Adversarial Joint-Learning Recurrent Neural Network for Incomplete Time Series Classification

Qianli Ma, *Member, IEEE,* Sen Li, and Garrison W. Cottrell

Abstract—Incomplete time series classification (ITSC) is an important issue in time series analysis since temporal data often has missing values in practical applications. However, integrating imputation (replacing missing data) and classification within a model often rapidly amplifies the error from imputed values. Reducing this error propagation from imputation to classification remains a challenge. To this end, we propose an Adversarial Joint-learning Recurrent Neural Network (AJ-RNN) for ITSC, an end-to-end model trained in an adversarial and joint learning manner. We train the system to categorize the time series as well as impute missing values. To alleviate the error introduced by each imputation value, we use an adversarial network to encourage the network to impute realistic missing values by distinguishing real and imputed values. Hence, AJ-RNN can directly perform classification with missing values and greatly reduce the error propagation from imputation to classification, boosting the accuracy. Extensive experiments on 68 synthetic datasets and 4 real-world datasets from the expanded UCR time series archive demonstrate that AJ-RNN achieves state-of-the-art performance. Furthermore, we show that our model can effectively alleviate the accumulating error problem through qualitative and quantitative analysis based on the trajectory of the dynamical system learned by the RNN. We also provide an analysis of the model behavior to verify the effectiveness of our approach.

Index Terms—Incomplete time series classification, recurrent neural networks, adversarial learning, exploding error

1 Introduction

Time series data are ubiquitous. They arise in a wide range of applications, and in many different domains, such as health care [1], activity recognition [2], financial markets [3] and urban traffic control [4]. In addition to these domains, time series data is also used to study human behavior. For example, Marjaninejad et al. [5] extracted amplitude modulation features from the electrocardiogram (ECoG) recordings in order to explain the finger movements. In the past two decades, although many time series classification (TSC) methods have been proposed, most of them assume that the observed data are complete. However, due to anomalies, communication errors, or malfunctioning sensors, real-world time series data are usually corrupted by missing values with various frequencies. An actual case of time series classification with missing values is shown in Fig 1, where inductive loop sensors deployed by CalTrans record traffic flow data on the 101 North freeway in Los Angeles.

Incomplete data make any kind of inference more difficult [6], [7] and degrade the data analyses results [8]. Compared to traditional time series classification based on complete data, incomplete time series classification (ITSC) remains a more practical and challenging problem, since it aims to capture the patterns and dynamics of a time series with incomplete information. Therefore, avoiding the negative impact of missing values in an ITSC model is an important issue in real-world time series data mining.

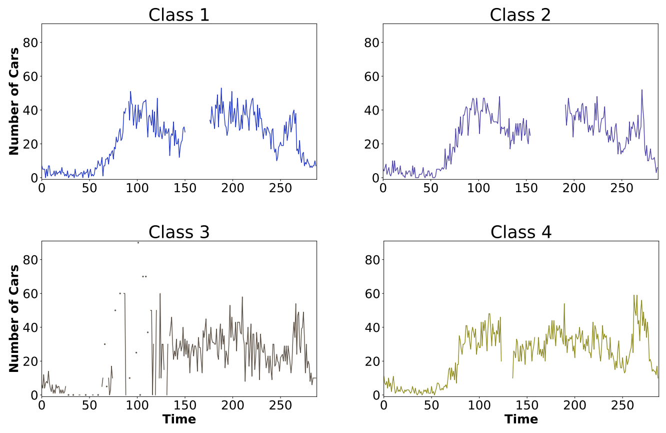


Fig. 1. The `DodgerLoopDay` dataset [9]. The graph shows the number of cars every five minutes (288 points per day). The classes of the `DodgerLoopDay` dataset are days of the week (we only show four classes with missing values).

A natural strategy for dealing with ITSC is to first impute missing values and then perform classification. However, this two-stage methodology has two significant drawbacks: 1) Imputation is treated as an independent step of data preprocessing and has no interaction with the training of the classifier, which can lead to suboptimal results [10], [11]. 2) The classification performance largely depends on the

Qianli Ma is with the School of Computer Science and Engineering, and the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou, 510006, China (e-mail: qianlima@scut.edu.cn, corresponding author).
Sen Li is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China (e-mail: awslee@foxmail.com).
Garrison W. Cottrell is with the Department of Computer Science and Engineering, University of California, San Diego, CA 92093, USA (e-mail: gary@ucsd.edu).

chosen imputation method because biased imputed values can have a significant impact on classification. Hence it is important to identify the cause of the missing data and apply approaches appropriate to that setting.

The causes of missing data can be divided into three types: 1) missing completely at random (MCAR), 2) missing at random (MAR), and 3) missing not at random (MNAR) [12]. MCAR refers to missingness that is independent of all observed and unobserved values. One such example is sensor failure, as in the `DodgerLoopDay` dataset. MAR refers to missingness that is independent of unobserved values but depends on the observed values: patients with very good vital signs may not need to undergo certain lab tests. MNAR refers to missingness that is informative of the category, where what data is missing correlates with the category. The latter two are considered to be informative missingness in [11]. In this paper, we assume the MCAR regime.

With the booming interest in deep learning, various types of deep neural network models have been used in the field of time series analysis and have achieved state-of-the-art performance in many practical applications [13], [14]. Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) networks [15] and Gated Recurrent Unit (GRU) networks [16], have shown excellent ability to model long-term temporal dependencies within sequence data. However, incomplete data presents problems for the learning mechanism of RNNs since they are designed to be used with complete data. Common methods for dealing with missing values, such as imputing the mean or the previous value can introduce errors that can accumulate over time, distorting the underlying dynamical system and thus misleading the RNNs' inference in classification.

In order to avoid suboptimal solutions obtained by using this two-step process [10], [11], one could train the RNN to impute missing values and classify the time series simultaneously. Under the MCAR setting, Shen et al. [17] adapt this joint learning strategy with some success. Under the informative missingness setting in the medical domain, Che et al. [11] simultaneously learn decays for mixing the last seen and mean value to impute missing values in a smooth fashion while conducting classification. Cao et al. [18] combine these two ideas and consider both learning decays and regression for imputation, and attempt to alleviate the error accumulation problem [19] with a bidirectional network. However, they do not provide an analysis to verify that their approach actually does alleviate the problem. Besides, for the recommendation, Li et al. consider dynamically clustering the users and propose the context-aware bandit (CAB) [20] and the graph cluster of bandits (GCLUB) [21] algorithms to reduce regret rate (error). However, in our case, clustering the incomplete data may also introduce errors.

Recently, Generative Adversarial Networks (GANs) have become popular in deep learning due to their potential for generating data according to the distribution of a real dataset [22], [23]. The adversarial approach leads to better generation results than previous approaches [24], [25]. GANs have also been used to impute missing data in time series prediction with some success [26], [27], [28]. However, the question of how to employ adversarial learning in the domain of incomplete time series *classification* (ITSC) has not been previously explored.

In this paper we integrate adversarial training and joint (imputation and classification) learning in recurrent neural networks (RNNs), and call our system Adversarial Joint-learning RNN (AJ-RNN). Our results demonstrate that this is an effective method to address the practical issue of incomplete time series classification in an end-to-end framework.

AJ-RNN is trained to approximate the value of the next input variable when it is revealed, and to fill it in with its prediction if it is missing. At the same time, AJ-RNN also learns to classify. Hence AJ-RNN can directly perform classification with missing values.

The main innovation is to use adversarial training to reduce the error propagation from imputation to classification through sharpening the model's predictions so that the distribution of predicted values is closer to the distribution of real values. As we will see, this approach reduces imputation bias, which can prevent small errors from snowballing into large ones, which we call the exploding bias problem [19]. Using these two strategies, AJ-RNN is an effective dynamical system for ITSC.

The main difference between our model and existing approaches [11], [17], [18], [26], [27], [28], [29] is the smooth integration of the adversarial and joint learning strategies into one end-to-end mechanism. Previous work has applied the adversarial and joint learning strategies separately, by using the adversarial strategy for imputation, and joint strategy for prediction/classification, for example. Here we co-adapt the adversarial and joint learning strategies. To the best of our knowledge, this is the first model to do so, and the first to solve the exploding bias problem in ITSC.

