Semi-supervised Learning of Dynamical Systems with Neural Ordinary Differential Equations: A Teacher-Student Model Approach

Yu Wang[1], Yuxuan Yin[1], Karthik Somayaji NS[1], Ján Drgoňa[2], Malachi Schram[4], Mahantesh Halappanavar[2], Frank Liu[5], Peng Li[1]

[1] Department of Electrical and Computer Engineering, University of California, Santa Barbara
[2] Pacific Northwest National Laboratory
[4] Thomas Jefferson National Accelerator Facility
[5] Oak Ridge National Laboratory

{yu95, karthi, y_yin, lip}@ucsb.edu, jan.drgona@pnnl.gov, schram@jlab.org, hala@pnnl.gov, liufy@ornl.gov

Abstract

Modeling dynamical systems is crucial for a wide range of tasks, but it remains challenging due to complex nonlinear dynamics, limited observations, or lack of prior knowledge. Recently, data-driven approaches such as Neural Ordinary Differential Equations (NODE) have shown promising results by leveraging the expressive power of neural networks to model unknown dynamics. However, these approaches often suffer from limited labeled training data, leading to poor generalization and suboptimal predictions. On the other hand, semi-supervised algorithms can utilize abundant unlabeled data and have demonstrated good performance in classification and regression tasks. We propose TS-NODE, the first semi-supervised approach to modeling dynamical systems with NODE. TS-NODE explores cheaply generated synthetic pseudo rollouts to broaden exploration in the state space and to tackle the challenges brought by lack of ground-truth system data under a teacher-student model. TS-NODE employs a unified optimization framework that corrects the teacher model based on the student's feedback while mitigating the potential false system dynamics present in pseudo rollouts. TS-NODE demonstrates significant performance improvements over a baseline Neural ODE model on multiple dynamical system modeling tasks.

Introduction

Precise modeling of system dynamics plays a pivotal role across multiple domains, including system identification, model predictive control, and reinforcement learning. Nonetheless, real-world systems frequently exhibit a high degree of non-linearity within a high-dimensional state space, posing considerable challenges in modeling dynamics without adequate prior knowledge.

Recent advancements in machine learning enable the use of data-driven methods to directly model system dynamics from observations (Greydanus, Dzamba, and Yosinski 2019; Deisenroth, Fox, and Rasmussen 2015; Lusch, Kutz, and Brunton 2018; Chen et al. 2018), effectively circumventing the challenges associated with making appropriate prior model assumptions. For instance, (Deisenroth, Fox, and Rasmussen 2015) employs a Gaussian Process to model

dynamical systems, providing a data-efficient and probabilistic method for learning in the presence of uncertainty and noise. Other studies, such as (Lusch, Kutz, and Brunton 2018; Peitz and Klus 2019), utilize deep neural networks to learn the Koopman operator, which represents a linear embedding of non-linear dynamical systems. Furthermore, Neural Ordinary Differential Equations (NODE)(Chen et al. 2018) leverages a black-box neural network to model dynamics as an ordinary differential equation (ODE), demonstrating significant potential in modeling unknown dynamical systems.

However, one significant obstacle for data-driven methods stems from the practical difficulty in acquiring sufficient data for modeling dynamical systems. The data collection process can be expensive, time-consuming, and in some cases, even infeasible due to physical constraints or safety considerations. The shortage of data results in limited coverage of the state space, leading to gaps in the model's understanding and poor prediction ability.

Semi-supervised learning methods have emerged as a promising solution to the problem of limited labeled data in classification and regression tasks. These approaches effectively leverage a large amount of unlabeled data along with some labeled data. Unlabeled data can be either obtained directly from observations with unknown or uncollected labels or generated through various data augmentation techniques (Chen et al. 2020; Cubuk et al. 2019, 2020). Semi-supervised methods typically assume that unlabeled data share the same structural information as labeled data and often utilize them in different ways, such as self-training (Pham et al. 2021), multi-view training (Chen et al. 2020; Wang and Isola 2020), or based on graph methods (Kipf and Welling 2016; Iscen et al. 2019). These approaches promote the model to produce similar outputs for similar instances or preserve the structure of the data in the learned representation.

Thus, it is logical to consider the integration of semi-supervised learning into the modeling of dynamical systems with neural ODEs, a concept yet to be explored by the research community. However, the adaptation of semi-supervised learning strategies to the problem is not straightforward and presents several challenges: the *definition, generation* and *utilization* of unlabeled data in dynamical sys-
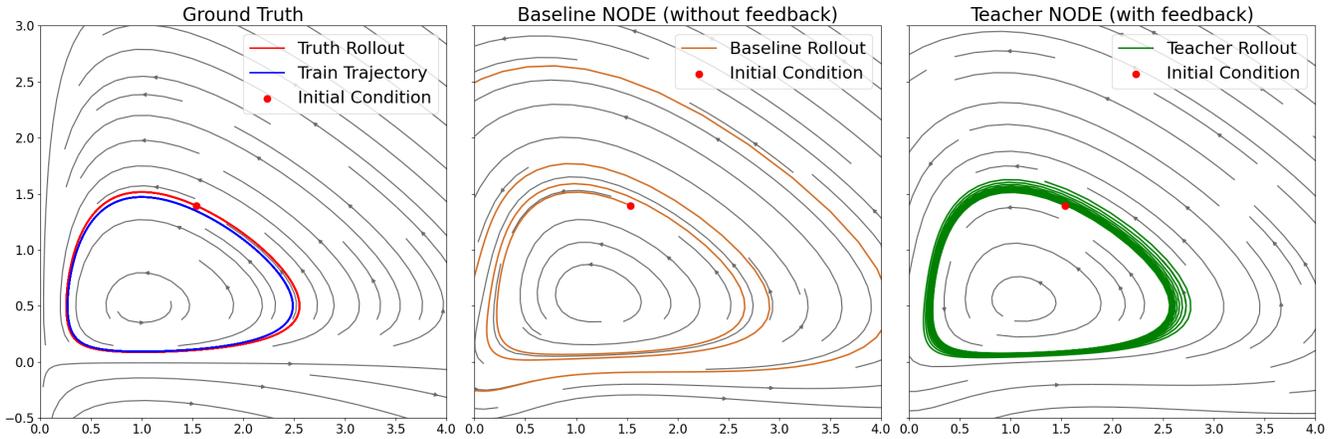
Figure 1: Phase portraits and rollouts on a test initial condition of different learned models in the phase space of a Lotka-Volterra system(Lotka 1909). Left: The ground truth. Middle: the baseline neural ODE. Right: the proposed teacher model with student feedback. All models are trained with a single train trajectory (blue line in the left).

