Learning Dynamics from Noisy Measurements using Deep Learning with a Runge-Kutta Constraint

Anonymous Author(s)
Affiliation
Address
email

Abstract

Measurement noise is an integral part while collecting data of a physical process. Thus, noise removal is a necessary step to draw conclusions from these data, and it often becomes quite essential to construct dynamical models using these data. We discuss a methodology to learn dynamic models using noisy and sparsely sampled measurements. In our methodology, the main innovation can be seen in the integration of deep neural networks with a classical numerical integration method. Precisely, we aim at learning a neural network that implicitly represents the data and an additional neural network that models the vector fields of the dependent variables. We combine these two networks by enforcing the constraint that the data at the next time-steps can be given by following a numerical integration scheme such as the fourth-order Runge-Kutta scheme. The proposed framework to learn a model predicting the vector field is highly effective under noisy measurements. We demonstrate the effectiveness of the proposed method to learning models using data obtained from various differential equations. The proposed approach provides a promising methodology to learn dynamic models, where the first-principle understanding remains opaque.

1 Introduction

Uncovering dynamic models explaining physical phenomena and dynamic behaviors has been active research for centuries. When a model describing the underlying dynamics is available, it can be used for several engineering studies such as process design, optimization, predictions, and control. Conventional approaches based on physical laws, and empirical knowledge are often used to derive dynamical models. However, this is impenetrable for many complex systems, e.g., understanding the Arctic ice pack dynamics, sea ice, power grids, neuroscience, or finance, to only name a few applications. Data-driven methods to discover models have enormous potential to better understand transient behaviors in the latter cases. Furthermore, data acquired using imaging devices or sensors are contaminated with measurement noise. Therefore, systematic approaches that learn a dynamic model with proper treatment of noise are required. In this work, we discuss a deep learning-based approach to learn a dynamic model by attenuating noise with a Runge-Kutta scheme, thus allowing us to learn models quite accurately even when data are highly corrupted with measurement noise.

Data-driven methods to learn the governing equations of dynamic models have been studied for several decades, see, e.g., [1–3]. Learning linear models from input-output data goes back to Ho and Kalman [4]. There have been several algorithmic developments for linear systems, for example, the eigensystem realization algorithm (ERA) [5, 6], and Kalman filter-based approaches [7–9]. Dynamic mode decomposition (DMD) has also emerged as a promising approach to construct models from input-output data and has been widely applied in fluid dynamics applications, see, e.g., [10–12]. Furthermore, there has been a series of developments to learn nonlinear dynamic models. This

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

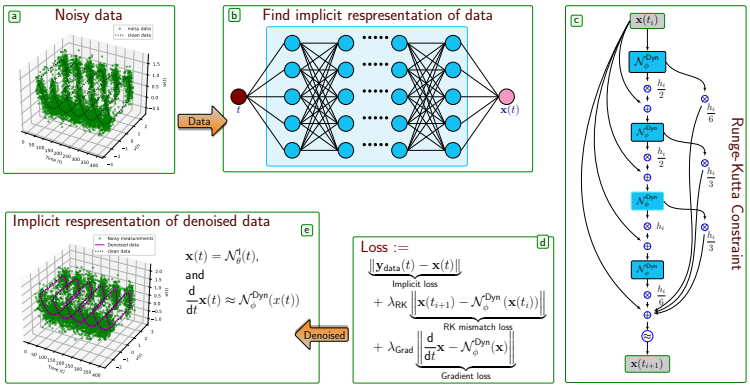

Figure 1: It illustrates the framework to de-noise data and learns a dynamical model. For this, we find an approximate implicit representation of data by a network ($\mathcal{N}_\theta^{\mathsf{I}}$) and another network for the vector field ($\mathcal{N}_\phi^{\mathsf{Dyn}}$). These networks are connected by enforcing a Runge-Kutta scheme, shown in (c), on the output of the network $\mathcal{N}_\theta^{\mathsf{I}}$.

includes, for example, equations free modeling [13], nonlinear regression [14], dynamic modeling [15], and automated inference of dynamics [16–18]. Utilizing symbolic regression and an evolutionary algorithm [19, 20], learning compact nonlinear models becomes possible. Moreover, leveraging sparsity (also known as sparse regression), several approaches have been proposed [21–26]. We also mention the work [27] that learns models using Gaussian process regression. All these methods have particular approaches to handle noise in the data. For example, sparse regression methods, e.g., [21, 22, 26] often utilize smoothing methods before identifying models, and the work [27] handles measurement noise as data represented like a Gaussian process.

Even though the aforementioned nonlinear modeling methods are appealing and powerful in providing analytic expressions for models, they are often built upon model hypotheses. For many complex dynamics, such as the melting Arctic ice, the utilization of these methods is not trivial. Thus, machine learning techniques, particularly deep learning-based, have emerged as powerful methods capable of expressing any complex function in a black-box manner given enough training data. Neural network-based approaches in the context of dynamical systems have been discussed in [28–31] decades ago. Neural-networks are incorporated in various ways in the course of learning models, e.g., for prediction, see [30, 32–43]. However, measurement data are often corrupted with noise, and these mentioned approaches do not perform any specific noise treatment. The work in [44] proposes a framework that explicitly incorporates the noise into a numerical time-stepping method. Though the approach has shown promising directions, its scalability remains ambiguous as the approach explicitly estimates noise and aims to decompose the signal explicitly into noise and ground-truth.

Our contributions: Our work introduces a framework to learn dynamics models by innovatively blending deep learning with numerical integration methods from noisy and sparse measurements. Precisely, we aim at learning two networks; one that implicitly represents given measurement data and the second one approximates the vector field; we connect these two networks by enforcing a numerical integration scheme as depicted in Figure 1. The appeal of the approach is that we do not require an explicit estimate of noise to learn a model. Furthermore, the approach is applicable even if the dependent variables are sampled on different time grids.

The remaining structure of the paper is as follows. In Section 2, we present our deep learning-based framework for learning dynamics from noisy measurements by combining two networks. One of these networks implicitly represents measurement data, and the other one approximates the vector field, which is connected by enforcing a numerical integration scheme. In the subsequent section, we demonstrate the effectiveness of the proposed methodology using various synthetic data with increasing levels of noise, describing various physical phenomena. We conclude the paper with a summary and future research directions.