The main contributions of this paper can be summarized as follows:

- We propose a novel end-to-end learning framework named Adversarial Joint-learning RNN (AJ-RNN) for ITSC, which can directly categorize the time series in the presence of the missing values.
- We apply adversarial training to alleviate the exploding bias problem and thus reduces the error propagation from imputation to classification, improving the classification accuracy.
- Our model is evaluated on 68 synthetic `UCR` incomplete time series datasets with different missing rates and on 4 real-world incomplete datasets. We show that AJ-RNN is state of the art on these problems.
- Finally, we present an error accumulation analysis into the exploding error problem, and provide an explanation based on the trajectory of the RNN (the learned dynamical system) for how our model solves this problem. Moreover, we conduct an experiment to explore the impact of normalization on classification performance in the case of complete and incomplete data.

The remainder of the paper is organized as follows. Section 2 discusses related work on addressing missing values in time series classification. Section 3 gives the network structure of Gated Recurrent Units. Section 4 presents our method more formally. Section 5 describes the details of the experimental settings, and Section 6 reports results and analysis. We conclude in Section 7.

2 Related Work

Incomplete time series classification (ITSC) methods can be roughly divided into two categories: pre-imputation methods and joint learning methods.

2.1 Pre-Imputation Methods

The presence of missing values in time series degrades classification performance, which has lead to a great deal of work on handling missing values. Common interpolation methods, such as splines and moving averages, estimate the missing value from nearby revealed values, which ignores any long-term dependencies in the time series. The Expectation Maximization (EM) algorithm [30] is widely applied for filling in missing values. Li et al. [31] propose DynaMMo, which learns a set of latent variables using the EM algorithm, to model the underlying dynamical system and hidden patterns of the observation sequences. Their model, however, is cubic in the dimensionality of the time series, which is a major limitation in the era of big data.

Matrix Factorization (MF) based imputation is an active research area because it is highly scalable to big data. It typically factorizes the incomplete matrix into low-rank matrices U and V and recovers the missing values from the obtained matrix. A representative work is the temporal regularized matrix factorization (TRMF) [32] algorithm, which integrates the temporal dependencies into the objective function of matrix factorization by adding a regularizer between observed and missing values. However, MF-based methods typically assume the time series is derived from linear dynamics, while non-linear dynamics is more common in time series [33], [34], [35].

Generative Adversarial Networks (GANs) have achieved great success in image generation [22], [23], [36]. Recently, GANs have been applied to sequence generation: SeqGAN [37] and MaskGAN [38]. However, these systems rely on complete data during training, and cannot be applied to the case of missing data. GANs also have been applied to impute missing values, such as GAIN [26] and MisGAN [27]. Most recently, Luo et al. [28] proposed GAN-based imputation for irregular time series (time series with different sampling rates).

Although these methods achieve good imputation performance, they are still just used as a data pre-processing step for the classification task, and hence are not optimized in concert with training the classifier, which can lead to suboptimal results [10], [11].

2.2 Joint Learning Methods

Instead of treating the imputation as a step in data preprocessing, some recent research has applied a joint learning strategy, where imputation and prediction/classification are simultaneously learned in an RNN. This approach optimizes the imputation and prediction/classification together, providing an end-to-end framework for modeling incomplete time series classification.

Under the MCAR setting, Ma and colleagues [17], [39] introduced the residual sum unit (RSU) into LSTMs. The RSU acts as an exponentially-decaying representation of past hidden unit activity, and it is used to predict missing values by training a network using the RSU to predict observed values. Under the informative missingness regime in the medical domain, Che et al. [11] propose GRU-D, which simultaneously mixes the last seen and global mean value with a decay factor to replace missing values and classify the time series. The imputed values are estimated by the network and used as input, which may result in the exploding bias problem [19]. The exploding bias problem initially exists in sequence prediction, and Bengio et al. [19] dealt with it by scheduled sampling. Since they assume the data is complete and focus on the prediction task, they can sample from real and predicted values with a certain probability and then use it as the next input of the model. However, in our missing case, there is no ground truth for the missing values, which is more challenging. Cao et al. [18] consider both temporal decay [11] and regression for imputation [17] and further adopt a bidirectional RNN to alleviate the exploding bias problem. Their intuition is to use a bidirectional network to shorten the path length of gradient propagation to the missing values, while our approach directly provides the gradient to the missing values.

Our approach does not use a bi-directional RNN (although the model does not preclude that) but a unidirectional RNN to focus on the exploding bias problem that exists in the RNN-based joint learning framework. Our contribution is to use a discriminator to deal with the exploding bias problem, reducing the error propagation from imputation to classification through sharpening the imputed values to make their distribution close to the real ones. Also, because of the joint training, including classification and imputation, the imputed values are influenced to be discriminative between different categories, aiding in classification performance.

3 Preliminaries

We here briefly introduce the used gated recurrent units (GRUs) [16] for clarifying the proposed method and making the paper more compact.

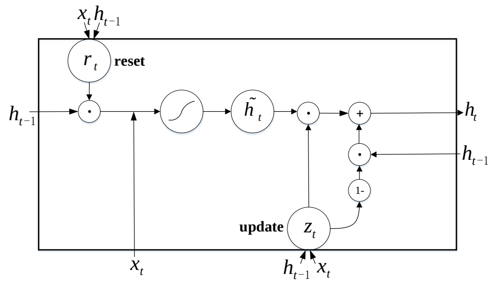


Fig. 2. The network structure of Gated Recurrent Units [16].

Cho et al. [16] proposed a lightweight version of recurrent neural networks, named GRUs (shown in Fig. 2), which can model long-term dependence and reduce training time. Given a sequential T-step time series $\boldsymbol{X} = \{x_1, x_2, \ldots, x_T\}$, a GRU encodes it as a hidden representation $\boldsymbol{H} = \{h_1, h_2, \ldots, h_T\}$, where the inputs are $x_t \in \mathbb{R}^n$, and the hidden state $h_t \in \mathbb{R}^m$. The detail update formula is as follows:

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z\boldsymbol{x}_t + \boldsymbol{U}_z\boldsymbol{h}_{t-1}) \quad (1)$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r\boldsymbol{x}_t + \boldsymbol{U}_r\boldsymbol{h}_{t-1}) \quad (2)$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h\boldsymbol{x}_t + \boldsymbol{U}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})) \quad (3)$$

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t + (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} \quad (4)$$

where $\boldsymbol{W}_z, \boldsymbol{U}_z$ and $\boldsymbol{W}_r, \boldsymbol{U}_r$ denotes the parameters related to update gate $\boldsymbol{z}$ and reset gate $\boldsymbol{r}$, respectively. $\boldsymbol{W}_h, \boldsymbol{U}_h$ denotes the parameters of the internal state $\tilde{\boldsymbol{h}}_t$.

4 Proposed Methods

In this section, we present our model, the Adversarial Joint-learning Recurrent Neural Network (AJ-RNN), to address the issue of time series classification with missing values. The framework is shown in Figure 3. Here we describe how adversarial and joint learning strategies are integrated into an RNN.

4.1 Joint Learning Strategy

We denote a time series $\boldsymbol{X} = \{x_1, x_2, ..., x_T\}$ as a sequence vector of T observations, where each observation $x_t \in \mathbb{R}^d$. Suppose a time series $\boldsymbol{X}$ has missing values which can be indicated by a T-dimensional mask vector $\boldsymbol{M} = \{m_1, m_2, ..., m_T\}$, where $m_t \in \{0,1\} \in \mathbb{R}^d$, m_t is 1 if x_t is revealed, and 0 if x_t is missing. Since we focus on incomplete time series classification, there is a target label $y^{(i)}$ for i-th time series $\boldsymbol{X}^i$ given the time series data set $\mathbb{D}$, where $\mathbb{D} = \{(\boldsymbol{X^i}, \boldsymbol{M^i}, y^i)\}_{i=1}^N$.

The framework of our AJ-RNN is shown in Fig. 3. Note that the RNN layer can use any kind of RNN unit; in this paper we use Gated Recurrent Units (GRUs, [16]) due to their simplicity and ease of training. To avoid the traditional two-step approach, we regard imputation and classification as two tasks in multitask learning. Specifically, the AJ-RNN is trained to approximate the value of the next input variable, using it as a target when it is revealed, while if the next value is missing, it will be filled in with the current prediction. At the same time, the AJ-RNN is trained to perform the classification task.

