{n+1} = (I + N)(w_n)\) where \(N = A \circ (F_1, \ldots, F_J)\), we can directly apply the above inner and outer recurrence scheme described above.

4. COMPUTATIONAL STUDIES

In this section, we present six numerical examples (four for ODEs and two for PDEs) to demonstrate the properties of the proposed approach for learning fine scale time dynamics from coarse scale time observations using DNNs with inner recurrence.

4.1 Computational Setting

The structures of the DNNs used to achieve fine time scale flow map learning are those of Eqs. (8) and (10) depending on whether the underlying system is an ODE or PDE, which is reviewed in the previous sections, where Figs. 1 and 4 illustrate the two network structures. Our numerical experimentation and previous work with this framework indicate that particularly wide or deep networks are not typically necessary for learning with this structure. All specifications such as number of layers and neurons are included with each example.

For benchmarking purposes, in all examples the true systems we seek to approximate are in fact known. However, these true models serve only two purposes: (1) to generate synthetic data with which to train the DNN flow map approximations, and (2) to generate reference solutions for comparison with DNN predictions in testing. Therefore, the knowledge of the true system does not in any way facilitate the DNN model approximation. Data generation for both of these tasks is achieved by solving the true systems using a high-order numerical solver, and observing this reference solution at discrete time steps of length \(\Delta\). To generate the training data, \(M\) initial conditions generate trajectories from which a sequence of length \(R_{\text{out}} + 1\) is collected to form the training data set (6). In our examples, typically \(M = 10,000\) and \(R_{\text{out}} \in \{1, \ldots, 10\}\).

Once the training data set has been generated, the learning task is achieved by training the DNN model with the data set (6). In particular, the network hyperparameters (weights and biases) are trained by minimizing the recurrent loss function (9) using the stochastic optimization method Adam (Kingma and Ba, 2014). Typically the models are trained for for 10,000 epochs with batch size 50 and a constant learning rate of \(10^{-3}\) in Tensorflow (Abadi et al., 2015).
After satisfactory network training, we obtain a predictive model for the unknown system which can be marched forward in time on the $\delta$ time scale from any new initial condition. To validate the network prediction, testing data are generated in the same manner as training data were above. In particular, new initial conditions (i.e. ones not fed through the network in training) generate reference solutions using the true governing equations. For prediction, the DNN model is marched forward starting with the first time step of the test trajectory and is compared against the reference. Note that we march forward significantly longer than $R_{out}\Delta$ (the length of each training sequence) to examine the long-term system behavior.

### 4.2 Example 1: Van der Pol Oscillator

We first consider the second-order Van der Pol oscillator,

$$\ddot{y}_1 - (1 - y_1^2)\dot{y}_1 + y_1 = 0,$$

which can be rewritten as the two-dimensional system

$$\begin{cases} \dot{y}_1 = y_2, \\ \dot{y}_2 = (1 - y_1)^2y_2 - y_1. \end{cases}$$

Training data are collected from the two variable system by sampling 10,000 initial conditions uniformly over the computational domain of $[-2, 2] \times [-1.5, 1.5]$, solving the system numerically, and observing at time step $\Delta = 2$ for a total time length of $T = 20$. A standard ResNet with three hidden layers with 20 nodes each is used, with inner recurrence $R_{in} = 10$ and outer recurrences $R_{out} = \{1, 10\}$ such that the fine time learning occurs on the scale of $\delta = \Delta / R_{in} = 0.2$. The mean squared loss function is minimized using Adam with a constant learning rate of $10^{-3}$ for 10,000 epochs.

Prediction is carried out for $T = 200$ via $\delta = 0.2$ (1000 time steps) using 100 new initial conditions uniformly sampled from the same domain. In Fig. 5, we see that despite the jagged appearance of the coarse time observations, the network accurately fills in the missing fine time scale dynamics. In Fig. 6, we look at the log of the average error of 100 test trajectories over time and notice that changing the outer recurrence $R_{out}$ from 1 to 10 has a positive effect on reducing long-term prediction error.

