SPARSITY GUIDED CO-TEACHING NEURAL NETWORKS FOR DYNAMICS IDENTIFICATION AND PREDICTION

Yanzhu Liu[1], Adams Wai-Kin Kong[2]

[1]Institute for Infocomm Research (I^{2}R) & Centre for Frontier AI Research (CFAR), A*STAR, Singapore
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore

ABSTRACT

Although deep learning methods have achieved promising performance on forecasting the future states of dynamical systems, most of them perform as a black box. This paper aims to discover the underlying equations of dynamical system from noisy observations. A dual Recurrent Neural Networks (RNNs) equipped with a sparse identifier is proposed for this problem. The identifier is optimized to determine the fewest terms in possible differential equations representing the dynamics, and the identification loss is used to co-teach the dual RNNs predicting the future dynamics. Experimental results on two benchmarking dynamical systems demonstrate the proposed method's capabilities on discovering the governing equations and predicting for future states.

Index Terms— Dynamic identification, sparse regression

1. INTRODUCTION

Estimating future states of a dynamical process from historical measurement data is important to a large number of applications in science and engineering, such as optimizing energy consumption, monitoring spread of disease, controlling traffic flow, simulating climate changes, predicting financial markets, and so on ([1] [2] [3]). Deep neural networks (DNNs) with powerful representation capability has achieved remarkable success in forecasting tasks. However, most deep learning models highly rely on the training data and lack interpretability for what they have learned. However, dynamics identification is a task aiming to automatically discover the explicit equations (i.e., ordinary differential equations (ODEs)) that describe the dynamical phenomena from the measurement data. Therefore, an intuition (**Motivation 1**) is that if DNNs and a dynamics identification module perform parallelly on the same samples and learn to predict for the same future states (see Figure 1(a)), when they make the same decisions, the equations outputted by the dynamics identification module can be viewed as an explanation of what the DNNs have learned and the DNNs helped identification module to boost the performance based on their excellent learning capabilities.

For many dynamical systems, although the observed states have complex, noisy and chaotic behaviours, there are only few dominant terms in the ODEs that govern the dynamics. Therefore, the assumption that the underlying equations are sparse combination of simple functions is widely used in the literature [4] [5] [6]. One important contribution to dynamics identification based on the sparsity assumption is SINDy (Sparse Identification of Nonlinear Dynamics) method [4], which searched for sparse coefficients (i.e., a set of coefficients, most of which are zeros) to combine a set of simple candidate functions as the time derivatives of the states. However, the SINDy method relied on accurate time derivatives of the measurement data, as they are the target values of the sparse regression.

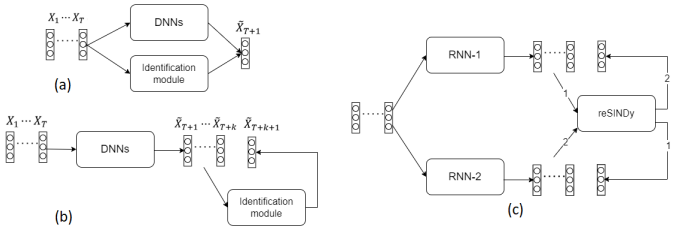


Fig. 1: Illustrated identification and prediction designs for 3-dimensional example dynamics (i.e., $X_t \in \mathbb{R}^3$). (a) Naive parallel structure (b) Revised structure (c) Co-teaching structure

The derivatives are unknown in most real-world cases and not easy to be computed precisely if the measurement data are noisy.

To make use of SINDy as the dynamics identification module in Figure 1(a), it requires to improve the robustness of SINDy for noisy observations. Otherwise, it will mislead the learning procedure when it learns together with DNNs. A simple adaptation, as shown in Figure 1(b), is to move the identification module to later time steps. Because the observed states $X_1 \cdots X_T$ on the time step $1, \cdots, T$ and the estimated states $\tilde{X}_{T+1} \cdots \tilde{X}_{T+k+1}$ on the time step $T+1, \cdots, T+k+1$ are governed by same equations, the DNNs and the identification module are still guiding each other under Motivation 1. In another view of point, DNNs works as a noise filter for the identification module in the revised structure. However, because the identification module in Figure 1(b) is optimized based on the DNNs' predictions, a concern is that the sparse identification is inaccurate especially in the early training stage. Fortunately, the research of neural network memorization effects [7][8] [9] overcame this concern. They found that neural networks learn the main data pattern first, gradually adapt to hard instances, and eventually memorize these noisy samples if any. Applying to the case in this paper, if the sparse assumption holds for the dynamics, then the sparse patterns are the main and clean patterns. Therefore, **Motivation 2** of this paper is that if samples with more sparse pattern are chosen to feedback to the DNNs, the learning will be guided towards more sparse results, which is the true pattern of data.