tem modeling significantly differ from their counterparts in other semi-supervised learning contexts such as classification and regression. For instance, generating additional dynamical training data by directly applying popular data augmentation and multi-view learning techniques may be detrimental, as alternating truthfully observed system data can inject false system dynamics.

To address these challenges, we introduce TS-NODE, a semi-supervised learning approach designed specifically for modeling dynamical systems with neural ODEs. Our contributions are:

- We present the first teacher-student model based approach to enabling semi-supervised modeling of dynamical systems;

- We explore synthetic *pseudo rollouts* generated by the teacher model to broaden exploration in the state space and to tackle the challenges brought by lack of ground-truth system data;

- We employ an unified teacher-student optimization framework that corrects the teacher model based on the student's feedback while mitigating the potential false system dynamics present in pseudo trajectories;

- We evaluate TS-NODE on multiple dynamical system modeling tasks, demonstrating significant performance improvements over a baseline Neural ODE model.

Preliminaries

Neural ODE for Modeling Dynamical Systems

A neural ODE parameterized by $\boldsymbol{\theta}$ models the dynamics of the evolution of the state $\boldsymbol{y}(t)$ of an unknown system with a neural network(Chen et al. 2018):

$$\dot{\boldsymbol{y}}(t) = \text{NN}(\boldsymbol{y}(t), t; \boldsymbol{\theta}). \quad (1)$$

Let $\boldsymbol{y} = \{\boldsymbol{y}(t_0), \boldsymbol{y}(t_1) \cdots \boldsymbol{y}(t_n)\}$ be an observed system trajectory in the state space over a period of time and de-

note its initial condition[1] by $\boldsymbol{y_0}$. A neural ODE is trained to learn the system dynamics by matching its predicted trajectory (rollout) with the true trajectory. The neural ODE's predicted trajectory is obtained by solving an initial value problem (IVP) for each $t_i, i = [0, 1 \cdots n]$:

$$\hat{\boldsymbol{y}}(t_i) = \boldsymbol{y}(t_0) + \int_{t_0}^{t_i} \text{NN}(\hat{\boldsymbol{y}}(\tau), \tau; \boldsymbol{\theta})d\tau \quad (2)$$

The parameters $\boldsymbol{\theta}$ are optimized by minimizing the mean squared error (MSE) between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{y}; \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \text{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}}(\boldsymbol{\theta})). \quad (3)$$

Challenges in Semi-supervised Modeling of Dynamical Systems with Neural ODEs

Training neural ODEs requires a substantial amount of data for optimal generalization. On the other hand, data for certain systems like a complex control system can be costly to collect and the observed trajectories often have limited coverage of the state space. Scarcity of data results in poor generalization and suboptimal long-horizon rollouts of the trained neural ODE models. The ability of making use of large amounts of cheap unlabeled data makes semi-supervised learning particularly appealing for dynamical system modeling. Nevertheless, semi-supervised learning has not been attempted for neural ODE based dynamical system modeling before. It is not immediately evident how to pursue semi-supervised learning under this new context, raising several crucial questions we outline below:

Q1: What does *unlabeled data* mean for dynamical system modeling? The concept of unlabeled data in classical semi-supervised learning(Ouali, Hudelot, and Tami 2020a,b), e.g. for classification and regression tasks is well defined: unlabeled data are input examples for which there

---

[1]In later sections, $\boldsymbol{y}$ refers to a time series and $\boldsymbol{y_0}$ represents its value at the first time point unless stated otherwise.

exist no ground-truth output label. However, when it comes to modeling dynamics with Neural ODEs, this definition becomes less clear as the system model is trained on observed system trajectories rather than a set of input/output pairs with or without the output label.

Q2: How to generate *useful* unlabeled data for dynamical system modeling? Data augmentation has been popular for introducing new training experiences without added data labeling effort in many learning tasks. In most of cases, it works on the premise that subtle input perturbations do not change or substantially alter the output labels. A brute-force application of existing data augmentation techniques such as crop-and-resize(Chen et al. 2020) or random erasing(Zhong et al. 2017) to dynamic system modeling may not be completely appropriate: these augmentation transformations either cannot directly operate on data that are dynamical in nature, or fail to introduce meaningful new experiences that are consistent with the underlying dynamics to be modeled, leading to degraded learning performance.

Q3: How to mitigate *misinformation* in the unlabeled data? Any generated unlabeled data are inevitable to exhibit dynamical behavior that deviates from the system under modeling. It is imperative to prevent the false dynamical information from such data to leak into the system model.

TS-NODE Framework Overview

We argue that the proposed teacher-student model based TS-NODE approach is well positioned to address the issues raised in aforementioned the three questions.