![FIG. 5: Ex. 1: Van der Pol oscillator—example test trajectory](image-url)
4.3 Example 2: Violating Conditions

Next we consider the following pendulum problem,

\[ \ddot{y}_1 = -\beta \sin y_1, \]

which can be rewritten as the two-dimensional first-order system

\[
\begin{aligned}
\dot{y}_1 &= y_2, \\
\dot{y}_2 &= -\beta \sin y_1,
\end{aligned}
\]

where \( \beta = 9.80665 \). Training data are collected by sampling 10,000 initial conditions uniformly over the computational domain of \([-\pi/2, \pi/2] \times [\pi, \pi]\), solving the system numerically, and observing at time step \( \Delta = 1 \) for a total time length of \( T = 10 \). A standard ResNet with three hidden layers with 20 nodes each is used, with outer recurrence \( R_{out} = 1 \) and inner recurrence \( R_{in} = 10 \) such that the fine time learning occurs on the scale of \( \delta = \Delta/R_{in} = 0.1 \). The mean squared loss function is minimized using Adam with a constant learning rate of \( 10^{-3} \) for 10,000 epochs.

Two networks are trained using identical configurations but different random seeds. The random seed controls the initialization and stochastic optimization. In this case the size of \( \Delta \) is too large, such that \( \Psi_\Delta \) is not unique. That is, there are multiple paths between the observed data points. Hence, Fig. 7 shows that in prediction one seed finds the “correct” path through the new testing data while the other seed finds another path that is equally accurate at the observed data but does not represent the flow map governing the data generation. This shows that although the learning occurs on the time scale of \( \delta \), it is still critical to have the observation time scale \( \Delta \) small enough so that the problem is uniquely determined.

4.4 Example 3: Differential–Algebraic System

Next we consider a system of nonlinear differential-algebraic equations, a model for an electric network from Pulch (2013) also studied in Chen and Xiu (2021).
Learning Fine Scale Dynamics

FIG. 7: Ex. 2: pendulum—example test trajectory using two trained networks

\[
\begin{align*}
\dot{u}_1 &= v_2 / C, \\
\dot{u}_2 &= u_1 / L, \\
0 &= v_1 - (G_0 - G_\infty)U_0 \tanh(u_1/U_0) - G_\infty u_1, \\
0 &= v_2 + u_2 + v_1,
\end{align*}
\]

where \( u_1 \) denotes the node voltage, while \( u_2 \) and \( v_1 \) are branch currents. The physical parameters are specified as \( C = 10^{-9}, L = 10^{-6}, U_0 = 1, G_0 = -0.1, \) and \( G_\infty = 0.25. \) Training data are collected from the two variable system for \( (u_1, u_2) \) by sampling 10,000 initial conditions uniformly over the computational domain of \([-2, 2] \times [-0.2, 0.2]\), solving the system numerically, and observing at time step \( \Delta = 5 \times 10^{-8} \) for a total time length of \( T = 10^{-6}\). A standard ResNet with three hidden layers with 20 nodes each is used, with outer recurrence \( R_{out} = 1 \) and inner recurrence \( R_{in} = 10 \) such that the fine time learning occurs on the scale of \( \delta = \Delta / R_{in} = 5 \times 10^{-9}\). The mean squared loss function is minimized using Adam with a constant learning rate of \( 10^{-3} \) for 10,000 epochs.

Prediction is carried out to \( T = 10^{-6} \) via \( \delta = 5 \times 10^{-9} \) (200 time steps) using 100 new initial conditions uniformly sampled from the same domain. The results are shown in Fig. 8. We see from this example test trajectory that the network is still able to perform very accurately on the algebraic variables despite the coarse observations at times missing entire spikes of the functions.