Motivation 1 and **Motivation 2** are instantiated together in this paper by a co-teaching schema. Co-teaching networks [9] were proposed for robust learning from noisy labels. This paper extends it to learn sparse patterns. To be more specific, as shown in Figure 1(c), the proposed framework consisting of two Encoder-Decoder fashion RNNs (RNN-1, RNN-2) and a revised SINDy module (reSINDy). RNNs use standard SGD optimizers to learn their weights separately and the reSINDy uses LASSO [10] as the sparse regressor to solve out the optimal coefficients. In each training iteration, a same batch of samples is forwarded to both RNNs and the followed reSINDy.

The samples with less errors given by reSINDy learned from RNN-1 outputs will be used to update RNN-2, and those with less errors learned from RNN-2 outputs will be used to update RNN-1. When the framework is well-trained, RNNs output the forecasting future states and reSINDy produces the discovered equations.

2. RELATED WORK

Physics-aware deep learning For dynamics prediction task, many attempts have been made to embed related physical knowledge into deep learning models. Raissi et al. [11] proposed physics-informed neural networks (PINNs) to learn nonlinear mappings from inputs and outputs which are the observations from a given partial differential equation (PDE). Xu et al. [12] presented a physics-constrained neural network which is guaranteed to numerically fulfill the PDE constraints during training. However, these existing methods share a basic assumption that the physical equations or PDEs are given in certain formulations, and are not able to discover explicit underlying equations automatically.

Sparse regression and sparse networks Sparse regression aims to find sparse solutions for systems of linear equations, i.e., find β for $\min_{\beta\in\mathbb{R}^m} \|\beta\|_0$ subject to $\mathbf{y} = X\beta$, where $X \in \mathbb{R}^{d\times m}, y \in \mathbb{R}^d$. $\|\beta\|_0$ is defined as the number of non-zero entries in the regression coefficients β. In the content of this paper, the systems of linear equations are $\dot{X} = \mathbf{f}(X)\beta$, where $\dot{X}$ is the time derivates of X and $\mathbf{f}(\cdot)$ is a set of candidate mappings with $\mathbf{f}_i(X) : \mathbb{R}^{d\times m} \rightarrow \mathbb{R}^m, i = 1, \cdots, d$. Recent works attempted to introduce sparsity into DNNs to reduce the network sizes and improve their generation capabilities and training efficiencies. The three main categories of methods include introducing ℓ_1 regularization to encourage sparsity of network weights [13]; approximating ℓ_0 regularization on weights [14]; and enhancing sparsity by revised activation functions [15]. However, sparse networks encourage the weights of random neurons to be zero, while for the dynamics identification, the coefficients (if implemented as weights of neurons in a network layer) in different positions have specific meanings as associated to different candidate functions. This issue gives difficulties to directly apply sparse DNNs for dynamics identification.

Co-teaching neural networks Co-teaching [9] was proposed to combat noisy labels for classification problems, which was justified by the study [7] showing that DNNs would first memorize the training data of clean labels and then those of noisy labels. Co-teaching is a sample selection strategy to train two DNNs jointly. In each training iteration, each network selects samples with small losses from the mini-batch data and exchanges them with its peer network for updating the parameters. Yang et al. [16] extended it to an asymmetric sample selection criteria for the unsupervised learning scenario. Co-Regularization [17] improved it further by selecting instances based on a new joint loss and updating the parameters of both networks simultaneously in each iteration. Inspired by co-teaching, this paper explores whether an additional loss from a different view of the problem can contribute to sample selection of DNNs training.