2 Learning Dynamical Models using Deep Learning Constraint by a Runge-Kutta Scheme

Data-driven methods to learn dynamic models have flourished significantly in the last couple of decades. For these methods, the quality of measurement data plays a significant role in ensuring the accuracy of the learned models. While dealing with real-world measurements, sensor noise in the

collected data is inevitable. Thus, before employing any data-driven method, de-noising the data is a vital step and is typically done using classical methods, e.g., smothering techniques, moving averages, or the noise is explicitly estimated along with dynamics that imposes a challenge in a large-scale setting. In this section, we discuss our framework to learn dynamic models using noisy measurements without explicitly estimating noise. To achieve the goal, we utilize the powerful approximation capabilities of deep neural networks and its automatic differentiation feature with a numerical integration scheme. In this work, we focus on the *fourth-order Runge-Kutta* (RK4) scheme; however, the framework is flexible to use any other numerical integration scheme or higher-order Runge-Kutta schemes. Before we proceed further, we briefly outline the RK4 scheme. For this, let us consider an autonomous nonlinear differential equation:

$$\tfrac{\mathsf{d}}{\mathsf{d}t}\mathbf{x}(t) = \mathbf{g}(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the solution at time t, and the continuous function $\mathbf{g}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ defines the vector field. Given (1) and $\mathbf{x}_{t_i}$, we can predict $\mathbf{x}(t)$ at time t_{i+1} using RK4 scheme as follows:

$$\mathbf{x}(t_{i+1}) \approx \mathbf{x}(t_j) + h_i \left(\frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4\right), \quad \text{with } \; h_i = t_{i+1} - t_i, \tag{2}$$

where

$$\mathbf{k}_1 = \mathbf{g}\left(\mathbf{x}(t_i)\right), \quad \mathbf{k}_2 = \mathbf{g}\left(\mathbf{x}(t_i) + \frac{h_i}{2}\mathbf{k}_1\right), \quad \mathbf{k}_3 = \mathbf{g}\left(\mathbf{x}(t_i) + \frac{h_i}{2}\mathbf{k}_2\right), \quad \mathbf{k}_4 = \mathbf{g}\left(\mathbf{x}(t_i) + h_i\mathbf{k}_3\right).$$

In this paper, we use a short-hand notation for (2) as $\mathbf{x}(t_{i+1}) := \Pi_{\mathsf{RK}}\left(\mathbf{x}(t_i)\right)$. In what follows, we assume that the ground-truth (or de-noised) sequence approximately follows the RK4 steps. We emphasize that the information of the vector field at $\mathbf{x}$ is directly utilized in the RK4 scheme.

Having described the RK4 scheme, we are now ready to proceed to discuss our framework to learn dynamical models from noisy measurements by blending deep neural networks with the RK4 scheme. The approach involves two networks. The first network implicitly represents the variable as shown in Figure 1(b), and the second network approximates the vector field, or the function $\mathbf{g}(\cdot)$. These two networks are connected by attenuating the RK4 constraints. That is, the output of the implicit network is not only in the vicinity of the measurement data but also approximately follows the RK4 scheme as depicted in Figure 1(c). To make things mathematically precise, let us denote noisy measurement data at time t_i by $\mathbf{y}(t_i)$. Furthermore, we consider a feed-forward neural network, denoted by $\mathcal{N}_\theta^{\mathsf{I}}$ parameterized by θ, that approximately yields an implicit representation of measurement data, i.e.,

$$\mathcal{N}_\theta^{\mathsf{I}}(t_i) := \mathbf{x}(t_i) \approx \mathbf{y}(t_i), \tag{3}$$

where $i \in \{1, \ldots, M\}$ with M being the total number of measurements. Additionally, let us denote another neural network by $\mathcal{N}_\phi^{\mathsf{Dyn}}$ parameterized by ϕ that approximates $\mathbf{g}(\cdot)$ in (1). We connect these two networks by enforcing the output of the network $\mathcal{N}_\theta^{\mathsf{I}}$ to respects the RK4 scheme, i.e.,

$$\mathbf{x}(t_{i+1}) \approx \Pi_{\mathsf{RK}}\mathbf{x}(t_i), \qquad \text{and} \quad \tfrac{\mathsf{d}}{\mathsf{d}t}\mathbf{x}(t_i) \approx \mathcal{N}_\phi^{\mathsf{Dyn}}\left(\mathbf{x}(t_i)\right). \tag{4}$$

As a result, our goal becomes to determine the network parameters $\{\theta, \phi\}$ such that the following loss is minimized:

$$\mathcal{L} = \mathcal{L}_{\mathsf{MSE}} + \lambda_{\mathsf{RK}} \cdot \mathcal{L}_{\mathsf{RK}} + \lambda_{\mathsf{Grad}} \cdot \mathcal{L}_{\mathsf{Grad}}, \tag{5}$$

in which $\mathcal{L}_{\mathsf{MSE}}$ denotes the mean square error of the output of the network $\mathcal{N}_\theta^{\mathsf{I}}$ and noisy measurements, i.e., $\left\|\mathcal{N}_\theta^{\mathsf{I}}(t_i) - \mathbf{y}(t_i)\right\|^2$, where $\|\cdot\|$ represents the mean square error. The term $\mathcal{L}_{\mathsf{RK}}$ links the two networks by the RK4 scheme. Precisely, the term $\mathcal{L}_{\mathsf{RK}}$ castigates the mismatch between $\mathbf{x}(t_{i+1})$ and $\Pi_{\mathsf{RK}}\mathbf{x}(t_i)$, i.e., $\|\mathbf{x}(t_{i+1}) - \Pi_{\mathsf{RK}}\left(\mathbf{x}(t_i)\right)\|^2$, and since the vector field at the output of the implicit network can also be computed directly using automatic differentiation, but it also can be computed using the network $\mathcal{N}_\phi^{\mathsf{Dyn}}$. The term $\mathcal{L}_{\mathsf{Grad}}$ penalizes its mismatch as follows: $\left\|\mathcal{N}_\phi^{\mathsf{Dyn}}(\mathbf{x}(t_i)) - \tfrac{\mathsf{d}}{\mathsf{d}t}\mathbf{x}(t_i)\right\|^2$. λ_{RK} and λ_{Grad} are the corresponding regularization parameters.

The total loss $\mathcal{L}$ can be minimized using a gradient-based optimizer such as Adam [45]. Once the networks are trained and have found their parameters that minimize the loss, we can generate the de-noised variables using the implicit network $\mathcal{N}_\theta^{\mathsf{I}}$, and the vector field by the network $\mathcal{N}_\phi^{\mathsf{Dyn}}$. Note that due to the implicit nature of the network, the measurement data can be at variable time steps, and we can estimate the solution at any arbitrary time. Moreover, we also obtain the network $\mathcal{N}_\phi^{\mathsf{Dyn}}(\cdot)$ that approximately provides the vector field for $\mathbf{x}$; hence, one can use it to make predictions. We have discussed an extension to partial differential equations as well in Appendix A.