First, under our context data that are useful must exhibit dynamical behavior relevant to the task of dynamical system learning. As such, meaningful "unlabeled data" can be considered to be state trajectories that do not necessarily fully reflect but bear resemblance to the underlying true system dynamics. We call such trajectories *pseudo rollouts*.

Second, to best aid dynamical system modeling, the employed pseudo rollouts shall comply with the typical physical constraints of the family of systems to which the targeted system belongs. In other words, augmenting known system trajectories via unconstrained non-physical transformations provides little value if it does not hamper system modeling. To this end, it is most straightforward to generate pseudo rollouts, a.k.a. unlabeled data, by a dynamical system model that is similar to the system under modeling.

TS-NODE achieves this by making use of a teacher neural ODE model, which is trained with (limited) true system trajectories, to generate pseudo rollouts, potentially covering part of the state space with little or no true system dynamical data. The generated trajectories, or pseudo rollouts, encapsulate the teacher's present belief about the underlying dynamics and provide contextual awareness of the target problem. Once generated, they can be readily used to train the student, another neural ODE.

Lastly, directly utilizing the pseudo rollouts in the final system model building can lead to undesired amplification of inaccurate knowledge. Instead, TS-NODE utilizes these trajectories to train the student model whose performance on a validation dataset acts as a feedback to the teacher to correct the teacher's belief.
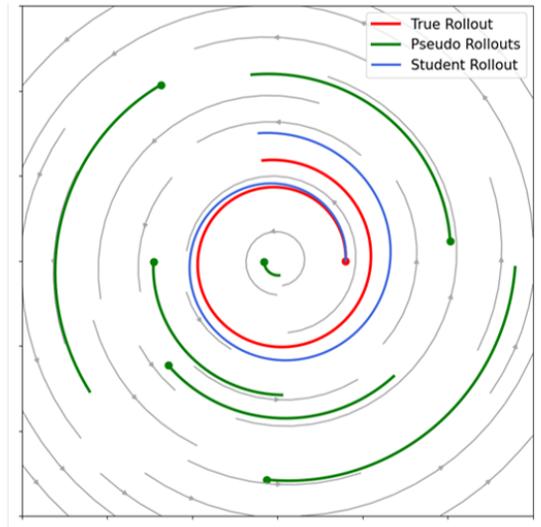
Figure 2: An illustrated overview of the learning mechanism of TS-NODE. The teacher model generates pseudo rollouts (green lines) with more coverage of the state space. The student model learns from those trajectories and then predict rollout (blue line) with the initial condition of the true trajectory (red line). The MSE between student's prediction and the ground truth serves as the feedback signal to refine the teacher's prediction.

Basic Design of TS-NODE Model   With the same high-level spirit of the semi-supervised learning technique for image classification (Pham et al. 2021), TS-NODE jointly optimizes the teacher and student model at each training step. First, pseudo rollouts starting from a set of initial conditions in the state space are generated by the teacher, which are used to train the student neural ODE whose performance is evaluated on the true system trajectories. The evaluation loss of the student, called the student feedback loss, is considered as a feedback to the teacher as it depends on the teacher's pseudo rollouts. The teacher is then updated by minimizing the student feedback loss and its direct fitting loss over the limited true system trajectories, a.k.a. the labeled dataset.

Basic Learning Flow of TS-NODE

Following the standard definitions in semi-supervised learning, we denote the labeled data, i.e. true system dynamical trajectories by: $\mathcal{D}^l := \left\{ y^l \right\}$. We denote the unlabeled data by the pseudo rollouts predicted from the teacher: $\mathcal{D}^u := \left\{ y^u \right\}$, with their initial conditions denoted by y_0^u. Two NODEs are introduced in TS-NODE, namely the teacher: $T(\cdot; \theta_T)$ and the student: $S(\cdot; \theta_S)$.

The training of TS-NODE involves four major steps:

1. Generate pseudo rollouts from the teacher:

$$\boldsymbol{y_T^u}(\boldsymbol{\theta_T}) = T(\boldsymbol{y_0^u}; \boldsymbol{\theta_T}) \tag{4}$$

2. Train the student[2] with an unlabeled loss ($\mathcal{L}_S^u$) defined on pseudo rollouts:

$$\mathcal{L}_S^u = \mathrm{MSE}\left(\boldsymbol{y_T^u}(\boldsymbol{\theta_T}),\ S(\boldsymbol{y_0^u}, \boldsymbol{\theta_S})\right) \tag{5}$$

$$\boldsymbol{\theta'_S} = \boldsymbol{\theta_S} - \eta_S \cdot \nabla_{\boldsymbol{\theta_S}} \cdot \mathcal{L}_S^u \tag{6}$$

3. Calculate the error of the updated student model on $\mathcal{D}^l$ as the feedback loss $\mathcal{L}_F^l$:

$$\mathcal{L}_F^l := \mathcal{L}_{S'}^l = \mathrm{MSE}(\mathcal{D}^l, S'(\boldsymbol{y_0^l}; \boldsymbol{\theta'_S}(\boldsymbol{\theta_T}))) \tag{7}$$

4. Train the teacher with both the labeled loss $\mathcal{L}_T^l$ and the feedback loss $\mathcal{L}_F^l$:

$$\mathcal{L}_T^l = \mathrm{MSE}(\mathcal{D}^l, T(\boldsymbol{y_0^l}; \boldsymbol{\theta_T})) \tag{8}$$

$$\boldsymbol{\theta'_T} = \boldsymbol{\theta_T} - \eta_T \cdot \nabla_{\boldsymbol{\theta_T}} \cdot (\mathcal{L}_T^l + \mathcal{L}_F^l) \tag{9}$$

Gradient for Updating the Teacher and Its Scalable Computation

To update the teacher with the feedback loss involves computing a Jacobian matrix or calculating per-sample gradients, which can result in a large computational graph and computational inefficiency (More details in Appendix A). Instead, we make a modification to the framework by employing a noisy teacher to generate the pseudo rollouts. Utilizing a noisy teacher enables the derivation of scalable gradients using the REINFORCE rule(Williams 1992) and more interpretable, alternative feedback loss.