3. THE PROPOSED METHOD

For a dynamical system, a training dataset $\mathcal{D} = \{X_i | i = 1, \cdots, n\}$ with n instances is given, where $X_i = (\mathbf{x}_{i,t_1}, \cdots, \mathbf{x}_{i,t_S}, \cdots, \mathbf{x}_{i,t_{S+H}})^T \in \mathbb{R}^{(S+H)\times d}$ is a $S+H$-length segment of observations on the d-dimensional state. S is the length of historical series, and H is the forecast horizon. For a simple presentation, when referring to any instance in the training set, we omit the subscript i, i.e., denote an arbitrary instance as $X = (\mathbf{x}_{t_1}, \cdots, \mathbf{x}_{t_{S+H}})^T$, where $\mathbf{x}_{t_k} \in \mathbb{R}^d$ is the observation of the d-dimensional measurement at the time t_k, $t_k = t_1 + (k-1)\times\delta, k = 1, \cdots, S+H$ and δ is the time interval. For each training instance X, the historical series $X_S = (\mathbf{x}_{t_1}, \cdots, \mathbf{x}_{t_S})^T$ is given as input and $X_H = (\mathbf{x}_{t_{S+1}}, \cdots, \mathbf{x}_{t_{S+H}})^T$ as the ground truths of future states for a certain forecast horizon H. X_S can be also represented as $X_S = (\mathbf{x}_1^{(S)}, \cdots, \mathbf{x}_d^{(S)})$, where $\mathbf{x}_j^{(S)} \in \mathbb{R}^S$ is the observation of the j-th dimension measurement within the period, and $j = 1, \cdots, d$. In other words, X_S is a $S \times d$ matrix whose rows (i.e., $\mathbf{x}_{t_k}$) are for time steps and columns (i.e., $\mathbf{x}_j^{(S)}$) for state dimensions. Similarly, $X_H = (\mathbf{x}_{t_{S+1}}, \cdots, \mathbf{x}_{t_{S+H}})^T = (\mathbf{x}_1^{(H)}, \cdots, \mathbf{x}_d^{(H)})$. The task of dynamics identification and prediction is to learn a model based on $\mathcal{D}$. Given a testing input $X_S^* = (\mathbf{x}_{t_1}^*, \mathbf{x}_{t_2}^*, \cdots, \mathbf{x}_{t_S}^*)^T$, the model is expected being able to predict its future states $X_H^* = (\mathbf{x}_{t_{S+1}}^*, \cdots, \mathbf{x}_{t_{S+H}}^*)^T$, and output the discovered differential equations $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$, where $\mathbf{x} \in \mathbb{R}^d$ is the state variable and $\dot{\mathbf{x}} \in \mathbb{R}^d$ is its time derivative. Denote the equation on the j-th dimension as $\dot{\mathbf{x}}_j = \mathbf{f}_j(\mathbf{x}, t)$ where $\mathbf{f}_j : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}$ and $j = 1, \cdots, d$.

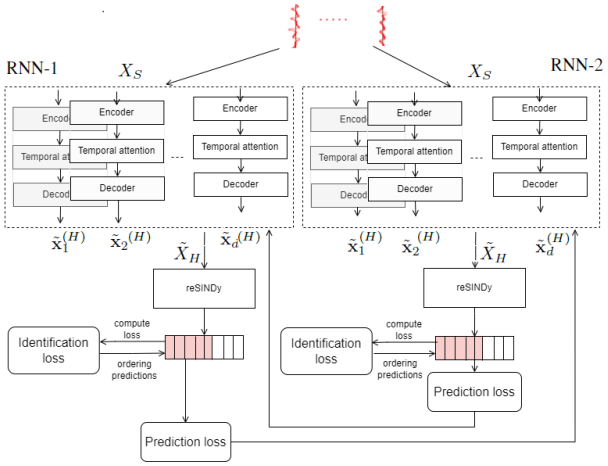


Fig. 2: The proposed architecture.