3 Numerical Experiments

Here, we demonstrate the proposed methodology using artificially corrupted data obtained using (partial) differential equations. A detailed numerical experiment discussion can be found in Appendices B and C including an extension to partial differential equations, neural network parameters, and architectures. To showcase a few results here, in Figure 2, we plot de-noised data and subsequently learned models, describing vector field $\mathbf{g}(\cdot)$ for threes examples, namely Fitz-Hugh Nagumo, a cubic oscillator, and Burgers' equations.

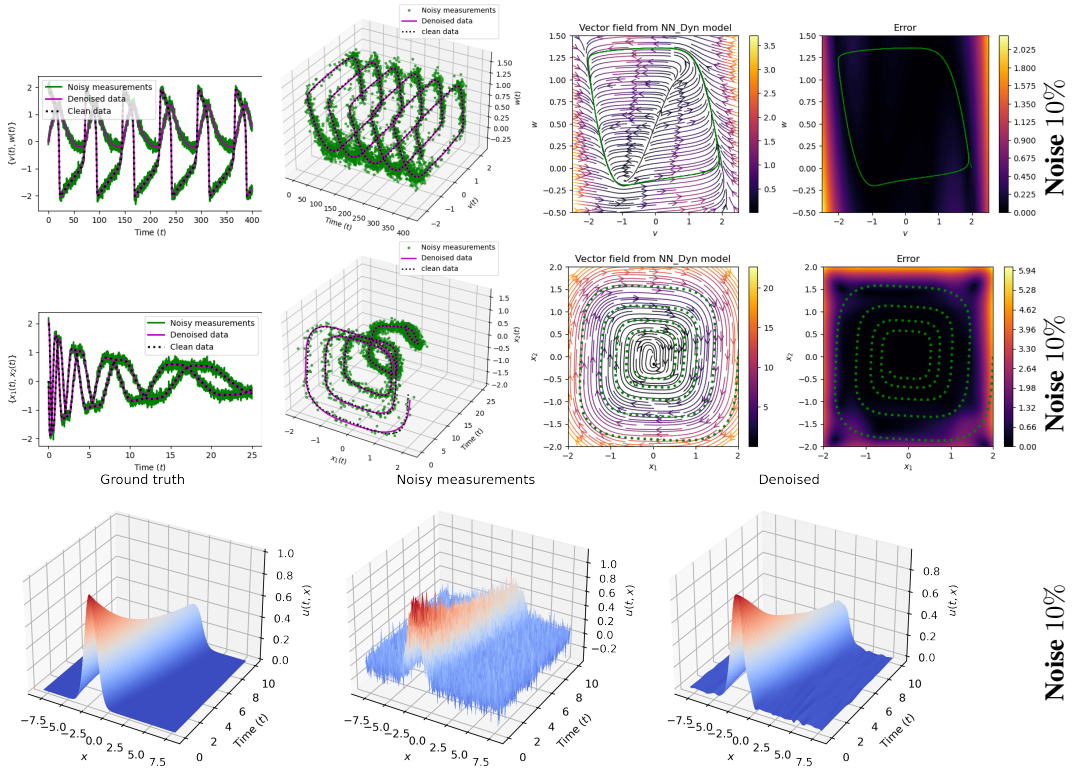

Figure 2: The figure showcases results obtained using the proposed methodology to obtain de-noised data, as well as dynamical systems or models describing vector fields. The measurements are corrupted using Gaussian white noise of 10%. The first row is of the Fitz-Hugh model, and the second one is of a cubic oscillator model. The last columns in these rows show the error in estimating the vector field. The third row presents the results of a Burgers' equation which is a PDE. For a more detailed discussion of numerical experiments, we refer to Appendix C.

4 Discussion

In this work, we have presented a new paradigm for learning dynamical models from highly noisy (spatial)-temporal measurement data. Our framework blends powerful approximation capabilities of deep neural networks with a numerical integration scheme, namely the fourth-order Runge-Kutta scheme. The proposed scheme involves two networks to learn an implicit representation of the measurement data and of the vector field. These networks are combined by enforcing that the output of the implicit network respects the integration scheme.

We note that the approach becomes computationally expensive when the spatial dimension increases. However, we know that the dynamics often lie in a low-dimensional manifold. Therefore, in our future work, we aim to utilize the concept of low-dimensional embedding to make learning computationally more efficient. Furthermore, it could be interesting to combine with priors to improve training and obtain interpretable and generalizable models.

References

[1] J.-N. Juang. *Applied System Identification*. Prentice-Hall, 1994.

[2] L. Ljung. *System Identification – Theory for the User*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[3] Stephen A Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. John Wiley & Sons, 2013.

[4] BL Ho and Rudolf E Kálmán. Effective construction of linear state-variable models from input/output functions. *Automatisierungstechnik*, 14(1-12):545–548, 1966.

[5] Jer-Nan Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *J. Guidance, Control, and Dyn.*, 8(5):620–627, 1985.

[6] Richard W Longman and Jer-Nan Juang. Recursive form of the eigensystem realization algorithm for system identification. *J. Guidance, Control, and Dyn.*, 12(5):647–652, 1989.

[7] Jer-Nan Juang, Minh Phan, Lucas G Horta, and Richard W Longman. Identification of observer/Kalman filter Markov parameters-theory and experiments. *J. Guidance, Control, and Dyn.*, 16(2):320–329, 1993.

[8] Minh Phan, Lucas G Horta, Jer-Nan Juang, and Richard W Longman. Linear system identification via an asymptotically stable observer. *J. Opt. Th. Appl.*, 79(1):59–86, 1993.

[9] Minh Phan, Jer-Nan Juang, and Richard W Longman. Identification of linear multivariable systems by identification of observers with assigned real eigenvalues. *J. Astronautical Sci.*, 40(2), 1992.

[10] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *J. Fluid Engrg.*, 82(1), 1960.

[11] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.*, 656:5–28, 2010.

[12] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: Theory and applications. *J. Comput. Dyn.*, 1(2):391–421, 2014.

[13] Ioannis G Kevrekidis, C William Gear, James M Hyman, Panagiotis G Kevrekidis, Olof Runborg, Constantinos Theodoropoulos, et al. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Comm. Math. Sci.*, 1(4):715–762, 2003.