We solve the missing value problem by two processes: approximation and imputation. As seen in Fig. 3, two kinds of links enable AJ-RNN to directly model time series in the presence of missing values: dashed blue links (for approximation) and solid blue links (for imputation). We train $\hat{x}_t$ to approximate the next value x_t when it is revealed using the last hidden states $\boldsymbol{h}_{t-1}$ as follows:

$$\hat{x}_t = \mathbf{W_{imp}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z \quad (5)$$

where $\mathbf{W_{imp}} \in \mathbb{R}^{n\times m}$ is a learned regression matrix and $\boldsymbol{b}_z$ is a bias term. Since $\hat{x}_t$ is trained to approximate the next value x_t, it can be used to impute it when missing. The input value u_t is computed as:

$$u_t = m_t \odot x_t + (1 - m_t) \odot \hat{x}_t \quad (6)$$

where m_t is the mask as defined above and $\odot$ is the element-wise product.

The RNN is trained using the completed input value u_t. Formally, the update equation of the RNN is:

$$\boldsymbol{h}_t = \mathcal{F}_{RNN}(\boldsymbol{h}_{t-1}, u_t; \mathbf{W}) \quad (7)$$

where $\boldsymbol{h}_t$ represents the hidden unit vector at time t, $\mathbf{W}$ encapsulates the input-to-hidden and hidden-to-hidden parameters, and $\mathcal{F}_{RNN}$ represents the update function of the particular RNN variant.

Finally, the last hidden state of the RNN, $\boldsymbol{h}_T$, is fed into the classifier to obtain the probability distribution over each category label using a softmax:

$$P(\hat{\boldsymbol{y}}_{\boldsymbol{j}}|\boldsymbol{h}_T) = \frac{\exp(\boldsymbol{W}_{\boldsymbol{j}}^{\intercal}\boldsymbol{h}_T)}{\sum_{l=1}^{K}\exp(\boldsymbol{W}_{\boldsymbol{l}}^{\intercal}\boldsymbol{h}_T)} \quad (8)$$

where K is the number of class labels and $\{\boldsymbol{W}_l\}_{l=1}^K$ are the class-specific weights of the softmax layer. The classifier can use more complicated networks depending on the task. We use this simple classifier because our main goal is to demonstrate mitigation of the exploding bias problem and report on the results achieved. For fairness, we use this approach for all methods, ours and the methods we implemented for comparison.

There are two tasks during the joint learning process, namely, imputation and classification. Let the superscript i denote the i-th sample of the time series data set $\mathbb{D}$. For the imputation task, we can obtain the imputation sequence vector $\hat{\boldsymbol{X}}^{\boldsymbol{i}} = \{\hat{x}_2^i, ..., \hat{x}_T^i\}$ of the i-th time series sample which can be divided into two parts, approximation values (orange units in Fig. 3) and imputation ones (purple units in Fig. 3). The imputation loss of all time series samples is calculated on the approximation values as follows:

$$\mathcal{L}_{\mathbf{imp}}(\boldsymbol{X}, \hat{\boldsymbol{X}}, \boldsymbol{M}) = \frac{1}{N}\sum_{i=1}^{N}\|(\boldsymbol{X}_{2:T}^i - \hat{\boldsymbol{X}}^i)\odot\boldsymbol{M}_{2:T}^i\|_2^2 \quad (9)$$

where N is the number of samples in the data set. Equation (9) is the mean squared error loss between the approximation and the revealed values. $\boldsymbol{M}_{2:T}^{\boldsymbol{i}}$ masks off the imputation values from the imputation loss since there is no ground truth for the missing values.

For the classification task, we obtain the predicted probability distribution $\hat{y}^i$ of i-th time series sample given by Equation (8), the loss of all samples of a time series can be calculated as follows:

$$\mathcal{L}_{\mathbf{cls}}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\mathbf{1}\{y^i = j\}\log\hat{y}^i \quad (10)$$

where K is the number of class labels. Equation (10) is the softmax cross entropy loss of the predicted label and the true label.

4.2 Adversarial Learning Strategy

This end-to-end framework inevitably suffers from the negative impact (error propagation from imputation to classification) of missing values because the imputed values will contain some amount of error simply due to the fact that they are predictions. This error can be quickly amplified as it is fed into the RNN, resulting in the exploding bias problem.

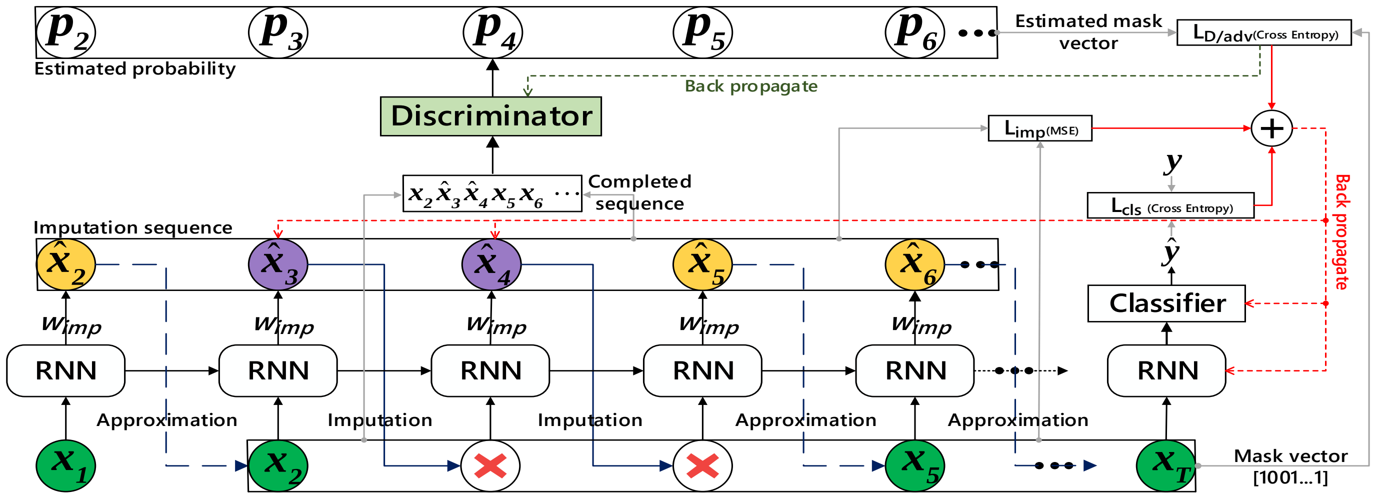


Fig. 3. The proposed AJ-RNN framework. We use green units to denote revealed inputs, yellow for the output of approximated values, purple for the imputed values and a red "X" for missing inputs. Dashed links are for approximation training and solid ones are for imputation. The discriminator receives a completed vector composed of revealed and imputed values as input, which provides a one-to-one supervisory signal for imputed values $\hat{x}_3$ and $\hat{x}_4$.

Here, we introduce a discriminator D to reduce the negative impact of missing values. The discriminator D distinguishes whether each value in the completed vector is real or imputed, rather than identifying the entire completed vector. In this way, the discriminator D can provide direct supervision to the imputed values.

Specifically, for the i-th time series training sample, there is a completed sequence vector $\boldsymbol{U}^i = \{u_2^i, ..., u_T^i\}$ given by Equation (6) and its corresponding mask vector $\boldsymbol{M_{2:T}^i} = \{m_2^i, ..., m_T^i\}$. In $\boldsymbol{U^i}$, some are real values, while the rest are imputed. We know which are which by the mask vector $\boldsymbol{M^i}$. We can take advantage of this knowledge to provide a supervision signal for the imputation network.

The adversarial learning strategy is defined as a two-player minimax game. We alternately update the parameters of discriminator D and the parameters of the AJ-RNN. First, D receives the completed sequence vector as input and is trained to distinguish which values in the completed sequence vector are revealed and which are imputed. The discriminator loss can be defined as follows:

$$\mathcal{L}_{\mathbf{D}}(\boldsymbol{U}, \boldsymbol{M}) = -[\underbrace{\mathbb{E}\log(D(X_{real}))}_{X_{real}\sim \boldsymbol{U}} + \underbrace{\mathbb{E}\log(1 - D(\hat{X}_{imp}))}_{\hat{X}_{imp}\sim \boldsymbol{U}}] \quad (11)$$

$$= -\frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{M}_{2:T}^i \odot \mathbf{log}(D(\boldsymbol{U}^i)) + (\mathbf{1} - \boldsymbol{M}_{2:T}^i) \odot \mathbf{log}(\mathbf{1} - D(\boldsymbol{U}^i))] \quad (12)$$

where $D(\cdot)$ denotes the estimated mask probability $\hat{\boldsymbol{P}}$ of the discriminator. In Equation (11), the first term is the log output of the discriminator on real values, and the discriminator tries to maximize this to 1. The second term is the loss for imputed values. Hence the discriminator tries to minimize its output for imputed values.