Figure 2 shows the proposed framework. It consists of a dual networks: RNN-1 and RNN-2. Each of them includes d recurrent predictors for d dimensions of the state. Encoder, temporal attention and decoder layers are connected following the structure of DARNN[18] to constitute each one of recurrent predictors. DARNN was originally proposed for time series forecasting which takes both exogenous features and 1d historical series as inputs and estimate the future values of the 1d series. In this study, we input $\mathbf{x}_{t_k}$ as the exogenous features at time t_k and $\mathbf{x}_j[1, \cdots, t_k]$ as the history of dimension j of the state. Let $\phi_j(X_S, \theta_j)$ be the mapping function of DARNN for dimension j specified by network parameters θ_j, which estimates H-horizon future steps of dimension j: $\tilde{\mathbf{x}}_j^{(H)} = \phi_i(X_S, \theta_i)$. Therefore, the estimation for the future states is concatenated by the d predictions, i.e., $\tilde{X}_H = (\tilde{\mathbf{x}}_1^{(H)}, \cdots, \tilde{\mathbf{x}}_d^{(H)})$. Both RNN-1 and RNN-2 have the same neural structure and receive the same mini-batch of input series in each training iteration. They are supervised by **Prediction Loss** defined in Eq. 1:

$$L_1(\mathcal{D}, \theta) = \frac{1}{|\mathcal{D}|} \sum_{X=(X_S, X_H)^T} \frac{1}{d} \sum_{j=1}^{d} \|\phi_j(X_S, \theta_j) - \mathbf{x}_j^{(H)}\|_2^2 \quad (1)$$

where $\theta = \{\theta_j | j = 1, \cdots, d\}$. L_1 is the mean square loss on the

estimation of future states.

In each training iteration, a mini-batch of instances is inputted into RNN-1 and RNN-2, but only those instances with small identification loss will be used to compute L_1 and back-propagated to the networks. The identification module reSINDy aims to discover differential equations $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ from the predictions of RNNs. Following SINDy method [4], we search $\mathbf{f}_j$ as $\dot{\mathbf{x}}_j = \mathbf{f}_j(\mathbf{x}, t)$ by sparse regression for each dimension j separately. A predefined function library $\{g_1(\mathbf{x}, t), g_2(\mathbf{x}, t), \cdots, g_m(\mathbf{x}, t)\}$ includes m candidate functions (i.e. $g_j : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$) in simple formulations, such as polynomial functions, triangular functions, and so on. Define a vector $\mathbf{g}_{\mathbf{x},t} = (g_1(\mathbf{x}, t), \cdots, g_m(\mathbf{x}, t))^T \in \mathbb{R}^m$ which concatenates function values of all candidate functions for an input $(\mathbf{x}, t)$. Furthermore, construct the matrix $G_{\tilde{X}_H} = (\mathbf{g}_{\tilde{\mathbf{x}}_{t_{S+1}} t_{S+1}}, \mathbf{g}_{\tilde{\mathbf{x}}_{t_{S+2}} t_{S+2}}, \cdots, \mathbf{g}_{\tilde{\mathbf{x}}_{t_{S+H}} t_{S+H}})^T \in \mathbb{R}^{H \times m}$ by substituting $\tilde{\mathbf{x}}_{t_{S+1}}, \cdots, \tilde{\mathbf{x}}_{t_{S+H}}$ into all candidate functions. Figure 4 gives an example of 3-dimensional dynamics, where the candidate function library includes polynomial functions $\{x_1, x_2, x_3, x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2\}$. The goal of sparse identification is to find coefficients $\beta_j \in \mathbb{R}^m$ that $\dot{\mathbf{x}}_j = \beta_{j1}x_1 + \beta_{j2}x_2 + \cdots + \beta_{jm}x_3^2$ fits dimension j data and most β_{jk} are zeros where $k = 1, \cdots, m$. The objective function is defined for dimension j by Eq. 2:

$$\min_{\beta_j \in \mathbb{R}^m} \quad \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \|\dot{\mathbf{x}}_j^{(H)} - G_{\tilde{X}_H}\beta_j\|_2^2$$
$$s.t. \quad \beta_{jk} \in \{0\} \cup [\lambda, +\infty) \qquad k = 1, \cdots, m \quad \lambda > 0 \quad (2)$$

where $\dot{\mathbf{x}}_j^{(H)} \in \mathbb{R}^H$ is the time derivatives of X_H on dimension j, $\beta_j \in \mathbb{R}^m$ is the coefficients and λ is a constant hyper-parameter. While optimizing Eq. 2 , the sparsity of coefficients β_j is controlled with λ by setting the feasible domains of β_{jk}: if β_{jk} is smaller than λ, then it must be zero. An iterative algorithm 1 based on least square optimizer is used to achieve an approximate solution of Eq. 2.