Noisy Teacher: We design a simple noisy teacher T_ϵ by adding independent constant noise to each step of the rollout of the standard teacher. The rollouts of T_ϵ are thus samples from a Gaussian distribution:

$$\boldsymbol{y_{T_\epsilon}^{n \times d}} \sim \mathcal{N}(\boldsymbol{y_T^{n \times d}}; \sigma \cdot \boldsymbol{I}^{(n \times d) \times (n \times d)}; \boldsymbol{\theta_T}) \tag{10}$$

where n is the number of observed time steps, d is the dimension of the state space.

Derivation of the Feedback Gradient: For a noisy teacher, since the pseudo rollouts are samples from a distribution, Eq. (6) needs to be rewritten as:

$$\boldsymbol{\theta'_S} = \boldsymbol{\theta_S} - \eta_S \cdot \mathbb{E}_{\boldsymbol{y_{T_\epsilon}^u} \sim p(\boldsymbol{y_{T_\epsilon}^u})}[\nabla_{\boldsymbol{\theta_S}} \cdot \mathcal{L}_S^u] \tag{11}$$

For convenience, we write the gradient to update the student as $\boldsymbol{g_S} := \nabla_{\boldsymbol{\theta_S}} \cdot \mathcal{L}_S^u$, the gradient of $\mathcal{L}_F^l$ with respect to $\boldsymbol{\theta_T}$ as $\boldsymbol{g_F} := \nabla_{\boldsymbol{\theta_T}} \cdot \mathcal{L}_F^l$. By the chain rule , we have:

$$[\boldsymbol{g_F}]^T = \frac{\partial \mathcal{L}_F^l}{\partial \boldsymbol{\theta_T}} = \frac{\partial \mathcal{L}_F^l}{\partial \boldsymbol{\theta'_S}} \cdot \frac{\partial \boldsymbol{\theta'_S}}{\partial \boldsymbol{\theta_T}} \tag{12}$$

By substituting the $\boldsymbol{\theta'_S}$ in Eq. (11), we have:

$$[\boldsymbol{g_F}]^T = -\eta_S \cdot \frac{\partial \mathcal{L}_F^l}{\partial \boldsymbol{\theta'_S}} \cdot \underbrace{\frac{\partial}{\partial \boldsymbol{\theta_T}}\left(\mathbb{E}_{\boldsymbol{y_{T_\epsilon}^u} \sim p(\boldsymbol{y_{T_\epsilon}^u})}[\boldsymbol{g_S}]\right)}_{\text{jacobian}} \tag{13}$$

---

[2]We follow the suggestion in (Pham et al. 2021) to update the student with one-step gradient descent.

By REINFORCE rule(Williams 1992), we rewrite the Jacobian as :

$$\frac{\partial}{\partial \boldsymbol{\theta_T}}\left(\mathbb{E}_{\boldsymbol{y_{T_\epsilon}^u} \sim p(\boldsymbol{y_{T_\epsilon}^u})}[\boldsymbol{g_S}]\right) = \boldsymbol{g_S} \cdot \frac{\partial \log p(\boldsymbol{y_{T_\epsilon}^u})}{\partial \boldsymbol{\theta_T}} \tag{14}$$

Finally, the feedback gradient is derived as:

$$[\boldsymbol{g_F}]^T = -\eta_S \cdot \underbrace{\frac{\partial \mathcal{L}_F^l}{\partial \boldsymbol{\theta'_S}}}_{a} \cdot \underbrace{\boldsymbol{g_S}}_{b} \cdot \underbrace{\frac{\partial \log p(\boldsymbol{y_{T_\epsilon}^u})}{\partial \boldsymbol{\theta_T}}}_{c} \tag{15}$$

In Eq. (15), all three parts can be calculated efficiently with one step of automatic differentiation.

Approximated Alternative Feedback Loss: We show that Eq. (15) can be further simplified by utilizing Taylor expansion. An approximated alternate feedback loss can be derived which can give us a direct interpretation of how the feedback loss helps to refine teacher's pseudo rollouts.

First, we rewrite $\boldsymbol{g_F}$ by applying Taylor expansion to $\mathcal{L}_F^l(\boldsymbol{\theta'_S})$ and ignoring higher order terms:

$$\mathcal{L}_F^l(\boldsymbol{\theta'_S}) \approx \mathcal{L}_F^l(\boldsymbol{\theta_S}) + \frac{\partial \mathcal{L}_F^l(\boldsymbol{\theta'_S})}{\partial \boldsymbol{\theta'_S}} \cdot (\boldsymbol{\theta'_S} - \boldsymbol{\theta_S}) \tag{16}$$

Combining Eq. (16) and Eq. (6) we have:

$$\begin{aligned}
\mathcal{L}_F^l(\boldsymbol{\theta'_S}) - \mathcal{L}_F^l(\boldsymbol{\theta_S}) &\approx \frac{\partial \mathcal{L}_F^l(\boldsymbol{\theta'_S})}{\partial \boldsymbol{\theta'_S}} \cdot (\boldsymbol{\theta'_S} - \boldsymbol{\theta_S}) \\
&= \frac{\partial \mathcal{L}_F^l(\boldsymbol{\theta'_S})}{\partial \boldsymbol{\theta'_S}} \cdot (-\eta_S \cdot \boldsymbol{g_S})
\end{aligned} \tag{17}$$