4.5 Example 4: Lorenz System

We consider the Lorenz 63 system,

\[
\begin{align*}
\dot{x} &= 10(y - x), \\
\dot{y} &= x(28 - z) - y, \\
\dot{z} &= xy - \frac{8}{3}z,
\end{align*}
\]

a three-dimensional nonlinear, deterministic, chaotic system. As in Churchill and Xiu (2022), which focuses on ResNet learning of the physics of chaotic systems, training data are collected

Volume 3, Issue 3, 2022
FIG. 8: Ex. 3: differential–algebriac equations—example test trajectory

from the full system by sampling 10,000 chunks uniformly from a single solution of the system starting from initial condition $(1, 1, 1)$ with time step $\Delta = 0.1$ for a total time length of $T = 10,000$. A standard ResNet with three hidden layers with 20 nodes each is used, with outer recurrence $R_{out} = 1$ and inner recurrence $R_{in} = 10$ such that the fine time learning occurs on the scale of $\delta = \Delta/R_{in} = 0.01$. The mean squared loss function is minimized using Adam with a constant learning rate of $10^{-3}$ for 10,000 epochs.

Prediction is carried out for $T = 100$ via $\delta = 0.01$ (10,000 time steps) using the new initial condition $(10, 10, 20)$ as an example. The results are shown in Fig. 9. As in Churchill and Xiu (2022), we highlight that long-term pointwise accuracy for chaotic systems is nearly impossible given the sensitivity of the true system behavior to perturbation and the fact that the system we learn is merely an approximation. Hence, we focus on other qualitative measurements such as bounded pointwise error, histogram and autocorrelation function matching, qualitatively similar phase plots, and measures that quantitatively measure chaos including correlation dimension (Theiler, 1987), approximate entropy (Pincus, 1991), and Lyapunov exponent (Rosenstein et al., 1993), to demonstrate that the fine-scale chaotic behavior of the Lorenz system can be learned from coarser observations. These tools have been used to classify chaotic behavior in Lorenz and other chaotic systems, e.g., in Bakker et al. (2000), Chattopadhyay et al. (2020), Kim et al. (1999), and Rudy et al. (2019). All metrics are computed using MATLAB (2022), in particular, the Econometrics and Predictive Maintenance (The MathWorks, 2022) toolboxes. See Section 3 of Churchill and Xiu (2022) for computational details.

Figure 9 shows the pointwise error of the prediction between $t = 95$ and $t = 100$ s. We see that by this point in time the system prediction is far off from the reference, the chaotic behavior is replicated even on the fine time scale, and the error even after this long time marching is bounded, which demonstrates the long term stability of the prediction. Figure 10 shows that the phase plots for the reference are qualitatively similar to that of the prediction, and also quantitatively demonstrates the accuracy by displaying the chaos complexity measures mentioned...
above. Figure 11 shows a strong match to the approximate density of the values of the reference solution, while finally Fig. 12 shows a good match to the reference autocorrelation functions in each variable.
4.6 Example 5: One-Dimensional PDE

Moving to PDEs, we consider the one-dimensional FitzHugh-Nagumo systems of equations with diffusion,

\[
\begin{align*}
\frac{\partial v}{\partial t} &= v - \frac{1}{3}v^3 - w + Dw, \\
\frac{\partial w}{\partial t} &= \epsilon(v + b - cw),
\end{align*}
\]

with periodic boundary conditions in the computational domain \([0, 5]\) discretized with 50 points, where \(D = 0.01, \epsilon = 0.08, b = 0.7, c = 0.8\). These reaction-diffusion equations simulate the propagation of waves in an excitable medium, such as in heart tissue or nerve fiber. Training data are collected from the two-variable system by sampling 10,000 initial conditions,

\[
v(x, 0) \sim \begin{cases} 
U[0.9, 1.1] & m_1 \leq x \leq m_2 \\
U[-1.2, -1] & m_1 \geq x \geq m_2,
\end{cases}
\]

\[
w(x, 0) \sim \begin{cases} 
U[-0.1, 0.1] & x \leq m_1 \\
U[-0.6, -0.4] & x > m_1,
\end{cases}
\]