The discriminator is composed of 3 fully connected layers to take advantage of global contextual information (since it takes a sequence as input), where the number of units in the hidden layers are set to T (i.e., the length of the sample), $T/2$, and T, respectively. We use tanh as the activation function of each layer except for the output layer where we use the sigmoid activation function to get the estimated real/fake probability.

The RNN is thus supervised to make the distribution of predicted values is closer to the revealed ones by fooling the discriminator D. The adversarial loss of the RNN can be defined as follows:

$$\mathcal{L}_{\mathbf{adv}}(\boldsymbol{U}, \boldsymbol{M}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{1} - \boldsymbol{M}_{2:T}^i) \odot \mathbf{log}(\mathbf{1} - D(\boldsymbol{U}^i)) \quad (13)$$

Hence the AJ-RNN tries to maximize the discriminator output for imputed values. This provides a supervisory signal for each imputed value in the completed sequence vector, which can reduce the bias introduced by the imputation operation and thus alleviate the exploding bias problem. Note that we can also feed the whole imputation vector $\hat{\boldsymbol{X}}^{\boldsymbol{i}}$ into the discriminator, providing supervision for both the approximated and imputed values. In our experiments, we found that the results obtained by these two methods are similar. Hence, considering computational efficiency, we adopted Equation (13) as the adversarial loss.

4.3 The Training of AJ-RNN

Finally, the overall training loss of AJ-RNN is defined as follows:

$$\mathcal{L}_{\mathbf{AJ-RNN}} = \mathcal{L}_{\mathbf{cls}} + \mathcal{L}_{\mathbf{imp}} + \lambda_d \mathcal{L}_{\mathbf{adv}} \quad (14)$$

where λ_d is hyper-parameter. This forms an end-to-end training framework for incomplete time series classification. This loss function can be optimized by the BPTT algorithm. The training method for AJ-RNN is presented in Algorithm 1.

AJ-RNN combines the merits of joint learning and adversarial learning. The discriminator D is trained with the revealed values and the mask vector effectively provides supervision on each imputed value. Therefore, the negative impact of missing values on AJ-RNN is reduced.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Auckland University of Technology. Downloaded on October 24,2020 at 10:18:17 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 AJ-RNN Training Method

```
1: for number of training iterations do
2:   Sample minibatch of $n$ examples which contain missing value $\{(X^i, M^i, y^i)\}_{i=1}^n$ from the dataset $\mathbb{D}$
3:   for $i = 1, \dots, n$ do
4:     for $t = 1, \dots, T$ do
5:       $u_t^{(i)} = m_t^i \odot x_t^i + (1 - m_t^i) \odot \hat{x}_t^i$
6:       if $t == T$ then
7:         $\hat{y}^i \leftarrow$ AJ-RNN($u_T^i$)
8:       else
9:         $\hat{x}_{t+1}^i \leftarrow$ AJ-RNN($u_t^i$)
10:      end if
11:    end for
12:    $U^i \leftarrow \{u_1^i, \cdots, u_t^i\}$ $\hat{X}^i \leftarrow \{\hat{x}_1^i, \cdots, \hat{x}_t^i\}$
13:  end for
14:  $U \leftarrow \{U^1, \cdots, U^n\}$ $\hat{X} \leftarrow \{\hat{X}^1, \cdots, \hat{X}^n\}$
15:  $\hat{y} \leftarrow \{\hat{y}^1, \cdots, \hat{y}^n\}$ $M \leftarrow \{M^1, \cdots, M^n\}$
16:  (1) Discriminator optimization
17:  Update $D$ by SGD:
         $\theta_D = \theta_D - \nabla_{\theta_D} \mathcal{L}_{\mathbf{D}}(U, M)$
18:  (2) AJ-RNN optimization
19:  for $S_k$ steps do
20:    Update $\theta_{AJ-RNN}$ by BPTT (fixed $D$):
         $\nabla_\theta \{ \mathcal{L}_{\mathbf{cls}}(y, \hat{y}) + \mathcal{L}_{\mathbf{imp}}(X, \hat{X}, M) + \lambda_d \mathcal{L}_{\mathbf{adv}}(U, M)\}$
21:  end for
22: end for
```

Here, we discuss how our method differs from a relevant recent work. Luo et al. [28] also use a GAN for time series imputation. They develop a GAN and a discriminator that use recurrence to model time series data. However, their GAN is first trained to generate time series from random noise input, which the discriminator then has to distinguish from real time series. Once the generator is well-trained, actually imputing data requires a backpropagation step using the time series to be imputed to induce the random noise vector that will match the revealed values in the time series. Hence it does not use any joint learning.

Our adversarial strategy thus differs considerably from Luo et al. [28], in four important ways: 1) Their method is solely designed for imputation, regardless of the task. It is unclear how to extend it to include a method to incorporate the task into the generative model. 2) Since they are working on time series with missing data, the "real" time series used to train their discriminator is one with missing data. During that training, they replace the missing data with 0s (personal communication), which makes their discriminator less reliable. On the other hand, our discriminator receives a completed vector containing both revealed values and generated ones, and must discriminate between the two. 3) Their discriminator outputs one probability value for real/fake after reading the whole generated time series, while ours takes the whole time series at once, and makes a judgment on each value. Consider, for instance, if there is an outlier in the imputed values. The discriminator will produce a high loss signal for the outlier, but not for the others, which provides a more informative error signal. 4) Their model generates imputed values via an iterative process, starting from an initial noise vector, which is then adapted to the time series with missing data. We believe this process is more likely to accumulate errors than our process, which starts with the real data and then learns the imputed values.

5 Experimental Details

In this section, we introduce the datasets and describe our procedure for creating missing data. We then describe the comparison methods and implementation details. Throughout, we assume time series with regular intervals between points. We do not compare our model with models designed for irregular time series.

5.1 UCR Benchmarks

The UCR time series archive [9] contains a large number of publicly available time series data sets derived from various domains for both time series classification and clustering problems. Each data set has a default train/test split. This dataset is frequently used by researchers to evaluate the classification performance of their models. To obtain a stable numerical evaluation, we used only the data sets from the archive with a total sample size greater than or equal to 200. This results in 68 of the original 85 training sets. Their characteristics are described in Table 1 of the Supplementary Material.

We also include 4 real-world incomplete datasets from the extension of the UCR archive [9].

5.2 Creating Missing Data

In order to generate time series with missing data for training and testing, given a desired missing ratio p, we firstly create a random matrix $\mathbb{P}$ with values drawn for the range [0,1] with the same dimensions as the data $\mathbb{D}$, and then set all elements of $\mathbb{D}$ to be missing if the corresponding value in $\mathbb{P}$ is less than or equal to p. This is the same procedure used in previous work [31], [32]. Other than this, no data preprocessing is done for the 68 UCR benchmark datasets (each data set has been z-scored by the data provider). In the following experiments, the missing ratio p is varied from 20% to 80% with increments of 20% for each dataset to evaluate the models at different missing ratios. The 4 real-world incomplete data sets are provided as raw data, so we also z-score these data sets.

Note that our procedure of creating missing values on the UCR data sets is slightly different from the 4 real-world incomplete data sets, as the UCR datasets are z-scored before dropping datapoints to create missing data. This is obviously impossible with the real-world data as the missing data is actually missing, so we are z-scoring only on the actual data. We will discuss the impact of these two data pre-processing methods on the classification results in section 6.4.

5.3 Comparison Methods

For a baseline model, we used an RNN using the joint training objective, but without a discriminator (similar to those used in [17], [18]). This model is labeled J-RNN in the results section.