As shown in Figure 2, **Identification Loss** is used to select samples to update the networks parameters. Eq. 3 gives its definition, which is the mean square loss on all dimensions of one instance based on the solved β_js.

$$L_2(\mathcal{D}, \beta) = \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \frac{1}{d} \sum_{j=1}^{d} \|\dot{\mathbf{x}}_j^{(H)} - G_{\tilde{X}_H}\beta_j\|_2^2 \quad (3)$$

For each iteration in the proposed co-teaching flow, a mini-batch of input series is forwarded to both RNN-1 and RNN-2 and output the network predictions $G_{\tilde{X}_H}^{(N1)}$ and $G_{\tilde{X}_H}^{(N2)}$. With respect to these two outputs, the identification module reSINDy optimizes β_{N1} and β_{N2} by solving Eq. 2. Then the instances are ordered by the identification loss L_2. Small-loss instances are selected out (as highlighted in colour in Figure 2) and are exchanged to update its peer RNN. The dual RNNs are trained in a standard SGD optimizer for loss $L1$. The final prediction is conducted by the RNN with small training loss and the coefficients deciding the ODEs are produced by the reSINDy with small training loss.

4. EXPERIMENTS

To evaluate the proposed method SpaCo-RNN, numerical experiments are performed and forecasting performance is evaluated by Root Mean Square Error $\sqrt{\frac{1}{H|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{i=1}^{H}\sum_{j=1}^{d}(\tilde{\mathbf{x}}_{ij}^{(H)} - \mathbf{x}_{ij}^{(H)})^2}$ where $\mathcal{D}$ is the testing set, H is the forecasting horizon, and d is the number of dimensions of the dynamics.

Algorithm 1 Pseudo code of solving Eq. 2

Input: $\{\dot{\mathbf{x}}_j^{(H)} \in \mathbb{R}^H | X \in \mathcal{D}\}$, $\{G_{\tilde{X}_H} \in \mathbb{R}^{H \times m} | X \in \mathcal{D}\}$, λ, M.
Output: β_j.

```
Concatenate all ẋ_j^(H) together as ẋ_j. (ẋ_j ∈ R^{H|D|})
Concatenate all G_{x̃_H} together as G. (G ∈ R^{H|D|×m})
Set big_ids = {1, ··· , m}
for iteration = 1 to M do
    β_j[big_ids] = lstsq(ẋ_j, G[big_ids])
    for k = 1 to m do
        if β_jk < λ then
            Set β_jk = 0, remove k from big_ids
    if big_ids is not changed then
        break
end for
Output β_j.
```

4.1. Benchmarks

Lorenz-63 dynamics is a 3-dimensional chaotic system with butterfly attractor. It is defined by the following ODEs: $\dot{x}_1 = \sigma \times (x_2 - x_1), \dot{x}_2 = x_1 \times (\rho - x_3) - x_2, \dot{x}_3 = x_1 \times x_2 - \beta \times x_3$, where x_1, x_2, x_3 are states changing upon the time and $\dot{x}_1, \dot{x}_2, \dot{x}_3$ are their time derivatives respectively. Following [19], we use 250 different random initial states to generate 250 series of 180-length with the time interval $\delta = 0.01$, and randomly split them as 200:50 training/testing partition. Then we construct two corrupted datasets by adding Gaussian noise with 15% signal-noise ratio (i.e., $r = std_\epsilon / std_{\mathbf{x}}$) and 33% signal-noise ratio into random 50% of training series. Two forecasting horizons are tested - 2 steps ahead and 20 time steps ahead. For the 50% noisy training series, both input segments (X_S) and target segments (X_H) are noisy. Int testing phase, we do not assume any ideal noise-free time points can be observed. Therefore, all input segments of testing series are noisy, and clean ground truth (X_H^*) are only used to compute the evaluate metrics.