[14] Henning U Voss, Paul Kolodner, Markus Abel, and Jürgen Kurths. Amplitude equations from spatiotemporal binary-fluid convection data. *Physical Rev. Lett.*, 83(17):3422, 1999.

[15] Hao Ye, Richard J Beamish, Sarah M Glaser, Sue CH Grant, Chih-Hao Hsieh, Laura J Richards, Jon T Schnute, and George Sugihara. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proc. Nat. Acad. Sci. U.S.A.*, 112(13):E1569–E1576, 2015.

[16] Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. Automated refinement and inference of analytical models for metabolic networks. *Phy. Biology*, 8(5):055011, 2011.

[17] Bryan C Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature Comm.*, 6(1):1–8, 2015.

[18] Bryan C Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using S-systems and alternating regression. *PLoS One*, 10(3):e0119821, 2015.

[19] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proc. Nat. Acad. Sci. U.S.A.*, 104(24):9943–9948, 2007.

[20] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[21] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[22] Niall M Mangan, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Trans. Molecular, Biological and Multi-Scale Comm.*, 2(1):52–63, 2016.

[23] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15(3):1108–1129, 2017.

[24] Hayden Schaeffer, Giang Tran, Rachel Ward, and Linan Zhang. Extracting structured dynamical systems using sparse optimization with very few samples. *Multiscale Modeling & Simulation*, 18(4):1435–1461, 2020.

[25] Niall M Mangan, J Nathan Kutz, Steven L Brunton, and Joshua L Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204):20170009, 2017.

[26] Pawan Goyal and Peter Benner. Discovery of nonlinear dynamical systems using a Runge-Kutta inspired dictionary-based sparse regression approach. *arXiv preprint arXiv:2105.04869*, 2021.

[27] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.*, 357:125–141, 2018.

[28] Sheng Chen, Stephen A Billings, and PM Grant. Non-linear system identification using neural networks. *Intern. J. Control*, 51(6):1191–1214, 1990.

[29] Ramiro Rico-Martinez and Ioannis G Kevrekidis. Continuous time modeling of nonlinear systems: A neural network-based approach. In *IEEE Int. Conf. on Neural Networks*, pages 1522–1525, 1993.

[30] R Gonzalez-Garcia, R Rico-Martinez, and IG Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engrg.*, 22:S965–S968, 1998.

[31] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *J. Comput. Phys.*, 182(1):1–26, 2002.

[32] Zhixin Lu, Brian R Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.

[33] Shaowu Pan and Karthik Duraisamy. Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity*, 2018, 2018.

[34] Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary J. Nonlinear Sci.*, 27(12):121102, 2017.

[35] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary J. Nonlinear Sci.*, 28(4):041101, 2018.

[36] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[37] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Comm.*, 9(1):1–10, 2018.

[38] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Proc. 31st Internat. Conf. Neural Information Processing Sys.*, pages 1130–1140, 2017.

[39] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[40] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proc. Nat. Acad. Sci. U.S.A.*, 116(45):22445–22451, 2019.

[41] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[42] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[43] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[44] Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *J. Comput. Phys.*, 396:483–506, 2019.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Adv. in Neural Information Processing Systems*, volume 33, 2020.

[47] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

[48] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Internat. Conf. Machine Learning*, pages 448–456. PMLR, 2015.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[50] Richard FitzHugh. Mathematical models of threshold phenomena in the nerve membrane. *The Bulletin Math. Biophys.*, 17(4):257–278, 1955.

[51] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Nat. Acad. Sci. U.S.A.*, 113(15):3932–3937, 2016.

[52] Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM J. Appl. Dyn. Syst.*, 18(2):643–660, 2019.

[53] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Sci. Adv.*, 3(4):e1602614, 2017.

[54] Yoshiki Kuramoto. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 1978.

A Possible Extensions of the Approach

In many instances, dynamical processes may involve system parameters, and by varying them, the processes exhibit different dynamics. Also, on several occasions, dynamics are governed by underlying partial differential equations. In this section, we shortly discuss extensions of the proposed approach to these two cases.

A.1 Parametric models

The approach discussed in the previous section readily extends to parametric cases. Let us consider a parametric differential equation as follows:

$$\frac{\mathsf{d}}{\mathsf{d}t}\mathbf{x}(t;\mu) = \mathbf{g}(\mathbf{x}(t;\mu),\mu), \tag{6}$$

where $\mu \in \mathbb{R}^d$ is the system parameter. To handle parameter μ, we can simply take the parameter μ as an additional input to the implicit network $\mathcal{N}_\theta^{\mathsf{I}}$ that yields the dependent variables at a given time and parameter. Furthermore, to learn the function $\mathbf{g}(\mathbf{x}(t;\mu),\mu)$, we take the parameter μ as an input as well along with $\mathbf{x}(t;\mu)$ to obtain a parameterized dynamical model to predict the vector field at a given $\mathbf{x}$ and parameter.

A.2 Partial differential equations

Many cases, for example, dynamics of flows, dynamical behaviors, are governed by partial differential equations; thus, the dependent variable $\mathbf{x}$ is highly influenced by its neighbors. In such a case, we construct an implicit representation for measurement data such that the implicit network takes time t and the spatial coordinates ζ as inputs and yields dependent variable $\mathbf{x}(t,\zeta)$. Then, we compute $\mathbf{X}_\zeta$ containing $\mathbf{x}$ at user-specified spatial locations. This can be used to learn a dynamic model that describes dynamics at these spatial locations. Consequently, with these discussed alterations, one can employ the approach discussed in the previous section. The strength of the approach is that the collected measurement data can be at any arbitrary spatial location. These locations can also vary with time since we construct an implicit network that is independent of any structure in the collected measurements.

B Suitable Neural Networks Architectures

Here, we briefly discuss neural network architectures suitable for our proposed approach. We require two neural networks for our framework, one for learning the implicit representation $\mathcal{N}_\theta^{\mathsf{I}}$ and the second one $\mathcal{N}_\theta^{\mathsf{Dyn}}$ is to learn the vector field. For implicit representation, we use a fully connected multi-layer perceptron (MLP) as depicted in Figure 3(a) with periodic activation functions (e.g., $\sin$) [46] which has shown its ability to capture finely detailed features as well as the gradients of a function. To approximate the vector field, we consider two possibilities depending on applications. If the data do not have any spatial dependency, then we consider a simple residual-type network as illustrated in Figure 3(b) with *exponential linear unit* (ELU) as an activation function [47]. We choose ELU as the activation function since it is continuous and differentiable and resembles a widely used activation function, namely rectified linear unit (ReLU). On the other hand, when the data has spatial correlations, e.g., dynamics in data are governed by a partial differential equation, then it is more intuitive to use a convolutional neural network (CNN) with residual connections as depicted in Figure 3(c). It explicitly makes use of the spatial correlation. For CNN, we also employ the batch normalization scheme [48] after each convolution step for a better distribution of the input to the next layer and use ELU as an activation function.