We also compare with methods that impute missing data first, and then classify the resulting time series. We include four representative methods from traditional approaches as well as recent research. The methods are:

- **Zero imputation**: This method simply replaces the missing value with zero.
- **Regularized EM (RegEM) imputation** [40]: This is a regularized variant of expectation maximization (EM) algorithm for imputation.
- **DynaMMo** [31]: This method is based on Expectation Maximization (EM) and the Kalman Filter. It learns a linear dynamical system in presence of missing values and imputes them.
- **TRMF** [32]: This is a recent matrix factorization framework for time series imputation and prediction, and incorporates temporal dependencies in their models.

We implement the pre-imputation methods using the code released by the original authors. We then use an RNN for classification. We note here that on the real-world datasets, we can only evaluate the classification performance and not the imputation performance, as there is no ground truth for the missing data. In any event, as suggested in [41], what one really cares about in the end is the classification performance. This is denoted ACC (for accuracy) in the tables.

5.4 Implementation Details

The pre-imputation methods described above do not have a mechanism for classification, therefore after imputation, we feed the completed data into an RNN, and use the last hidden state $\boldsymbol{h}_T$ as input to a single-layer softmax classifier, the same method we use for AJ-RNN. For the RNN, we use a single-layer RNN with 100 GRU units for all methods. The initial learning rate is set to $1e$-3, and the Adam optimizer [42] is used. The batch size is set to 20 except for the `DiaSizReduc` dataset, where it is set to 16, which is the size of the training set for this dataset. We use $L2$ regularization with a λ of $1e$-4 to avoid overfitting. All methods share the same hyperparameter settings.

For the hyperparameter λ_d of Equation (14) and the number of iterations for training the AJ-RNN, S_k, in Algorithm 1, we find that the best choices for these are 1 and 5, respectively, when the missing ratio is less than or equal to 40%. For higher missing ratios, we use 0.1 and 1, respectively. These values were found through the 5-fold cross validation on the training set (since the `UCR time series archive` does not have validation set). The criteria used was the median [43] of the last 10 test accuracies for 200 epochs averaged over the 5 folds. An intuitive explanation is that when the missing ratio is high, the discriminator's ability to guide imputation becomes weak because there is not enough data to learn from. After selecting the hyperparameters, we train the model on the entire training set and report the median [43] of the last 10 accuracy scores for 200 epochs as test accuracy. The Adam optimizer [42] is again used for training the discriminator, with the same initial learning rate of $1e$-3.

TABLE 1
Averaged results on the 68 UCR time series data sets (standard deviations are in parentheses)

Missing Ratio	Methods	AVG Acc	AVG Rank	P-value
20%	Zero	68.24(1.90)	5.809	7.641E-13
	Regem	72.22(1.25)	3.978	2.746E-11
	DynaMMo	72.69(0.99)	3.765	5.891E-11
	TRMF	72.70(1.01)	3.574	1.405E-10
	J-RNN	73.57(0.54)	2.537	1.120E-12
	AJ-RNN	**74.98(0.09)**	**1.338**	-
40%	Zero	63.78(2.40)	5.853	7.641E-13
	Regem	69.83(1.54)	3.890	8.205E-10
	DynaMMo	70.25(1.22)	3.772	3.713E-09
	TRMF	70.17(1.34)	3.669	1.203E-09
	J-RNN	71.61(1.08)	2.404	2.326E-11
	AJ-RNN	**72.45(0.50)**	**1.412**	-
60%	Zero	59.05(2.02)	5.809	7.641E-13
	Regem	65.81(1.70)	3.824	1.928E-10
	DynaMMo	65.99(1.31)	3.801	9.257E-10
	TRMF	66.01(1.50)	3.610	1.457E-10
	J-RNN	67.04(0.65)	2.654	7.641E-13
	AJ-RNN	**68.51(0.03)**	**1.301**	-
80%	Zero	54.09(4.52)	5.243	1.172E-12
	Regem	57.28(1.29)	4.074	6.231E-11
	DynaMMo	57.73(0.94)	3.721	9.941E-10
	TRMF	57.42(1.58)	3.882	8.205E-10
	J-RNN	58.73(0.81)	2.691	7.641E-13
	AJ-RNN	**59.98(0.31)**	**1.390**	-

The experiments were run on the TensorFlow platform [44] using an Intel(R) Core(TM) i7 $-$ 6850K 3.60GHz CPU, 64GB of RAM and a GeForce GTX 1080-Ti 11G graphics card. For each missing ratio, we randomly created missing data 5 times and reported the average results.

6 Results and Analysis

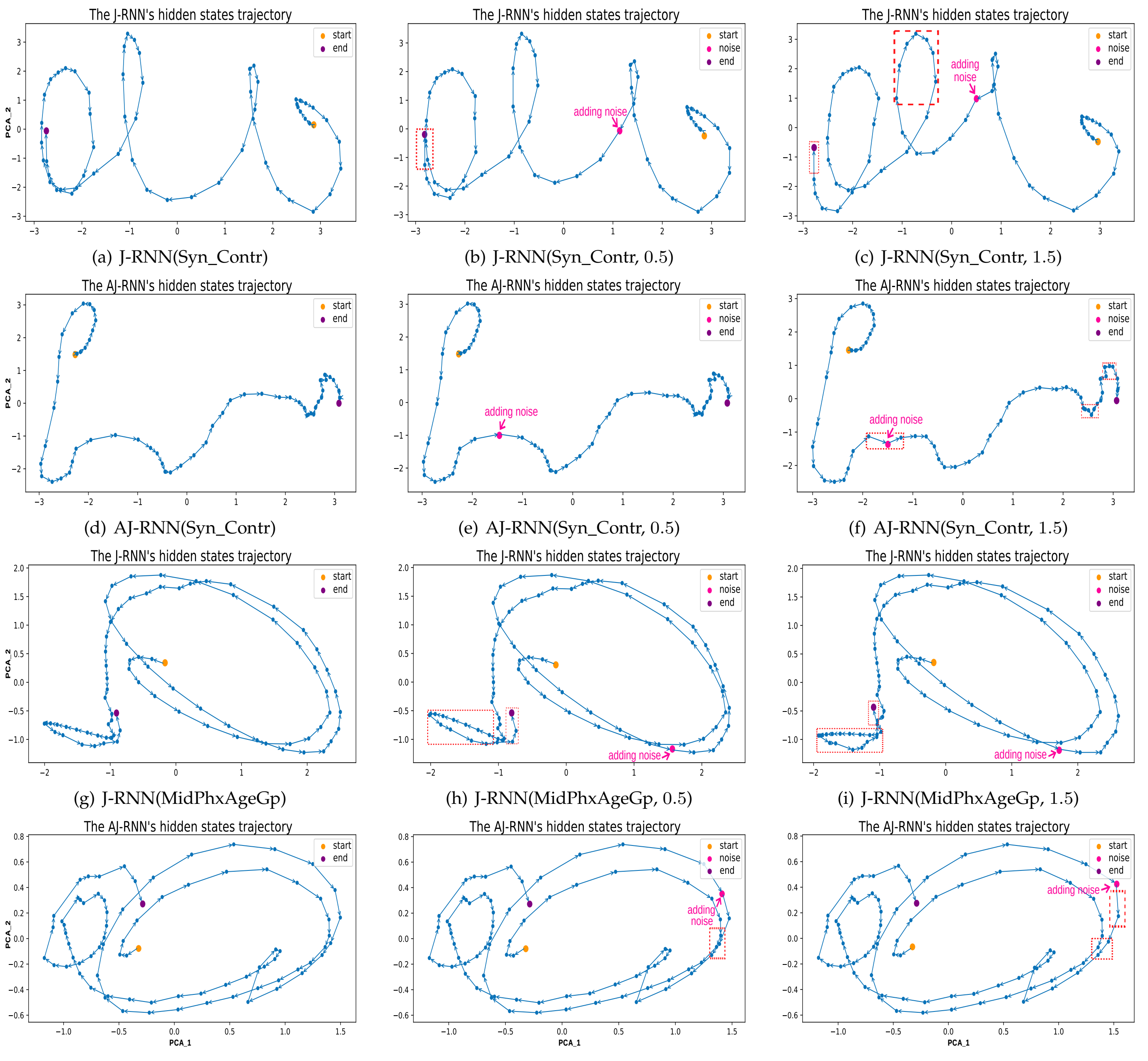
In this section, we report the results of the four imputation methods described above, our baseline J-RNN, and AJ-RNN. We then provide an analysis of the error accumulation on two datasets and the analysis of our model in order to better understand why it works. Finally, we investigate the effects of hyperparameters. Unless otherwise stated, the analysis or investigation provided here is in the MCAR setting with a missing ratio of 20%.