where \(m_1 \sim U[0, 1.25]\) and \(m_2 \sim m_1 + U[0.25, 1.5]\), solving the system numerically using a high order method, and observing at time step \(\Delta = 0.25\) of total time length of \(T = 50\). The \(\sim\) notation along with \(U[\cdot, \cdot]\) indicates that this value is drawn from a uniform distribution between the two values. A network structure as in Eq. (10) (also Fig. 4) with five disassembly channels each with one hidden layer and 100 neurons and an assembly layer with one hidden layer and five neurons is used, with outer recurrence \(R_{out} = 3\) and inner recurrence \(R_{in} = 5\) such that the fine time learning occurs on the scale of \(\delta = \Delta/R_{in} = 0.05\). See Chen et al. (2022) for more details on this structure and its mathematical properties. The mean squared loss function is minimized using Adam with a constant learning rate of \(10^{-3}\) for 10,000 epochs.

Prediction is carried out for \(T = 50\) via \(\delta = 0.05\) (2000 time steps) using 100 new initial conditions uniformly sampled from the same domain as well as a demonstrative initial condition defined for \(x \in [0, 5]\) by

\[
v(x, 0) = \begin{cases} 
1 & 0.75 \leq x \leq 1 \\
-1.1 & \text{else}
\end{cases}
\]

\[
w(x, 0) = \begin{cases} 
0 & x \leq 0.75 \\
-0.5 & \text{else.}
\end{cases}
\]

The results are shown in Fig. 13. We see a strong match between the reference and prediction even after the wave has evolved for a long period of time. We show the prediction at times \((t = 12.45, 24.95, 37.45, 49.95\) s\) exclusively on the \(\delta\) time scale (as opposed to a time also coinciding with the \(\Delta\) time scale) to demonstrate accuracy on this finer scale.
4.7 Example 6: Two-Dimensional PDE

As a final ambitious example, we consider the two-dimensional advection-diffusion equation,

\[
\frac{\partial u}{\partial t} + \nabla \cdot (\alpha u) = \nabla \cdot (\kappa \nabla u),
\]

discretized on a 16 × 16 uniform grid grid over domain \((x, y) \in (-1, 1)^2\) with zero Dirichlet boundary conditions. The transport velocity field is set as \(\alpha(x, y) = (y, -x)^T\), and the viscosity is \(\kappa = 5 \times 10^{-3}\). Training data are collected from the equation by sampling 100,000 initial conditions, \(u_0(x, y) = \sum_{k=1}^{N_c} \sum_{l=1}^{N_c} c_{k,l} \sin\left(\frac{k\pi}{2}(x+1)\right) \sin\left(\frac{l\pi}{2}(y+1)\right)\), where \(N_c = 7\) and the coefficient \(c_{k,l} \sim [1/(k + l)]U[-1, 1]\), and observing a numerical solution at time step \(\Delta = 0.01\) of total time length of \(T = 0.1\). A neural network with structure as in Eq. (10) is used with three disassembly channels each with one hidden layer consisting of 256 neurons, and an assembly with one hidden layer and three neurons. The outer recurrence is \(R_{\text{out}} = 1\) and inner recurrence is \(R_{\text{in}} = 5\) such that the fine time learning occurs on the scale of \(\delta = \Delta/R_{\text{in}} = 0.002\). The mean squared loss function is minimized using Adam with a constant learning rate of \(10^{-3}\) for 10,000 epochs.

Prediction is carried out for \(T = 4\) via \(\delta = 0.002\) (2000 time steps) using the initial condition

\[
u_0(x, y) = \frac{C}{2\pi \sigma_x \sigma_y} \exp\left[ -\frac{1}{2} \frac{(x - \mu_x)^2}{\sigma_x^2} - \frac{1}{2} \frac{(y - \mu_y)^2}{\sigma_y^2} \right],
\]
where \( C = 0.2, \mu_x = \mu_y = 0.2, \) and \( \sigma_x = \sigma_y = 0.18. \) The qualitative behavior of this example test trajectory is a Gaussian cone revolving around the origin counter-clockwise while diffusing. The results for prediction up to \( T = 4 \) are shown in Fig. 14, where we see a strong match for a significantly longer time period than the training data sequences. As above, the times displayed are only on the \( \delta \) scale. Interpolatory shading is used to better view results on the fairly coarse \( 16 \times 16 \) grid.