C Detailed Numerical Experiments

In this section, we investigate the performance of the approach discussed in Section 2 to de-noise measurement data as well as learning a model for estimating the vector field. To that aim, we consider data obtained by solving several (partial) differential equations that are then corrupted using white

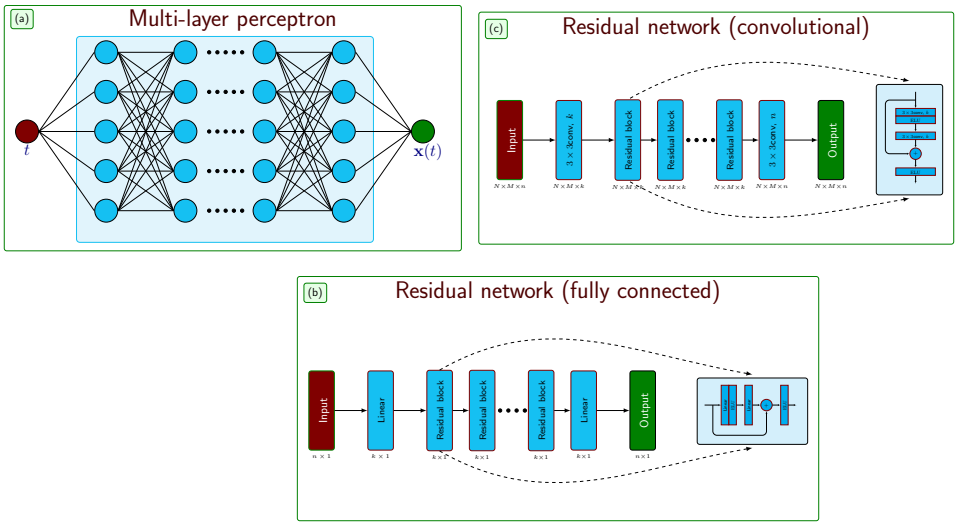

Figure 3: The figure shows three potential simple architectures that can be used to learn either implicit representation or to approximate the underlying vector field. Diagram (a) is a simple multi-layer perceptron, and (b) is a residual-type network but fully connected, and (c) is a residual-type network with convolutional layers. The notation $\{3 \times 3 \text{ conv}, k\}$ means k filters of 3×3 receptive field.

Example	Networks	Neurons	Layers or residual blocks	Learning rates
FHN	For implicit representation	20	4	$5 \cdot 10^{-4}$
	For approximating vector field	20	4	10^{-3}
Cubic oscillator	For implicit representation	20	4	$5 \cdot 10^{-4}$
	For approximating vector field	20	4	10^{-3}

Table 1: The table shows the information about network architectures and learning rates.

Gaussian noise by varying the noise level. For a given percentage of noise, we determine the noise as follows:

$$\nu \sim \mathcal{N}(0, \Sigma), \quad \text{with } \Sigma = \text{Noise}\%. \tag{7}$$

We have implemented our framework using the deep learning library PyTorch [49] and have optimized all networks together using the Adam optimizer [45]. Furthermore, to train implicit networks, we map the input data to $[-1, 1]$ as recommended in [46]. Additionally, to avoid over-fitting, we add L_2-regularization (also referred to as *weight decay*) of the parameters of the networks and set the regularization parameter to 10^{-4} for all examples. All the networks are trained for 15 000 epochs with batch size 1, and learning rates used to train networks are stated in each example in their respective subsections. We have run all our experiments on a NVIDIA® A100 GPU.

C.1 Fitz-Hugh Nagumo model

In the first example, we discuss the Fitz-Hugh Nagumo (FHN) model that explains neural dynamics in a simplistic way [50]. This has been used as a test case to discover the model using dictionary-based sparse regression [26]. The dynamics are given as follows:

$$\begin{aligned} \mathbf{v}(t) &= \mathbf{v}(t) - \mathbf{w}(t) - \tfrac{1}{3}\mathbf{v}(t)^3 + 0.5, \\ \mathbf{w}(t) &= 0.040\mathbf{v}(t) - 0.028\mathbf{w}(t) + 0.032, \end{aligned} \tag{8}$$

where $\mathbf{v}(t)$ and $\mathbf{w}(t)$ describe the dynamics of activation and de-activation of neurons. We collect 4 000 measurements in time $t \in [0, 400]$ at a regular interval by simulating using the initial condition $[\mathbf{v}(0), \mathbf{w}(0)] = [2, 0]$. We then corrupt the data artificially by adding various levels of noise. We build two networks with the information provided in Table 1. We have set both the parameters λ_{Dyn} and λ_{Grad} in the loss function (5) to 1.

Having trained networks, we obtain de-noised measurement data using the implicit network and estimate the vector field using the neural network. The results are shown in Figure 4. The figure demonstrates the robustness of the approach with respect to noise. The method can recover the data very close to clean data (see the first two columns of the figure) even when the measurements are corrupted with relatively more significant noise, e.g., 20% to 40%. Furthermore, the vector field is estimated quite accurately, at least in the regime of the collected measurements, see the third and fourth columns of the figure. However, as expected, the vector field estimates are inadequate away from the measurements, thus, showing the limitation in extrapolating to the regime where no data is available. Nevertheless, this can be improved by collecting more measurements in a different regime by varying initial conditions.