Thus, the feedback gradient can be approximated as:

$$\begin{aligned}
[\boldsymbol{g_F}]^T &\approx \left(\mathcal{L}_F^l(\boldsymbol{\theta'_S}) - \mathcal{L}_F^l(\boldsymbol{\theta_S})\right) \cdot \frac{\partial \log p(\boldsymbol{y_{T_\epsilon}^u})}{\partial \boldsymbol{\theta_T}} \\
&= \left(\mathcal{L}_{S'}^l - \mathcal{L}_S^l\right) \cdot \frac{\partial \log p(\boldsymbol{y_{T_\epsilon}^u})}{\partial \boldsymbol{\theta_T}}
\end{aligned} \tag{18}$$

Note that the feedback gradient in Eq. (18) corresponds to the gradient of a scaled negative log-likelihood (NLL) loss and we can rewrite the original feedback loss as:

$$\mathcal{L}_F^l \approx \underbrace{(\mathcal{L}_S^l - \mathcal{L}_{S'}^l)}_{h:\text{student's improvement}} \cdot \ \mathrm{NLL}(\boldsymbol{y_{T_\epsilon}^u}; \boldsymbol{\theta_T}) \tag{19}$$

The alternate feedback loss refines the noisy teacher by modulating the likelihood of generating particular pseudo rollouts based on the student's improvement. After trained with given a rollout, if the student's performance sees an enhancement (i.e., $h \geq 0$), minimizing $\mathcal{L}_F^l$ effectively maximizes the log-likelihood of that rollout, thereby augmenting the probability that the teacher will generate similar rollouts in future iterations. Conversely, if the student's performance deteriorates, the feedback mechanism will work to diminish the likelihood of generating such rollouts.

In our experiments, optimization of TS-NODE follows the procedure from Eq. (4) to Eq. (9). The original feedback loss in Eq. (7) is replaced with the one defined in Eq. (19) for efficient computation. We summarize the pseudo code for TS-NODE in Appendix B for further reference.

Experiments

We evaluate TS-NODE on several aspects: 1) We compare TS-NODE with the baseline NODE to validate its overall efficacy; 2) We show TS-NODE provides a better way of generating and utilizing unlabeled data by compared with NODE trained with basic data augmentations. 3) We compare TS-NODE with a variant that disables the student's feedback to underscore the necessity of the feedback loop.

Datasets and Model Configuration

We adopt three dynamical systems, the converging cubic system (Chen et al. 2018), Lotka-volterra system (Lotka 1909) and a simple pendulum, which are commonly in the literature, as the test cases. We simulate each ground-truth system to generate a single state trajectory from a predetermined initial condition as the only labeled training data, to evaluate the baseline (Chen et al. 2018) and proposed TS-NODE under limited observation. To evaluate the generalization of each learned model, we randomly select a set of 20 initial conditions in the state space, and simulate the ground-truth model to obtain 20 state trajectories as the testing data, as detailed next.

Converging Cubic System: The converging cubic system we employed is a 2-dimensional system, adapted from (Chen et al. 2018). The system dynamics is described as below:

$$\dot{x} = a \cdot x^3 + b \cdot y^3$$
$$\dot{y} = c \cdot x^3 + d \cdot y^3 \tag{20}$$

We set $[a, b, c, d] = [0.1, 2, -2, -0.1]$. The training trajectory is a rollout from $[x_0, y_0] = [3, -1]$. The testing initial conditions are sampled from $\mathcal{N}([x_0, y_0]; 0.3 \cdot \boldsymbol{I}^{2 \times 2})$. The simulation time is 10s.

Lotka-Volterra System : The Lotka-Volterra system(Lotka 1909), or the predator-prey system, is dictated by the dynamics as:

$$\dot{x} = \alpha \cdot x - \beta \cdot xy$$
$$\dot{y} = \delta \cdot xy - \gamma \cdot y \tag{21}$$

We set $[\alpha, \beta, \delta, \gamma] = [\frac{2}{3}, \frac{4}{3}, 1, 1]$. The training trajectory is a rollout from $[x_0, y_0] = [1.4, 1.4]$ and the testing initial conditions are sampled from $\mathcal{N}([x_0, y_0]; 0.3 \cdot \boldsymbol{I}^{2 \times 2})$. The simulation time is 10s.

Simple Pendulum: A simple Pendulum is modeled as:

$$\ddot{\theta} = -a \cdot \dot{\theta} - b \cdot \sin\theta \tag{22}$$

Simple pendulumn can be fully described by the angular position θ and the angular speed $\dot{\theta} := \omega$. In our experiments, we set $[a, b] = [0, 1]$, (i.e., simulating an ideal pendulum). The training trajectory is a rollout from$[\theta_0, \omega_0] = [2, 0]$. The initial conditions for generating the testing data are sampled from $\mathcal{N}([\theta_0, \omega_0]; 0.3 \cdot \boldsymbol{I}^{2 \times 2})$. The simulation time is 10s.

The length of training or testing trajectories is 1,000 time steps. We split the training trajectory as batches of short observations (10 steps) for efficiency as in (Chen et al. 2018). The initial conditions used by the teacher to generate pseudo rollouts are sampled around the training trajectory in each iteration.

Baseline and Evaluation Metrics:

We compare TS-NODE with the standard neural ODE in (Chen et al. 2018), as the baseline. Neural networks in the baseline and TS-NODE share the identical architecture.