6.1 Performance Comparison

6.1.1 UCR datasets

The detailed results of each algorithm on the 68 `UCR` datasets are reported in Table 4 (section C) of the Supplementary Material. Here we report the average results in Table 1. While there is no "vanilla" bi-directional RNN in the literature for this problem, one might expect that the bi-directional structure may alleviate error accumulation. We present a comparison with a bi-directional RNN in section B of Supplementary Material. The comparison shows that AJ-RNN is better for addressing the problem of error accumulation.

First, as expected, the state-of-the-art imputation methods show a clear advantage over zero-imputation. Second, we can see that J-RNN is a very strong baseline, as it is already better than all of the pre-imputation methods. This shows the clear advantage of joint training, where the imputation is co-optimized with the classification; pre-imputation is suboptimal for classification. AJ-RNN achieves lowest



(a) J-RNN(Syn_Contr) (b) J-RNN(Syn_Contr, 0.5) (c) J-RNN(Syn_Contr, 1.5)

(d) AJ-RNN(Syn_Contr) (e) AJ-RNN(Syn_Contr, 0.5) (f) AJ-RNN(Syn_Contr, 1.5)

(g) J-RNN(MidPhxAgeGp) (h) J-RNN(MidPhxAgeGp, 0.5) (i) J-RNN(MidPhxAgeGp, 1.5)

(j) AJ-RNN(MidPhxAgeGp) (k) AJ-RNN(MidPhxAgeGp, 0.5) (l) AJ-RNN(MidPhxAgeGp, 1.5)

Fig. 4. The trajectories of the hidden units are plotted by performing PCA on a sample trajectory and then plotting the first two principal components. The X-axis and Y-axis are the first and second principal components of the data, respectively. The first two rows are the trajectories of J-RNN and AJ-RNN on the `Syn_Contr` dataset as the injected noise goes from 0 to 0.5, to 1.5. The last two rows are the trajectories on the data set `MidPhxAgeGp`. The arrows indicate the point in time when noise is added. The red dotted boxes indicate the area where the difference in the trajectory is obvious before and after adding noise.

rank and highest accuracy at all missing levels. The difference is statistically significant ($p << 0.001$) using a Wilcoxon signed rank test [45]. Adversarial learning is indeed boosting the classification performance of AJ-RNN.

In addition to comparison with RNNs, we also provide a comparison with the classical time series classification method 1-NNDTW [46] in section E of Supplementary Material. AJ-RNN is significantly superior to 1-NNDTW with the Wilcoxon signed rank test [45] at $p < 0.01$ level, both in terms of higher average accuracy and lower rank.

6.1.2 *UCR real-world incomplete datasets*

The results on the 4 real-world incomplete datasets are shown in Table 2. AJ-RNN again achieves the best performance. In addition to the UCR real-world datasets, we also provide the evaluation results on the incomplete clinical dataset (PhysioNet2012 [47]) in the section D of the Supplementary Material. AJ-RNN again outperforms the other methods.

6.2 Error Accumulation Analysis

Inevitably, the missing values imputed by the network should yield errors. As the imputed values are fed into the network, the errors will accumulate, which affect the classification ability of the model, especially since the classification model used here relies on the last hidden state. Here we argue that these errors will not accumulate quickly along the forward propagation of AJ-RNN since the training by the discriminator adds additional constraints on imputed values - they must plausibly fit in the context of neighboring revealed values. The idea is that the discriminator trains the hidden state trajectories to be better attractors to the dynamics induced by the inputs.

TABLE 2
The detailed test classification accuracy of 4 UCR real incomplete time series data sets

Methods \ Dataset	Dodger LpDay	Dodger LpGame	Dodger LpWend	Melbourne Pedestrian
Zero	58.75	85.51	96.38	84.95
Regem	60.00	86.23	97.10	86.26
Dynammo	62.50	86.23	97.10	86.26
TRMF	60.00	86.23	97.83	86.26
J-RNN	62.50	86.96	97.10	86.88
AJ-RNN	**65.00**	**87.68**	**98.55**	**88.43**

TABLE 3
Comparison of AJ-RNN and J-RNN's averaged distance on the Syn_Contr and MidPhxAgeGp data sets over 5 runs

Noise Level	The averaged distance			
	Syn_Contr		MidPhxAgeGp	
$U(a,b)$*	AJ-RNN	J-RNN	AJ-RNN	J-RNN
(-0.5,0.5)	**0.337**	0.608	**0.101**	0.145
(-1,1)	**0.597**	1.193	**0.227**	0.266
(-1.5,1.5)	**0.787**	1.656	**0.270**	0.324
(-2,2)	**1.168**	2.206	**0.314**	0.388

*a and b are the range of values of the noise distribution U.

To test this idea, we trained AJ-RNN and J-RNN on the time series Syn_Contr and MidPhxAgeGp. We then performed PCA on the hidden states in the two networks. The first two principal components are shown in Figure 4. We add a small amount of noise (sampled from the uniform distribution $U(a,b)$) into the input at a specific time step (40% of the length of the time series), and observe how long it takes the network hidden states to return to the true trajectory, These are the lines in Figure 4. As shown, the AJ-RNN trajectory is very close to the original trajectory when the noise is 0.5 or 1.5. However, the J-RNN trajectory is greatly offset, and the final trajectory end point is also changed. As expected, the adversarial training is causing the trajectory to be more of an attractor.

In Table 3, we quantitatively report the averaged distance between the true trajectory and the perturbed one after the perturbation for different sizes of noise on these two datasets over 5 runs. It is clear that the hidden state dynamics of AJ-RNN are better attractors than the hidden states of J-RNN. The comparison between AJ-RNN and bi-directional RNN is in section B of Supplementary Material, demonstrating that AJ-RNN is causing the trajectory to be more of an attractor again.

6.3 Model Interpretation

Here we explore the behavior of AJ-RNN in order to better understand why it works. We examine the features in the last hidden layer that are the input to the classifier and the characteristics of the imputed data, respectively.

6.3.1 Visualization of the learned features

To illustrate the robustness of the different imputation methods to missing data, we train four models on the CBF dataset: DynaMMo, TRMF, J-RNN and AJ-RNN. Using the last hidden state, which is the input to the classifier, we can see how well the different methods separate the three classes of CBF. By increasing the missing ratio, we can visualize the robustness of the methods to missing data. We use t-SNE [48] to map the hidden states into 2D for visualization purposes. The result is shown in Figure 5. In the 20% missing ratio case, we can see that the features extracted by the joint learning methods and the pre-imputation methods all form well-separated clusters. However, as the missing ratio increases, the pre-imputation methods gradually break down, while AJ-RNN (and to some extent, J-RNN) maintain the clusters fairly well. At the level of an 80% missing ratio, the features learned based on the pre-imputation methods have become very disorganized, presumably due to the accumulated error arising from inappropriate imputed values. On the other hand, AJ-RNN, while certainly making plenty of mistakes, is better able to maintain the cluster structure. J-RNN's performance is between the two. J-RNN is clearly able to impute better values for the model to use in classification. The difference between the AJ-RNN panels and the J-RNN panels reveal the advantage of discriminator training.

6.3.2 Analysis of the imputed data

The process of imputation and classification have been coupled and jointly trained in our model, which means that the imputation process has been optimized for the time series classification task. To investigate the effects of this interaction, we randomly selected 47 data sets and performed the following comparison. We use AJ-RNN to impute missing values with 5% missing levels vs the original complete data set (i.e., raw data set). We then fed them into the classic time-series classifier 1-NNDTW [46] and obtained the classification accuracies.