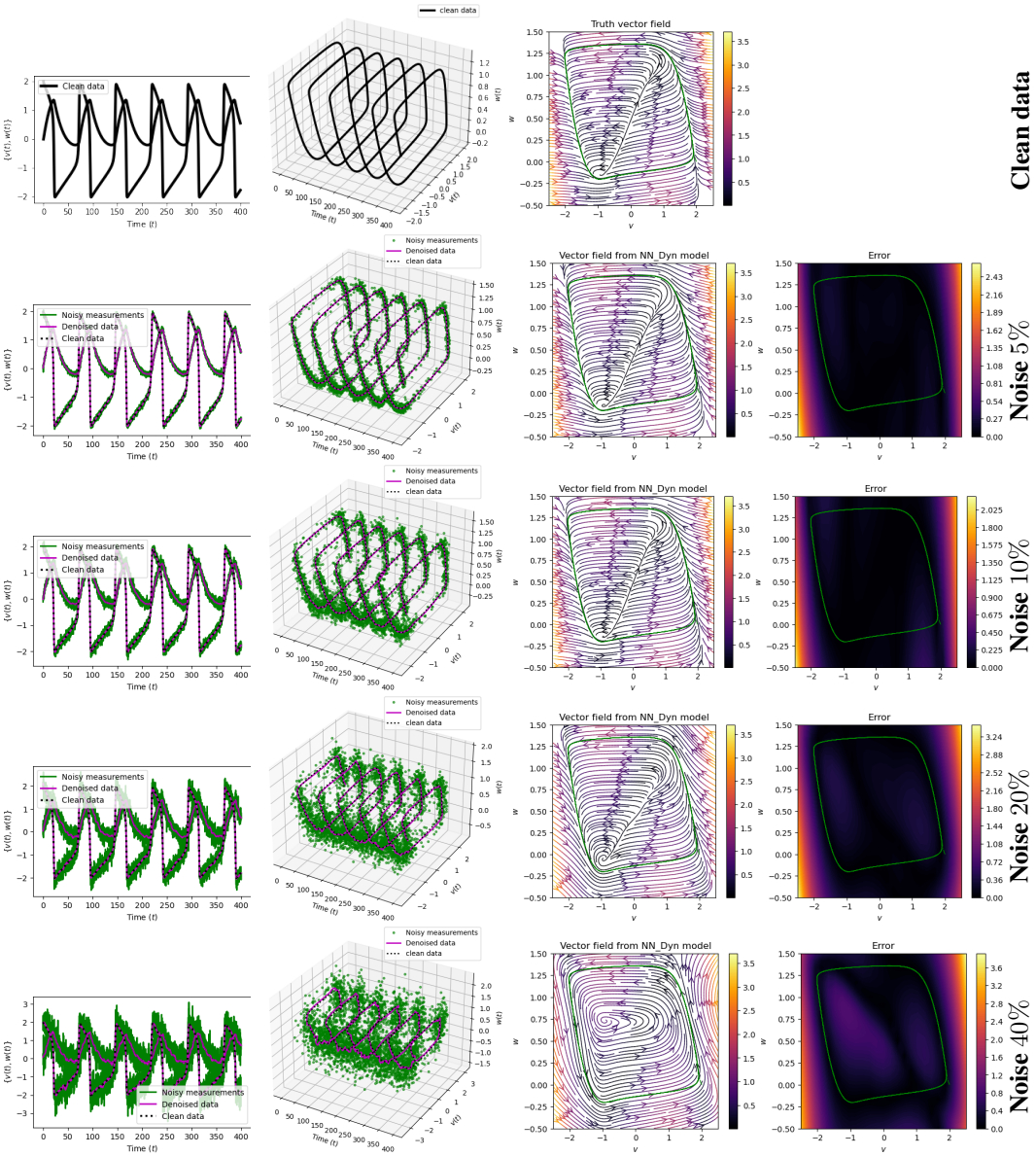

Figure 4: Fitz-Hugh Nagumo Model. The figure shows the performance of the proposed approach to recover the truth signal. In the first and second columns, we present the noisy, clean, and recovered data. In the third and fourth columns, we show the estimate of the vector field using the learned neural model for it, and the green dots are the clean data in the domain. We observe that the vector fields are very well estimated in the regime of the collected data.

Example	Networks	Neurons or filters	Layers or residual blocks	Learning rates
Burgers example	For implicit representation	10	4	$5 \cdot 10^{-4}$
	For approximating vector field	8	4	10^{-3}
Kuramoto–Sivashinsky example	For implicit representation	50	4	$5 \cdot 10^{-4}$
	For approximating vector field	16	4	10^{-3}

Table 2: The table shows the information about the network for Burgers and Kuramoto–Sivashinsky examples.

C.2 Cubic damped model

In the second example, we consider a damped cubic system, which is described by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= -0.1\mathbf{x}(t)^3 + 2.0\mathbf{y}(t)^3, \\ \dot{\mathbf{y}}(t) &= -2.0\mathbf{x}(t)^3 - 0.1\mathbf{y}(t)^3. \end{aligned} \tag{9}$$

It has been one of the benchmark examples in discovering models using data, see, e.g., [26, 51] but there, it is assumed that the dynamics can be given sparsely in a high-dimensional feature dictionary. Here, we do not make any such assumptions and instead learn the vector field using a neural network along the line of [52]. For this example, we take $2\ 500$ data points in the time interval $[0, 10]$ by simulating the model using the initial condition $[2, 0]$ as done in [52]. We add various levels of noise in the clean data to have noisy measurements synthetically. We, again, perform similar experiments as done in the previous example. We construct neural networks for implicit representation and the vector field with the parameters given in Table 1.

Having trained networks with parameters $\lambda_{\mathsf{Dyn}} = 1$ and $\lambda_{\mathsf{Grad}} = 0.05$ in the loss function (5), we have an implicit network to obtain de-noised signal and a neural network approximating the vector field. We plot the results in Figure 5, where we show noisy, clean, and de-noised data in the first two columns, and in the third and fourth columns, we plot the streamlines of the vector field, obtained using the trained neural network. We observe that the de-noised data faithfully matches with the clean data even for a high noise level, and the vector field is also close to the ground truth, at least in the region where measurement data are sampled. However, in the region where no data are available, the vector field approximation is poor, as one can expect. However, having richer data covering a larger training regime can improve the performance of the neural network, approximating the vector field.

C.3 Burgers equation

Next, we examine the case, where collected measurements have spatial correlation as well, meaning there is an underlying partial differential equation, describing the dynamics. Here, we consider a 1D viscous Burger equation. It explains several phenomenons occurring in fluid dynamics and is governed by

$$\frac{\partial \mathbf{u}(t,\zeta)}{\partial t} = -\mu \mathbf{u}_{\zeta\zeta} f(t,\zeta) + \mathbf{u}(t,\zeta)\mathbf{u}_{\zeta}(t,\zeta), \tag{10}$$

where μ is the viscosity; $\mathbf{u}_{\zeta}$ and $\mathbf{u}_{\zeta\zeta}$ denote the first and second derivatives with respect to the spatial variable ζ, and the equation is also subject to a boundary condition. We have taken the data from [53], followed by artificially corrupting them using various levels of Gaussian white noise. In brief, the measurements are collected at 256 grid point in the domain $\zeta \in [-8, 8]$ and at the time interval $\mathsf{d}t = 0.1$. For more details, we refer to [53].