Lorenz-96 dynamics [20] is a 40-dimensional chaotic system. It is defined by $\dot{x}_i = (x_{i+1} - x_{i-2}) \times x_{i-1} - x_i + F$ with $i = 1, \cdots, 40$ and cyclic boundaries $x_{-1} = x_{39}, x_0 = x_{40}, x_{41} = x_1$. The forcing parameter F is chosen as 8 in the experiments as previous works [21][19][22]. Lorenz-96 is used to evaluate the performance of methods on high dimension. We use 250 different random initial states to generate 250 series of 800-length with time step $\delta = 0.01$, and randomly split them as 200:50 training/testing partition. The horizon and noise ratio settings are the same as those for Lorenz-63.

4.2. Baseline methods & implementation details

Firstly, three state-of-the-art DNNs-based forecasting methods specially designed for noisy time series are compared: classic sequence-to-sequence RNN network ("rnn-seq2seq"), GRU-based ODE network ("ode-gru") [23] and Latent ODE [24] network ("Latent ODE"). These methods only focus on future prediction but not able to identify the explicit underlying equations. The implementations are using code [25]. For the experiments on Lorenz-63 dataset, 2 generative layers and receive layers are used. The mini-batch size is 200, and the number of training epochs is 1000 for rnn-seq2seq and LatentODE, 300 for ode-gru, and the learning rate is 0.01. For Lorenz-96 dataset, 3 generative layers and receive layers are used, and the number of training epochs is 300.

Secondly, original SINDy is compared. All parameters are set as default in [26]. Polynomial functions to 2nd order are used as candi-

Table 1: RMSE results on Lorenz-63 dynamics

	$h = 2$		$h = 20$	
	r=15%	r=33.3%	r=15%	r=33.3%
rnn-seq2seq	0.6972	1.3102	0.7637	1.2342
ode-gru	0.7708	1.1658	0.8634	1.4388
LatentODE	0.7394	1.3173	0.9874	1.5143
SINDy	1.0873	2.5002	1.5462	3.4590
att-RNN	0.4315	0.8413	0.4943	1.0282
dual-RNN	0.4540	0.8253	**0.4892**	**0.8589**
SpaCo-RNN	**0.4195**	**0.7792**	0.5161	0.9090

Table 2: RMSE results on Lorenz-96 dynamics

	$h = 2$		$h = 20$	
	r=15%	r=33.3%	r=15%	r=33.3%
rnn-seq2seq	2.0395	2.1716	2.9088	3.0212
ode-gru	1.4765	1.9738	3.2870	3.3001
LatentODE	1.8661	2.1759	3.6118	3.3858
SINDy	0.5467	1.2154	**0.7822**	1.5207
att-RNN	0.4966	0.8077	1.1784	1.2330
dual-RNN	0.5079	0.8319	1.5066	1.6442
SpaCo-RNN	**0.4906**	**0.7955**	1.1467	**1.1072**

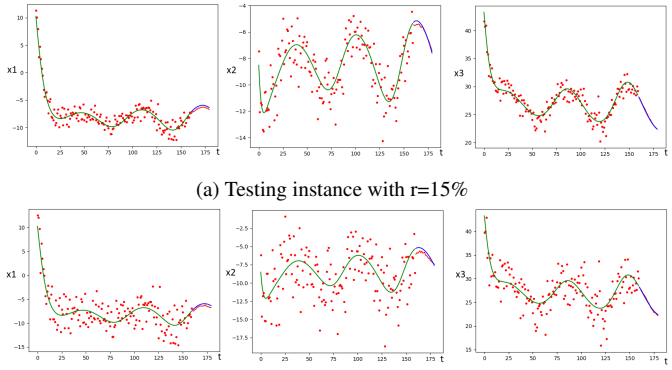
date function library, e.g., for 3d dynamics $(x_1, x_2, x_3)^T$, the candidate functions include $\{x_1, x_2, x_3, x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2\}$. SpaCo-RNN uses the same candidate library as SINDy for both Lorenz-63 and Lorenz-96 datasets.