TABLE 4
Averaged classification results comparison of 1-NNDTW with raw data vs. 5% imputed data

Dataset	Raw_data	Imputed_data
AVG Acc	73.65	**73.93**
AVG Rank	1.745	**1.255**
P-value	2.817E-03	-

The detailed results of each algorithm are reported in Table 5 (section D) of the Supplementary Material. Here we briefly report the average results in Table 4. As inspection of Table 4 shows, the imputed data improves the accuracy of the classifier in most cases, in terms of higher average accuracy and rank. It is remarkable that the model achieves this, suggesting that our imputed values are smoother than the original data, making classification slightly easier. While the overall average accuracy of the two are only slightly different in numerical value, their performance is significantly different (see Table 5 of Supplementary Material) in terms

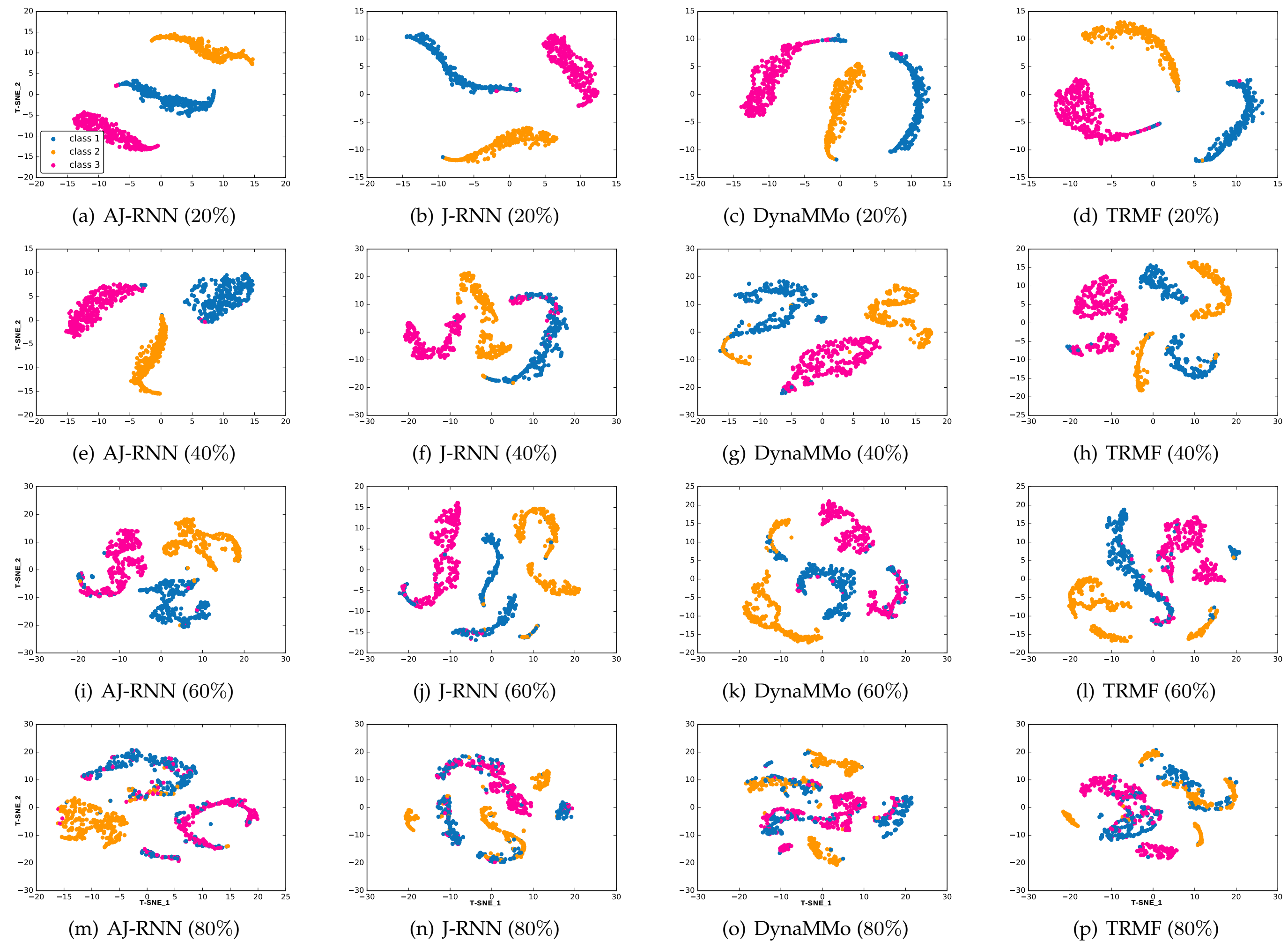


Fig. 5. The visualization results of the last hidden state $\boldsymbol{h}_T$ (the features before entering the softmax layer) on the CBF dataset with t-SNE. The X-axis and Y-axis are denoted as T-SNE_1 and T-SNE_2, respectively.

of average rank over the datasets, which is confirmed by a Wilcoxon signed rank test at the $p < 0.01$ level.

This also explains why the two-stage approach obtains a suboptimal solution, because the joint learning approach can optimize the filled-in data for classification accuracy.

6.4 The Effect of Z-scoring Before vs. After Removing Data

TABLE 5
The performance of 1-NNDTW on data z-scored **before** removing data (first column) vs. **after** (second column)

Ratio \ Dataset	GunPoint AgeSpan		GunPoint OldVsYoung		GunPoint MaleVsFemale	
20%	91.46	92.72	82.22	83.49	99.68	100.00
40%	90.82	89.56	81.59	80.95	99.05	96.84
60%	90.82	87.03	78.41	69.21	98.73	97.78
80%	78.16	70.25	71.43	67.62	93.99	94.94

We created our datasets with missing values by removing the values after the data were z-scored, as that's how they are given in the UCR database. However, for the real-world incomplete data set, the data is necessarily z-scored without the missing data. Whether the data is normalized

TABLE 6
The performance of AJ-RNN on data z-scored **before** removing data (first column) vs. **after** (second column)

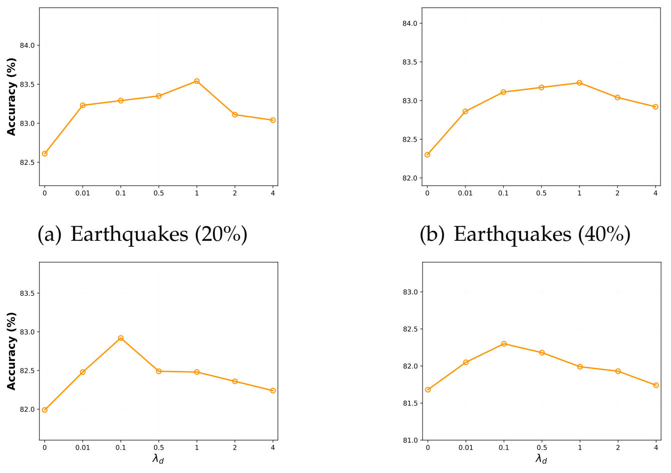
Ratio \ Dataset	GunPoint AgeSpan		GunPoint OldVsYoung		GunPoint MaleVsFemale	
20%	94.30	94.62	92.70	93.02	99.37	99.37
40%	93.99	93.67	91.43	91.75	99.05	98.73
60%	92.72	92.41	89.84	89.52	96.20	95.87
80%	86.39	87.03	80.95	80.63	94.30	94.62

with missing values will slightly affect the scale of the sequence because the missing data will affect the mean and standard deviation. Therefore, the difference between normalizing first and then removing data vs. removing data first and then normalizing is worth exploring.

Since the old UCR datasets have been normalized, we can only perform experiments on the recently expanded UCR datasets (since these have not been normalized). We here provide the experimental results of 1-NNDTW and our model AJ-RNN on the famous GunPoint dataset. As shown in Table 5, the performance of 1-NNDTW is indeed affected, especially at higher missing ratios, as would be expected. However, as shown in Table 6, this variation has little effect on AJ-RNN. As explained in [49], comparison-based algorithms should remove the impact of sequence scale, but in our case, the features learned by the RNN, which capture the dynamics of the time series, are relatively insensitive to the sequence scale.

6.5 Hyper-parameter Analysis

In this subsection, we perform an investigation of the λ_d hyper-parameter. We perform this evaluation using the classification task on the `Earthquakes` data set with different missing ratios to demonstrate how adversarial learning affects the classification performance. We do the evaluation by varying λ_d while maintaining the other parameters fixed.



(a) Earthquakes (20%) (b) Earthquakes (40%)

(c) Earthquakes (60%) (d) Earthquakes (80%)

Fig. 6. The influence of λ_d on the classification performance with the `Earthquakes` dataset under different missing ratios.