We randomly select 20 initial conditions and simulate each ground-truth system to generate 20 1000-step full test trajectories.

We adopt two evaluation metrics for our study. The first, referred to as *local error*, assesses the quality of local dynamic modeling around the testing regions. We randomly sample parts of the 20 full-length testing trajectories to create a set of 1,000 short testing rollouts of 10 steps. The local error is defined as the sum of the MSE between these testing rollouts and the corresponding rollouts from each learned model starting from the same initial condition.

The *rollouts error* metric, measures the models' ability to forecast over different lengths of time horizon. We let learned models generate rollouts from the initial condition of each pre-generated full testing trajectory for 5%, 10%, 20%, 50%, and 100%, respectively of the full length (1,000 steps). The rollouts error of a particular length is computed as the average MSE between the ground truth model and each learned model over the 20 rollouts of that length.

1) Comparing TS-NODE with baseline NODE: We train TS-NODE and the baseline neural ODE with Adam optimizer for 10,000 iterations. We record the error metrics of the two models once every 100 iterations. Table 1 reports the performances of two models based on the averages of the last five recorded errors.

For all systems, TS-NODE consistently surpasses the baseline model across all metrics, particularly when making long-horizon predictions. The baseline model exhibits a significant error in generating extended rollouts for the LK system and the pendulum, suggesting its inability to accurately generalize the precise dynamics of these systems, leading to accumulation and subsequent explosion of error over time.

To further illustrate this point, we provide visual representations of the rollouts of varying lengths, as predicted by both the final trained baseline model and the teacher model of TS-NODE, originating from a single test initial condition (see Fig. 3). It is evident that even in scenarios where the baseline model fails in producing accurate predictions, TS-NODE remains capable of accurately capturing and replicating the inherent dynamics over an extended duration.

We also plot the rollouts error during the training process for the baseline and the teacher in Fig. 4. We take the logarithm of the rollouts error and apply a moving average to smooth the curve. The scarcity of the data results in a large rollout error of the baseline, which cannot be reduced or can even increase with extended training. In contrast, TS-NODE consistently shows better test performance across varying rollout lengths as the training proceeds, clearly demonstrating the overall effectiveness of TS-NODE framework.

2) Comparing TS-NODE with NODEs trained by basic data augmentations: We investigated the effects of two common data augmentation techniques: the addition of white noise ($\sigma = 0.01$) and rescaling by 0.95X. We apply them to the labeled dataset to generate an unlabeled

Dataset	Models	Local Error↓	Rollouts Error↓				
			5%	10%	20%	50%	100%
Cubic	Baseline	39.84	0.95	1.35	1.21	1.42	1.08
	TS-NODE	**32.77**	**0.87**	**1.21**	**1.07**	**1.12**	**0.96**
LK	Baseline	20.93	0.10	2.54	64.79	1200.50	7346.09
	TS-NODE	**12.84**	**0.02**	**0.14**	**0.55**	**20.58**	**244.88**
Pendulum	Baseline	40.49	0.06	0.20	0.51	5.77	507.17
	TS-NODE	**5.65**	**0.01**	**0.05**	**0.25**	**0.95**	**1.89**

Table 1: Local error and rollouts error (both lower the better) of the baseline NODE and the TS-NODE on three systems.
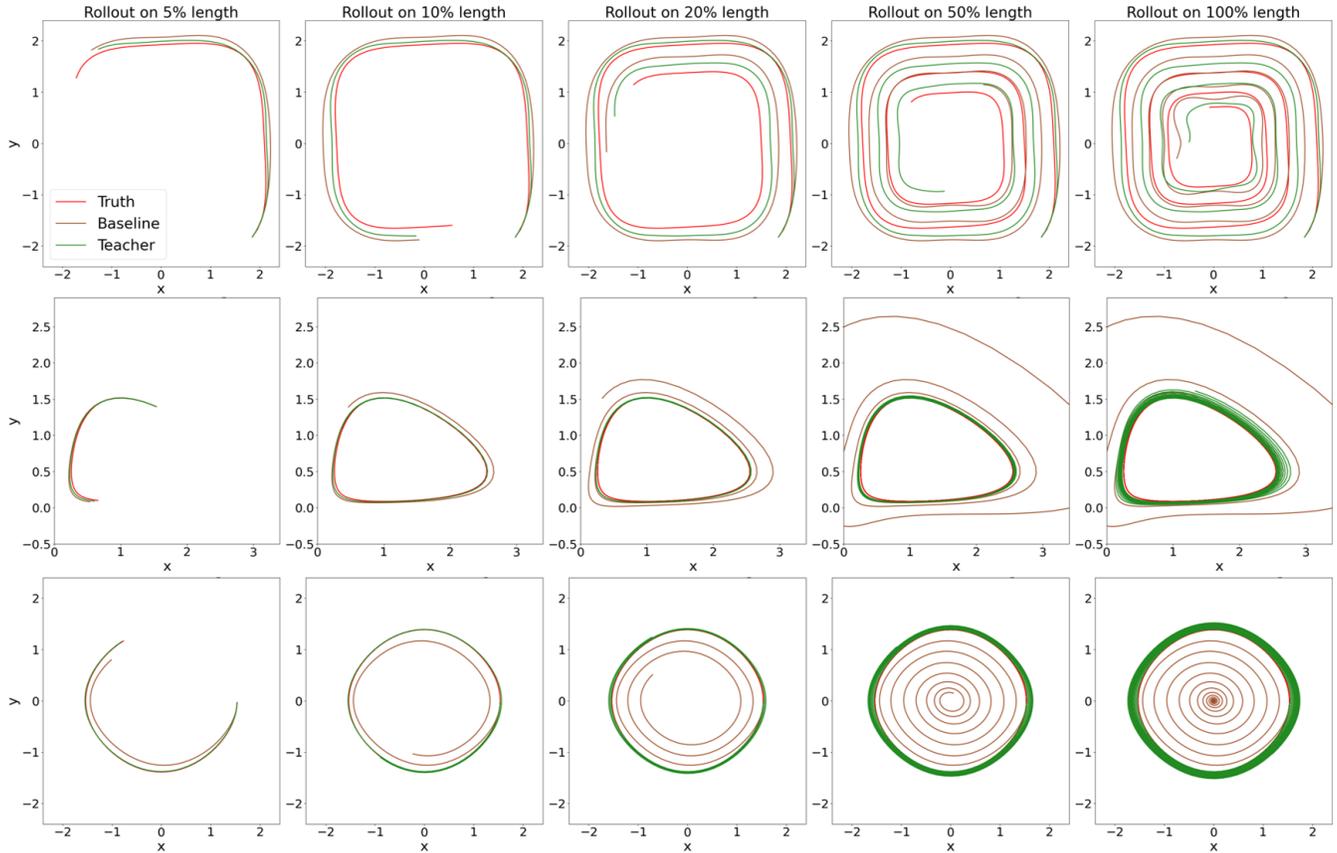
Figure 3: Rollouts from the trained baseline model and teacher model of TS-NODE in the phase space for three different systems. The rollouts are simulated from the initial condition of one pre-generated full testing trajectory for 5%, 10%, 20%, 50%, and 100%, of the full length (1,000 steps). Top: converging cubic system. Middle: Lotka-Volterra system. Bottom: pendulum.