Lastly, single set of RNNs in Figure 2 ("att-RNN") and the dual set of RNNs using typical co-teaching strategy ("dual-RNN") are compared to evaluate the benefits of different components of the proposed method. Both att-RNN and dual-RNN are not able to output the ODEs for the dynamical systems. The RNNs in att-RNN, dual-RNN and SpaCo-RNN are implemented as same architecture by using DARNN code [27], where the number of units in both encoder hidden layer and decoder hidden layer are 24. As DARNN is window based approach, the input instances are segmented further into 25-length sub series as inputs for all these three models. For both Lorenz-63 and Lorenz-96 datasets, the mini-batch size is 2048, learning rate is 0.01 for all att-RNN, dual-RNN and SpaCo-RNN. The number of training epochs is 100 and 200 for forecasting of horizon 2 and horizon 20 respectively.

4.3. Performance on Lorenz-63

Table 1 summarizes the results of different noise ratios and horizons on Lorenz-63 dataset. For horizon 2 forecasting, the proposed method SpaCo-RNN outperforms all baselines. For horizon 20 forecasting, dual-RNN leads the performance. Because the co-teaching strategy based on an observation that in the early stage of deep network training, networks learn the primary features even from the noisy label. When we use sparse regression to solve the optimal coefficients based on the early stage estimation, noise affects more on regressor than pure networks as in dual-RNN. However, it should be emphasized that att-RNN and dual-RNN are not able to identify the underlying equations. The goal of this study is using deep network to discover the ODEs while not sacrifice forecasting accuarcy much.

Figure 3 shows examples of predictions, where red dots are testing inputs, blue lines are ground truths and red lines are predictions of the proposed method. The identified coefficients for Lorenz-63 datasets are listed in Figure 4. Comparing Figure 4(c) and (d), with the increasing noise level, the coefficients identified by SpaCo-RNN become inaccurate, e.g., the coefficient of x_2 in β_2 is identified as



(a) Testing instance with r=15%

(b) Testing instance with r=33.3%

Fig. 3: Examples of prediction results

$$\dot{x}_1 = -10x_1 + 10x_2$$
$$\dot{x}_2 = 28x_1 - x_2 - x_1x_3$$
$$\dot{x}_3 = x_1x_2 - \frac{8}{3}x_3$$

(a) Ground truth ODEs

(b) Ground truth coefficients

	x_1	x_2	x_3	x_1^2	x_1x_2	x_1x_3	x_2^2	x_2x_3	x_3^2
β_1	-10.21	9.99	0	0	0	0	0	0	0
β_2	30.01	-1.23	0	0	0	-1.04	0	0	0
β_3	0	0	-2.56	0	0.98	0	0	0	0

(c) Identified coefficients by SpaCo-RNN on data with r=15%

	x_1	x_2	x_3	x_1^2	x_1x_2	x_1x_3	x_2^2	x_2x_3	x_3^2
β_1	-10.17	10.28	0	0	0	0	0	0	0
β_2	25.37	0	0	0	0	-0.94	0	0	0
β_3	0	0	-2.55	0	0.91	0	0	0	0

(d) Identified coefficients by SpaCo-RNN on data with r=33.3%

	x_1	x_2	x_3	x_1^2	x_1x_2	x_1x_3	x_2^2	x_2x_3	x_3^2
β_1	-8.98	9.18	0	0	0	0	0	0	0
β_2	23.45	0.72	0	0	0	-0.90	0	0	0
β_3	0	0	-2.55	0	0.98	0	0	0	0

(e) Identified coefficients by SINDy on data with r=15%

	x_1	x_2	x_3	x_1^2	x_1x_2	x_1x_3	x_2^2	x_2x_3	x_3^2
β_1	-3.55	1.61	0	0	0	-0.30	0	0.15	0
β_2	19.64	1.42	0	0	0	-0.79	0	0	0
β_3	0.12	-0.11	-2.34	0	0.89	0	0	0	0

(f) Identified coefficients by SINDy on data with r=33.3%

Fig. 4: Identification results on Lorenz-63 system.

zero which is -1 in the true dynamics. While for SINDy method, the identified equations on data with r=33.3% (i.e., Figure 4 (f)) are far away from reasonable estimation of true equations.