As shown in Figure 6, every non-zero value of λ_d gives better performance than $\lambda_d = 0$, so adversarial learning improves accuracy. Further, in the case where the missing ratio is 20% and 40%, the accuracy reaches its maximum at $\lambda_d = 1$ and then decreases. At the missing rate of 60% and 80%, the accuracy reaches its maximum at $\lambda_d = 0.1$ and then decreases. This phenomenon indicates that λ_d can take relatively larger values when there is more data available, which provides a more accurate gradient to mitigate error accumulation, boosting the classification performance. Furthermore, when the discriminative loss dominates the overall loss ($\lambda_d > 1$), the classification performance degrades as expected, but it is still better than not using the discriminator.

7 Conclusion

In this paper, we presented a novel framework, Adversarial Joint-learning RNN (AJ-RNN), that combines the idea of joint classification and imputation learning with adversarial learning. The model learns to conduct time series classification in the presence of missing values, but uses a discriminator to improve the imputation, making the distribution of the imputed values close to the actual ones. This approach effectively alleviates the error accumulation problem and thus reduces the negative impact of missing values on time series classification. We evaluated AJ-RNN on 68 UCR datasets with syntheically-removed data, and 4 real-world UCR incomplete time series data sets. Our results demonstrate that AJ-RNN significantly outperforms existing methods on classification with missing values. We further provide qualitative and quantitative analyses of the characteristics of the trained AJ-RNN, explaining why AJ-RNN achieves better performance.

Throughout the paper, the analysis and evaluation we provide is under the missing completely at random (MCAR) setting. Investigating how to employ and improve our adversarial learning strategy in other missing settings (MAR and MNAR) is left for future work.

Acknowledgments

The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 61502174, 61872148, 61751205), the Natural Science Foundation of Guangdong Province (Grant Nos. 2017A030313355, 2017A030313358, 2019A1515010768), the Guangzhou Science and Technology Planning Project (Grant Nos. 201704030051, 201902010020), and the Key R&D Program of Guangdong Province (Grant No. 2018B010107002).

References

[1] M. Spiotta, P. Terenziani, and D. T. Dupr, "Temporal conformance analysis and explanation of clinical guidelines execution: An answer set programming approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2567–2580, 2017.
[2] C. Hou, F. Nie, H. Tao, and D. Yi, "Multi-view unsupervised feature selection with adaptive similarity and view weight," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1998–2011, 2017.
[3] S. Lei, Z. Teng, W. Le, Z. Yue, and A. Binder, "Deepclue: Visual interpretation of text-based deep stock prediction," *IEEE Transactions on Knowledge & Data Engineering*, vol. PP, no. 99, pp. 1–1.
[4] T. Anwar, C. Liu, H. L. Vu, M. S. Islam, and T. Sellis, "Capturing the spatiotemporal evolution in road traffic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1426–1439, 2018.
[5] A. Marjaninejad, B. Taherian, and F. J. Valero-Cuevas, "Finger movements are mainly represented by a linear transformation of energy in band-specific ECoG signals," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 986–989.
[6] J. Aitchison and C. G. G. Aitken, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.
[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
[8] Cheema and R. J., "A review of missing data handling methods in education research," *Review of Educational Research*, vol. 84, no. 4, pp. 487–508, 2014.
[9] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018, https://www.cs.ucr.edu/ eamonn/time_series_data_2018/.
[10] B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan, "Strategies for handling missing data in electronic health record derived data," *eGEMs*, vol. 1, no. 3, 2013.
[11] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, p. 6085, 2018.
[12] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data*. Wiley, 1987.
[13] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, no. 34, 2017, pp. 1–5.
[14] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecastinga novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2018.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[17] L. Shen, Q. Ma, and S. Li, "End-to-End time series imputation via residual short paths," in *Proceedings of The 10th Asian Conference on Machine Learning*, 2018, pp. 248–263.

[18] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 6774–6784.

[19] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[20] S. Li and P. Kar, "Context-aware bandits," *arXiv preprint arXiv:1510.03164*, 2015.

[21] S. Li, C. Gentile, and A. Karatzoglou, "Graph clustering bandits for recommendation," *arXiv preprint arXiv:1605.00596*, 2016.

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *International Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 3, 2017, p. 4.

[24] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2017.

[25] Y. Yang, J. Zhou, J. Ai, B. Yi, A. Hanjalic, H. T. Shen, and Y. Ji, "Video captioning by adversarial lstm," *IEEE Transactions on Image Processing*, vol. PP, no. 99, pp. 1–1, 2018.

[26] J. Yoon, J. Jordon, and M. van der Schaar, "GAIN: Missing data imputation using generative adversarial nets," in *International Conference on Machine Learning*, 2018, pp. 5689–5698.

[27] S. C.-X. Li, B. Jiang, and B. Marlin, "Learning from incomplete data with generative adversarial networks," in *International Conference on Learning Representations*, 2019.

[28] Y. Luo, X. Cai, Y. Zhang, J. Xu, and Y. Xiaojie, "Multivariate time series imputation with generative adversarial networks," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 1601–1612.

[29] C. Shang, A. Palmer, J. Sun, K.-S. Chen, J. Lu, and J. Bi, "Vigan: Missing view imputation with generative adversarial networks," in *Big Data (Big Data), 2017 IEEE International Conference on*, 2017, pp. 766–775.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[31] L. Li, J. Mccann, N. S. Pollard, and C. Faloutsos, "DynaMMo: mining and summarization of coevolving sequences with missing values," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 507–516.

[32] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in Neural Information Processing Systems*, 2016, pp. 847–855.

[33] H. Lei, Y. Xia, X. Qin *et al.*, "Estimation of semivarying coefficient time series models with ARMA errors," *The Annals of Statistics*, vol. 44, no. 4, pp. 1618–1660, 2016.

[34] Z. Cai, J. Fan, and Q. Yao, "Functional-coefficient regression models for nonlinear time series," *Journal of the American Statistical Association*, vol. 95, no. 451, pp. 941–956, 2000.

[35] D. Tjøstheim and B. H. Auestad, "Nonparametric identification of nonlinear time series: Projections," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1398–1409, 1994.

[36] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie, "Stacked generative adversarial networks." in *The IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, p. 3.

[37] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI Conference on Artificial Intelligence*, 2017, pp. 2852–2858.

[38] W. Fedus, I. Goodfellow, and A. M. Dai, "MaskGAN: Better text generation via filling in the _," in *International Conference on Learning Representations*, 2018.

[39] Q. Ma, S. Li, L. Shen, J. Wang, J. Wei, Z. Yu, and G. W. Cottrell, "End-to-end incomplete time-series modeling from linear memory of latent variables," *IEEE Transactions on Cybernetics*, pp. 1–13, 2019.

[40] T. Schneider, "Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values," *Journal of Climate*, vol. 14, no. 5, pp. 853–871, 2001.

[41] M. C. De Souto, P. A. Jaskowiak, and I. G. Costa, "Impact of missing data imputation methods on gene expression clustering and classification," *BMC Bioinformatics*, vol. 16, no. 1, p. 64, 2015.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] S. Lin and G. C. Runger, "Gcrnn: Group-constrained convolutional recurrent neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4709–4718, 2017.

[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *USENIX Symposium on Operating Systems Design and Implementation)*, 2016, pp. 265–283.

[45] J. Ar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.

[46] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop. Technical Report WS-94-03*, 1994, pp. 359–370.

[47] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012," in *2012 Computing in Cardiology*. IEEE, 2012, pp. 245–248.

[48] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[49] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping," *ACM Transactions on Knowledge Discovery from Data*, vol. 7, no. 3, p. 10, 2013.

Qianli Ma (M'17) received the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 2008. He is a Professor with the School of Computer Science and Engineering, South China University of Technology. From 2016 to 2017, he was a Visiting Scholar with the University of California, San Diego.

His current research interests include machine learning algorithms, data-mining methodologies, and their applications.

Sen Li received the master's degree in computer science from the South China University of Technology, Guangzhou, China, in 2020.

His research interests include machine learning and deep learning.

Garrison W. Cottrell received his Ph.D. in computer science from the University of Rochester. He then did a postdoc with David E. Rumelhart at the Institute for Cognitive Science, University of California, San Diego. He is a Professor with the Computer Science and Engineering Department at the University of California, San Diego.

His current research interests include cognitive modeling, neural networks, and deep network modeling of the primate visual system.