dataset and combine both datasets to train the standard neural ODEs. We denote the models trained with different augmentations by "White Noise" and "Re-scale", respectively. Experiment results on the LK dataset are shown in Table 2.

TS-NODE surpasses all models trained using basic data augmentations, which can potentially compromise the dataset's intrinsic physical properties. This performance underscores that TS-NODE offers a superior approach to generating and utilizing unlabeled data through its pseudo rollouts and the feedback loop.

3) The necessity of the feedback loop: We disable the feedback loop by training the teacher only with labeled loss and maintaining the student's learning from teacher-generated pseudo rollouts. The performance of the final student is tabulated as "No Feedback" in Table 2.

Without the feedback signal, the misinformation in the pseudo rollouts cannot get exposed via students evaluation, and the teacher cannot refine itself to generate better pseudo rollouts, resulting a bad-behaved final student. With the feedback signal, TS-NODE attains substantially superior results, highlighting the critical role of the feedback loop.
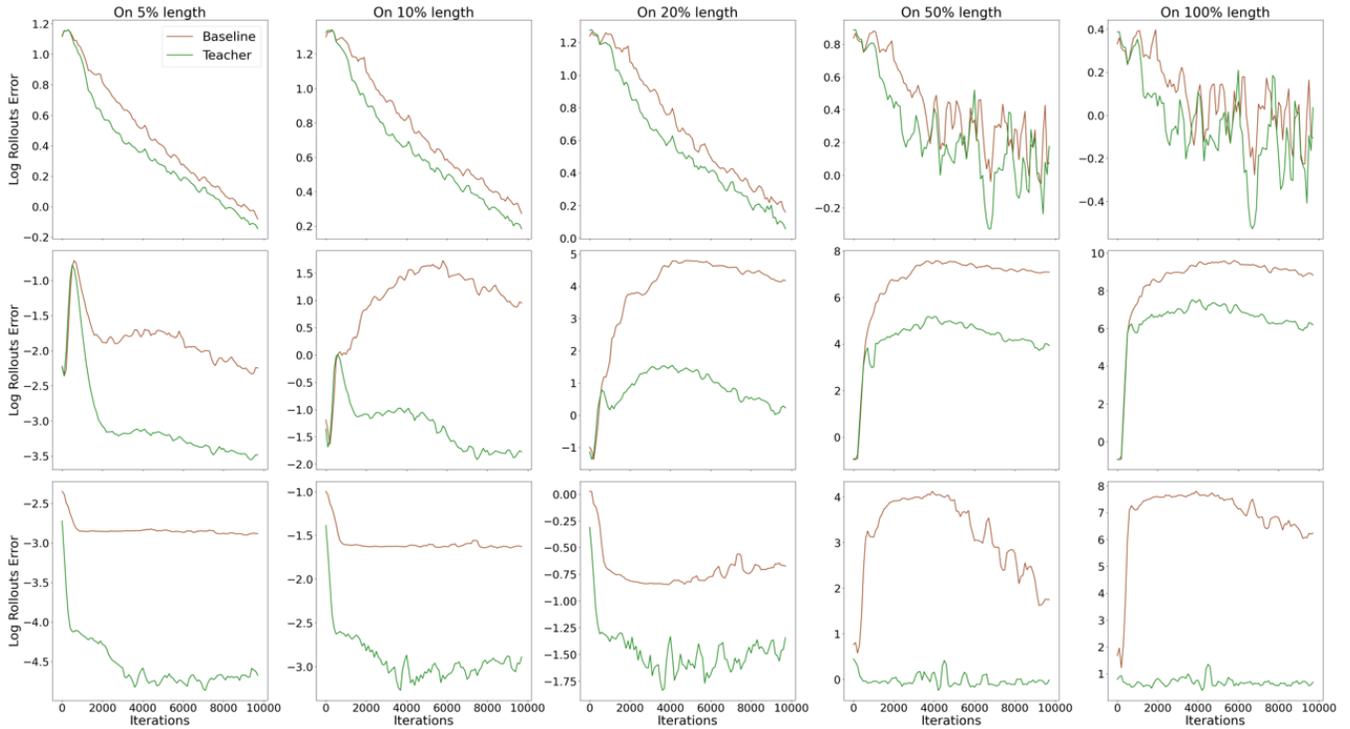
Figure 4: The rollouts errors of different length (5%, 10%, 20%, 50%, and 100% of the full length (1000 steps)) of the baseline model and the teacher model. Top: converging cubic system. Middle: Lotka-Volterra system. Bottom: simple pendulum.