4.4. Performance on Lorenz-96

The results of Lorenz-96 are summarized in Table 2. The three deep learning methods rnn-seq2-seq, ode-gru and latentODE perform worse than SINDy and the proposed method with different components. The traditional co-teaching strategy dual-RNN decreases the accuracy of its peer model att-RNN without co-teaching. The SINDy method identifies equations first and then forecasts future values based on the equations. Therefore, predicting for different horizons of same data, the results of SINDy are not changed as much as the proposed method. Because in SpaCo-RNN, deep networks take in charge of prediction, and identification module focuses on outputting the equations. With the increasing length of horizons, the prediction performance drops due to the difficulty of decoding. Taking $H = 20$ as an example, the input series of encoder is 25-length rolled by window, and the output target is 20-length series, which is challenging for the DARNN network.

5. CONCLUSIONS

This paper integrates a sparse regression model with a dual RNNs under co-teaching framework for dynamic identification and prediction problems. The RNNs are trained end-to-end guided by a sparse loss in the stochastic mini-batch manner, and the underlying differential equations and future states of dynamic systems are learned by the identification module and prediction module simultaneously.

6. REFERENCES

[1] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *AAAI*, 2019, vol. 33, pp. 3656–3663.

[2] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl, "Neural message passing for quantum chemistry," 2017, vol. 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272.

[3] Anand Pratap Singh, Shivaji Medida, and Karthik Duraisamy, "Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils," *AIAA journal*, vol. 55, no. 7, pp. 2215–2227, 2017.

[4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[5] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22445–22451, 2019.

[6] Steven L Brunton and J Nathan Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*, 2019.

[7] Devansh Arpit, Stanisaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al., "A closer look at memorization in deep networks," in *ICML*, 2017, pp. 233–242.

[8] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[9] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *NeurIPS*, 2018, vol. 31.

[10] Robert Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[11] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[12] Kailai Xu and Eric Darve, "Physics constrained learning for data-driven inverse modeling from sparse observations," *arXiv preprint arXiv:2002.10521*, 2020.

[13] Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani, "Lassonet: A neural network with feature sparsity," *Journal of Machine Learning Research*, vol. 22, no. 127, pp. 1–29, 2021.

[14] Christos Louizos, Max Welling, and Diederik P Kingma, "Learning sparse neural networks through l_0 regularization," in *ICLR*, 2018.

[15] Andre Martins and Ramon Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *ICML*, 2016, pp. 1614–1623.

[16] Fengxiang Yang, Ke Li, Zhun Zhong, Zhiming Luo, Xing Sun, Hao Cheng, Xiaowei Guo, Feiyue Huang, Rongrong Ji, and Shaozi Li, "Asymmetric co-teaching for unsupervised cross-domain person re-identification," in *AAAI*, 2020, vol. 34, pp. 12597–12604.

[17] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An, "Combating noisy labels by agreement: A joint training method with co-regularization," in *CVPR*, 2020, pp. 13726–13735.

[18] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison W Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *IJCAI*, 2017, pp. 2627–2633.

[19] Duong Nguyen, Said Ouala, Lucas Drumetz, and Ronan Fablet, "Variational deep learning for the identification and reconstruction of chaotic and stochastic dynamical systems from noisy and partial observations," *arXiv preprint arXiv:2009.02296*, 2020.

[20] Edward N Lorenz, "Predictability: A problem partly solved," in *Proc. Seminar on predictability*, 1996, vol. 1.

[21] Redouane Lguensat, Pierre Tandeo, Pierre Ailliot, Manuel Pulido, and Ronan Fablet, "The analog data assimilation," *Monthly Weather Review*, vol. 145, no. 10, pp. 4093–4107, 2017.

[22] Ronan Fablet, Said Ouala, and Cédric Herzet, "Bilinear residual neural network for the identification and forecasting of geophysical dynamics," in *26th european signal processing conference*, 2018, pp. 1477–1481.

[23] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud, "Neural ordinary differential equations," in *NeurIPS*, 2018.

[24] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud, "Latent ordinary differential equations for irregularly-sampled time series," in *NeurIPS*, 2019.

[25] "latentode," https://github.com/YuliaRubanova/latent_ode.

[26] "pysindy," https://github.com/dynamicslab/pysindy.

[27] "Daoden," https://github.com/CIA-Oceanix/DAODEN.