Since the data has spatial correlations, we make use of convolutional neural networks to learn the vector field of $\mathbf{u}$, instead of a classical MLP as shown in Figure 3. Thus, we build an MLP for the implicit representation and a CNN with details given in Table 2. Once we train the network, in Figure 6, we plot the performance of the proposed approach to de-noise the spatial-temporal data for an increasing level of noise. We observe that the proposed methodology is able to recover the data faithfully even with significant noise in data. Furthermore, in the last columns of Figure 6, we observe the approximating capability of the convolutional NN for the vector field, e.g., $\frac{\partial}{\partial t}\mathbf{u}(t,\zeta)$. We observe that the model also predicts the vector field with good accuracy. We mark that the vector field of the clean data is estimated using a finite-difference scheme on the clean data since the true function is not known to us.

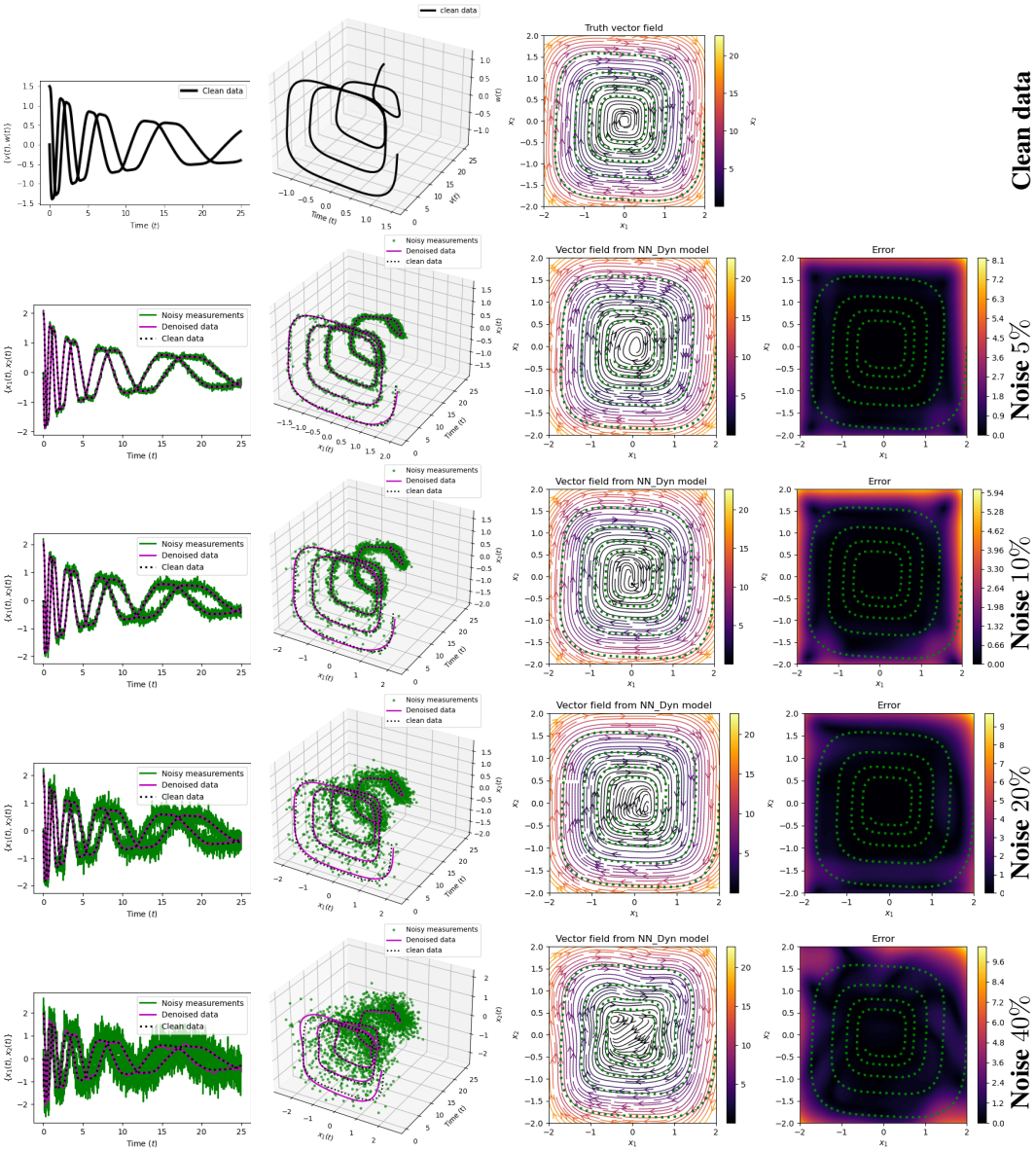

Figure 5: Damped cubic model. We visualize the noisy, clean, and de-noised signal for various levels of noise in the measurement data (in the first two columns). We also show a comparison of the vector field obtained using the neural network with the ground truth (in the last two columns). We observe that the approach accurately recovers the clean data from the highly noisy measurements, and accurately predicts the vector field for the model.

C.4 Kuramoto–Sivashinsky equation

In our last test case, we take the data of a chaotic dynamics which are obtained by simulating the Kuramoto–Sivashinsky equation which is of form:

$$\frac{\partial}{\partial t}\mathbf{u}(t,\zeta) = \mathbf{u}(t,\zeta)\mathbf{u}_\zeta(t,\zeta) + \mathbf{u}_{\zeta\zeta}(t,\zeta) + \mathbf{u}_{\zeta\zeta\zeta\zeta}(t,\zeta), \tag{11}$$

where $\mathbf{u}_\zeta$, $\mathbf{u}_{\zeta\zeta}$, and $\mathbf{u}_{\zeta\zeta\zeta\zeta}$ denote the first, second, and fourth derivatives with respect to ζ. The equation explains several physical phenomena such as instabilities of dissipative trapped ion modes in plasma, or fluctuation in fluid films, see, e.g., [54]. We again use the data provided in [53] for the equations which is simulated using a spectral method with $1\,024$ spatial grid points and 251 time-steps. Since the dynamics present in the data is very rich, complex, and exhibits chaotic behavior,