\[
\begin{align*}
\text{reference} & \quad \text{predicted} & \quad \text{absolute error} \\
\text{t = 0.998} & \\
\text{t = 1.998} & \\
\text{t = 2.998} & \\
\text{t = 3.998} & \\
\end{align*}
\]

FIG. 14: Ex. 6: two-dimensional PDE
5. CONCLUSION

We have presented a method for fine timescale DNN learning of unknown systems from trajectories observed on a coarse time scale using the concept of inner recurrence. This new addition builds upon the existing framework for flow map approximation by DNNs and opens it up to real-world problems where data are often sparsely measured in time. A set of six examples is presented which confirms the properties predicted by the foundation and shows that the proposed approach is able to robustly model a variety of standard ODE and PDE problems from data.

ACKNOWLEDGMENT

This work was partially supported by AFOSR FA9550-22-1-0011.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, accessed from tensorflow.org, 2015.

Bakker, R., Schouten, J.C., Giles, C.L., Takens, F., and Van den Bleek, C.M., Learning Chaotic Attractors by Neural Networks, *Neural Comput.*, vol. 12, no. 10, pp. 2355–2383, 2000.

Brunton, S.L., Proctor, J.L., and Kutz, J.N., Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems, *Proc. Natl. Acad. Sci. USA*, vol. 113, no. 15, pp. 3932–3937, 2016.

Chattopadhyay, A., Hassanzadeh, P., and Subramanian, D., Data-Driven Predictions of a Multiscale Lorenz 96 Chaotic System Using Machine-Learning Methods: Reservoir Computing, Artificial Neural Network, and Long Short-Term Memory Network, *Nonlin. Proces. Geophys.*, vol. 27, no. 3, pp. 373–389, 2020.

Chen, Z., Churchill, V., Wu, K., and Xiu, D., Deep Neural Network Modeling of Unknown Partial Differential Equations in Nodal Space, *J. Comput. Phys.*, vol. 449, p. 110782, 2022.

Chen, Z. and Xiu, D., On Generalized Residual Network for Deep Learning of Unknown Dynamical Systems, *J. Comput. Phys.*, vol. 438, p. 110362, 2021.

Churchill, V., Manns, S., Chen, Z., and Xiu, D., Robust Modeling of Unknown Dynamical Systems via Ensemble Averaged Learning, 2022. arXiv: 2203.03458

Churchill, V. and Xiu, D., Deep Learning of Chaotic Systems from Partially-Observed Data, 2022. arXiv: 2205.08384

Fu, X., Chang, L.B., and Xiu, D., Learning Reduced Systems via Deep Neural Networks with Memory, *J. Machine Learning Model. Comput.*, vol. 1, no. 2, pp. 97–118, 2020.

He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Kang, S.H., Liao, W., and Liu, Y., IDENT: Identifying Differential Equations with Numerical Time Evolution, 2019. arXiv: 1904.03538

Kim, H., Eykholt, R., and Salas, J., Nonlinear Dynamics, Delay Times, and Embedding Windows, *Phys. D: Nonlinear Phenom.*, vol. 127, nos. 1-2, pp. 48–60, 1999.

Kingma, D.P. and Ba, J., Adam: A Method for Stochastic Optimization, 2014. arXiv: 1412.6980
Long, Z., Lu, Y., and Dong, B., PDE-Net 2.0: Learning PDEs from Data with a Numeric-Symbolic Hybrid Deep Network, 2018a. arXiv: 1812.04426

Long, Z., Lu, Y., Ma, X., and Dong, B., PDE NET: Learning PDEs from Data, Proc. of the 35th Int. Conf. on Machine Learning, Stockholmsmässan, Stockholm, Sweden, pp. 3208–3216, 2018b.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G.E., Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators, Nat. Machine Intel., vol. 3, no. 3, pp. 218–229, 2021a.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G.E., DeepXDE: A Deep Learning Library for Solving Differential Equations, SIAM Rev., vol. 63, no. 1, pp. 208–228, 2021b.