Models	Local Error$^\downarrow$	Rollouts Error$^\downarrow$				
		5%	10%	20%	50%	100%
Baseline	20.93	0.10	2.54	64.79	1200.50	7346.09
TS-NODE	**12.84**	**0.02**	**0.14**	**0.55**	**20.58**	**244.88**
White Noise	23.78	0.05	0.19	1.02	42.06	306.50
Re-scale	106.97	0.01	1.09	158.53	4398.57	31925.17
No Feedback	23.60	0.06	0.23	2.00	192.03	2057.37

Table 2: Result on LK dataset of the Baseline, TS-NODE, standard NODE trained with different augmentations, and the student in a TS-NODE without feedback.

Related Works

Neural ODEs (Chen et al. 2018) (NODEs) show great potential in modeling complex, unknown dynamical systems. However, they can be challenging to train effectively, particularly when data is limited. Subsequent research works(Dupont, Doucet, and Teh 2019; Finlay et al. 2020; Ghosh et al. 2020) attempt to enhance the generalization of NODEs by introducing augmented dimensions, kinetic regularization, or temporal perturbations. These approaches aim to refine the structure of NODEs without directly addressing the issue of data scarcity. Parallel line of works (Raissi et al. 2018; Greydanus, Dzamba, and Yosinski 2019) tackle the problem by incorporating prior physics knowledge, e.g., symmetry, into modeling of dynamical systems with neural networks.

Overall, semi-supervised learning has not been thoroughly explored for dynamical system modeling. Data augmentation (Chen et al. 2020; Cubuk et al. 2019, 2020) are not fully appropriate in this context as they can disrupt the underlying dynamical information. Among all recent works, (Pham et al. 2021) most closely resembles TS-NODE. However, (Pham et al. 2021) focuses on classification tasks where unlabeled data are readily available, while TS-NODE meaningfully generates and utilizes unlabeled data for dynamical systems. This contextual awareness is crucial for effective use of unlabeled data for system modeling tasks.

Limitations and Future Work

Currently, the pseudo rollouts are simply generated around the training trajectory in a suboptimal manner, i.e., we do not optimize the locations to generate pseudo rollouts during training. An intriguing future direction is to incorporate uncertainty quantification into the framework. This could allow the model to prioritize exploration and evaluation of highly uncertain regions within the state space. By actively generating pseudo rollouts in these areas of uncertainty, TS-NODE could effectively streamline its exploration process, enhancing its understanding of the state space. Uncertainty-guided exploration may prove particularly beneficial when dealing with high-dimensional problems, where efficiently covering the state space with pseudo rollouts poses significant challenges. We leave this open problem to future exploration.

Acknowledgments

References

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A Simple Framework for Contrastive Learning of Visual Representations. *CoRR*, abs/2002.05709.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. 2020. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 18613–18624. Curran Associates, Inc.

Deisenroth, M. P.; Fox, D.; and Rasmussen, C. E. 2015. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2): 408–423.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented Neural ODEs. arXiv:1904.01681.

Finlay, C.; Jacobsen, J.-H.; Nurbekyan, L.; and Oberman, A. M. 2020. How to train your neural ODE: the world of Jacobian and kinetic regularization. arXiv:2002.02798.

Ghosh, A.; Behl, H. S.; Dupont, E.; Torr, P. H. S.; and Namboodiri, V. P. 2020. STEER : Simple Temporal Regularization For Neural ODEs. *CoRR*, abs/2006.10711.

Greydanus, S.; Dzamba, M.; and Yosinski, J. 2019. Hamiltonian Neural Networks. *CoRR*, abs/1906.01563.

Iscen, A.; Tolias, G.; Avrithis, Y.; and Chum, O. 2019. Label Propagation for Deep Semi-supervised Learning. *CoRR*, abs/1904.04717.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907.

Lotka, A. J. 1909. Contribution to the Theory of Periodic Reactions.

Lusch, B.; Kutz, J. N.; and Brunton, S. L. 2018. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1).

Ouali, Y.; Hudelot, C.; and Tami, M. 2020a. An Overview of Deep Semi-Supervised Learning. *CoRR*, abs/2006.05278.

Ouali, Y.; Hudelot, C.; and Tami, M. 2020b. An Overview of Deep Semi-Supervised Learning. *CoRR*, abs/2006.05278.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR*, abs/1912.01703.

Peitz, S.; and Klus, S. 2019. Koopman operator-based model reduction for switched-system control of PDEs. *Automatica*, 106: 184–191.

Pham, H.; Dai, Z.; Xie, Q.; and Le, Q. V. 2021. Meta Pseudo Labels. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11552–11563.

Raissi, M.; Wang, Z.; Triantafyllou, M. S.; and Karniadakis, G. E. 2018. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861: 119–137.

Wang, T.; and Isola, P. 2020. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *International Conference on Machine Learning*, 9929–9939. PMLR.

Williams, R. J. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8: 229–256.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2017. Random Erasing Data Augmentation. arXiv:1708.04896.