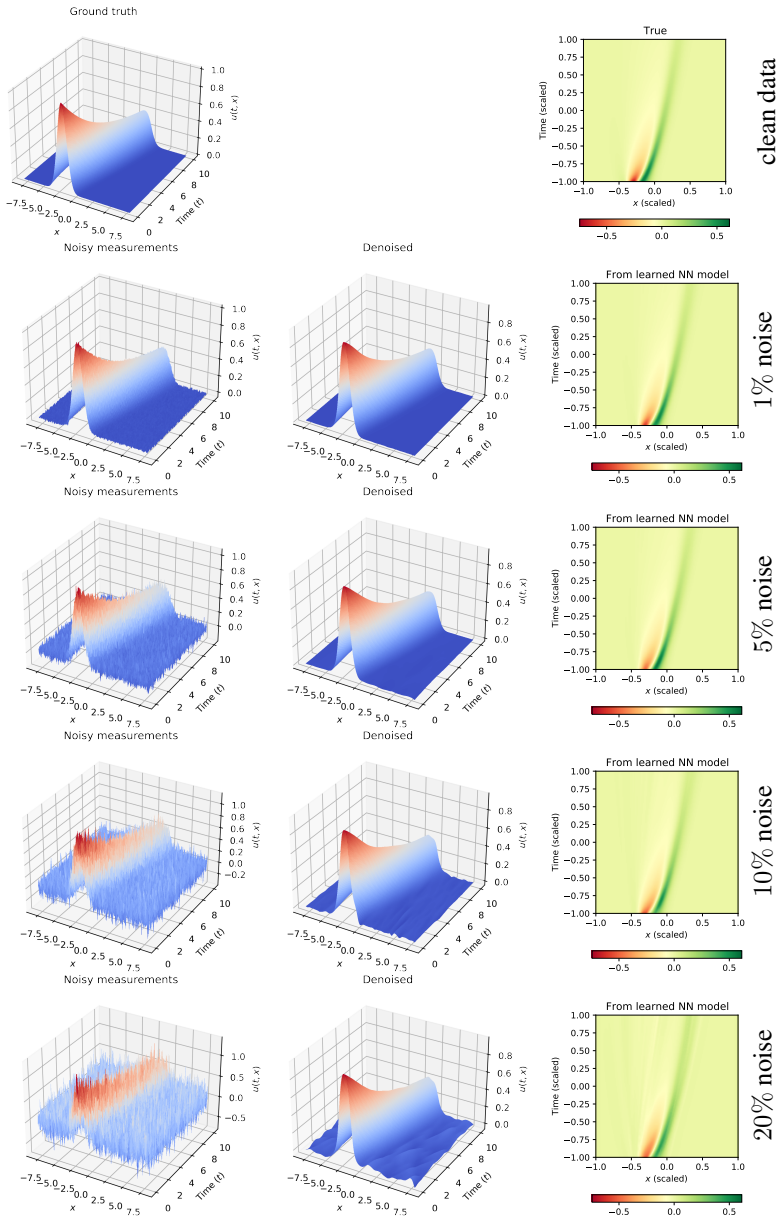

Figure 6: Burgers' equation: The figures show performance of the proposed framework to recover the original spatio-temporal data for various levels of noise. The first and second columns are the noisy and de-noised measurements, and the last column is the prediction of the vector field using the learning convolutional neural network.

we require networks that are more expressive as compared to the previous example; the details about the networks are provided in Table 2.

In Figure 7, we report the ability of our method to remove the noise from the spatial-temporal data. We observe that the proposed methodology profoundly removes noise from the data. Also, the vector field is approximated very well using the learned CNN (see the last column of the figure). The vector field of the clean data is computed using a finite difference method. We draw particular attention to the last row of Figure 7. The algorithm recovers several minor details that are damaged due to the presence of a high-level noise (20%).

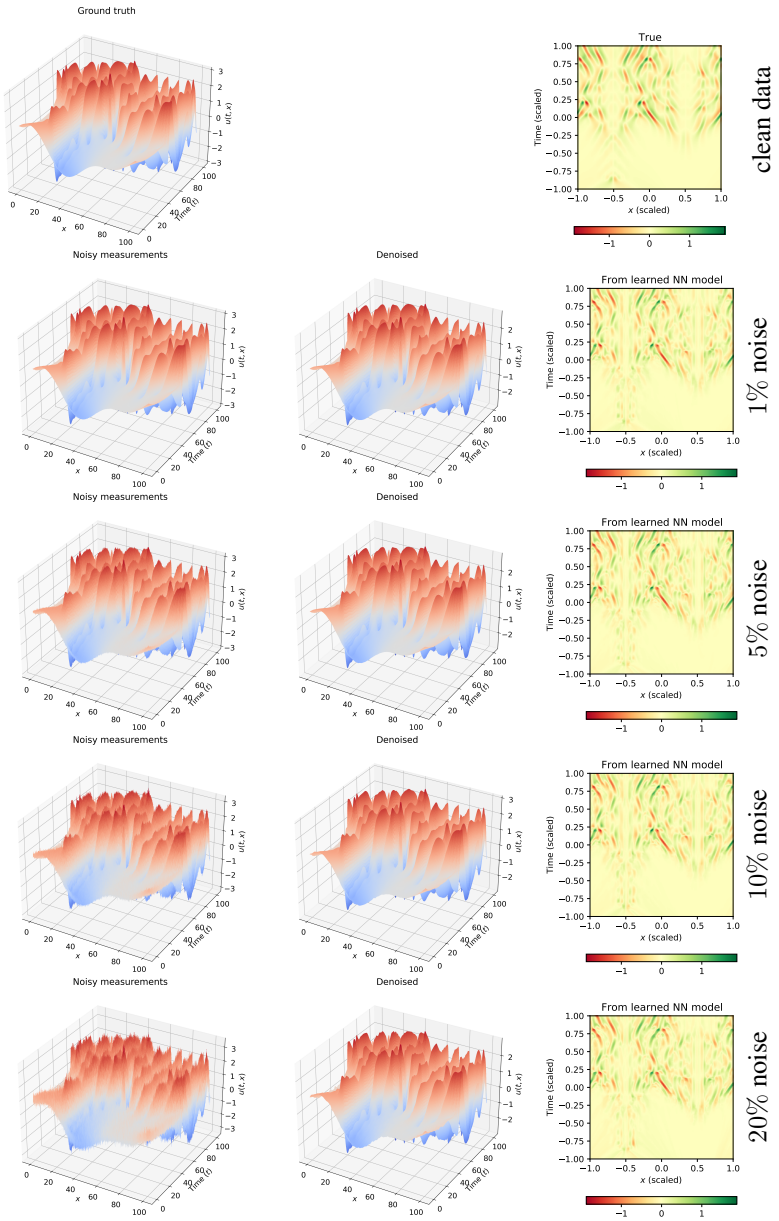

Figure 7: Kuramoto–Sivashinsky equation: The figure shows the noisy measurements (the left column), de-noised measurements (middle column), and the vector field approximation in the right column.