MATLAB, R2022a, The MathWorks Inc., Natick, MA, USA, 2022.

Pincus, S.M., Approximate Entropy as a Measure of System Complexity, Proc. Nat. Acad. Sci., vol. 88, no. 6, pp. 2297–2301, 1991.

Pulch, R., Polynomial Chaos for Semiexplicit Differential Algebraic Equations of Index 1, Int. J. Uncertain. Quantif., vol. 3, no. 1, 2013.

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Deep Learning of Parameterized Equations with Applications to Uncertainty Quantification, Int. J. Uncertain. Quantif., 2020. DOI: 10.1615/IntJ.UncertaintyQuantification.2020034123

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Data-Driven Learning of NonAutonomous Systems, SIAM J. Sci. Comput., 2021. (in press)

Qin, T., Wu, K., and Xiu, D., Data Driven Governing Equations Approximation Using Deep Neural Networks, J. Comput. Phys., vol. 395, pp. 620–635, 2019.

Raissi, M., Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, J. Machine Learn. Res., vol. 19, no. 25, pp. 1–24, 2018.

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Physics Informed Deep Learning (Part I): Data-Driven Solutions of Nonlinear Partial Differential Equations, 2017a. arXiv: 1711.10561

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Physics Informed Deep Learning (Part II): Data-Driven Discovery of Nonlinear Partial Differential Equations, 2017b. arXiv: 1711.10566

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Multistep Neural Networks for Data-Driven Discovery of Nonlinear Dynamical Systems, 2018. arXiv:1801.01236

Rosenstein, M.T., Collins, J.J., and De Luca, C.J., A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets, Phys. D: Nonlinear Phenom., vol. 65, nos. 1-2, pp. 117–134, 1993.

Rudy, S.H., Brunton, S.L., Proctor, J.L., and Kutz, J.N., Data-Driven Discovery of Partial Differential Equations, Sci. Adv., vol. 3, no. 4, p. e1602614, 2017.

Rudy, S.H., Kutz, J.N., and Brunton, S.L., Deep Learning of Dynamics and Signal-Noise Decomposition with Time-Stepping Constraints, J. Comput. Phys., vol. 396, pp. 483–506, 2019.

Schaeffer, H., Learning Partial Differential Equations via Data Discovery and Sparse Optimization, Proc. Roy. Soc. London A: Math., Phy. Eng. Sci., vol. 473, p. 2197, 2017.

Schaeffer, H. and McCalla, S.G., Sparse Model Selection via Integral Terms, Phys. Rev. E, vol. 96, no. 2, p. 023302, 2017.

Schaeffer, H., Tran, G., and Ward, R., Extracting Sparse High-Dimensional Dynamics from Limited Data, SIAM J. Appl. Math., vol. 78, no. 6, pp. 3279–3295, 2018.

Sun, Y., Zhang, L., and Schaeffer, H., NeuPDE: Neural Network Based Ordinary and Partial Differential Equations for Modeling Time-Dependent Data, 2019. arXiv: 1908.03190

The MathWorks, I., Predictive Maintenance Toolbox, Natick, Massachusetts, USA, accessed from https://www.mathworks.com/products/predictive-maintenance.html, 2022.
Theiler, J., Efficient Algorithm for Estimating the Correlation Dimension from a Set of Discrete Points, *Phys. Rev. A*, vol. 36, no. 9, p. 4456, 1987.

Tran, G. and Ward, R., Exact Recovery of Chaotic Systems from Highly Corrupted Data, *Multiscale Model. Simul.*, vol. 15, no. 3, pp. 1108–1129, 2017.

Wu, K. and Xiu, D., Data-Driven Deep Learning of Partial Differential Equations in Modal Space, *J. Comput. Phys.*, vol. 408, p. 109307, 2020.