Classification, Denoising, and Deinterleaving of Pulse Streams With Recurrent Neural Networks

ZHANG-MENG LIU
National University of Defense Technology, Changsha, China

PHILIP S. YU, Fellow, IEEE
University of Illinois at Chicago, Chicago, USA

Pulse streams of many emitters have flexible features and complicated patterns. They can hardly be identified or further processed from a statistical perspective. In this paper, we introduce recurrent neural networks (RNNs) to mine and exploit long-term temporal patterns in streams and solve problems of sequential pattern classification, denoising, and deinterleaving of pulse streams. RNNs mine temporal patterns from previously collected streams of certain classes via supervised learning. The learned patterns are stored in the trained RNNs, which can then be used to recognize patterns-of-interest in testing streams and categorize them to different classes, and also predict features of upcoming pulses based on features of preceding ones. As predicted features contain sufficient information for distinguishing between pulses-of-interest and noises or interfering pulses, they are then used to solve problems of denoising and deinterleaving of noise-contaminated and aliasing streams. Detailed introductions of the methods, together with explanative simulation results, are presented to describe the procedures and behaviors of the RNNs in solving the aimed problems. Statistical results are provided to show satisfying performances of the proposed methods.

Manuscript received January 24, 2018; revised June 1, 2018; released for publication September 22, 2018. Date of publication October 4, 2018; date of current version August 7, 2019.

DOI. No. 10.1109/TAES.2018.2874139

Refereeing of this contribution was handled by H. Mir.

This work was supported by the National Science Foundation of China (61771477).

Authors' addresses: Z.-M. Liu is with the State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System, National University of Defense Technology, Changsha 410073, China, E-mail: (liuzhangmeng@nudt.edu.cn); P. S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA, E-mail: (psyu@uic.edu). *(Corresponding author: Zhang-Meng Liu.)*

I. INTRODUCTION

Analyses of pulse streams play important roles in categorizing mixed data, locating emitters, and identifying their attributes [1]. Three of the most widely studied topics in the area of pulse stream processing are classification [2]–[4], denoising [5], and deinterleaving [6]. These problems are becoming more and more difficult nowadays, as the electromagnetic environment is much more crowded than ever before due to fast developments and usages of various advanced communication [7], navigation [8], and radar [9] systems.

A simple but previously effective idea for classification is categorizing pulses according to their statistical features, such as frequency, pulse width (pw), angle-of-arrival, and so on [1]. Intrapulse features embedded in emitter signals can also be exploited to categorize pulses [3], [4], but they are not available in some applications when emitter signals are not retained due to heavy storage and transmission burdens. Therefore, we only take into account the above-mentioned descriptive features to address pulse processing tasks in this paper. If pulses from more than one emitter can not be separated directly with respect to statistical features, the temporal feature of pulse repetitive interval (pri) has been exploited via numerical clustering to make in-depth analyses into interleaved streams [10], which falls into the area of denoising and deinterleaving [5], [6]. This idea exploits the pri feature also from a statistical perspective, by preassuming that typical pris of an emitter's pulse streams will be highlighted on the pri spectrum if the stream is long enough [11], [12]. Some denoising and deinterleaving methods have been proposed following this guideline, such as cumulant difference histogram (CDIF) [6] and sequential difference histogram (SDIF) [13], and they have been widely studied and used in the past few decades.

There are many significant shortcomings in these stream processing ideas and methods. First, only sufficiently long streams can be processed by them, that is because statistical features can hardly be extracted from short streams stably, such as the pri spectra [11], [12]. Moreover, counting statistical characteristics to realize pulse processing can only be realized after collecting all the pulses, which greatly blocks online applications of these algorithms. Second, features of pulses and pulse tuples are separated to obtain multiple statistical characteristics to realize stream categorization, while joint patterns between them and temporal long-term patterns are abandoned, which causes losses to available features. For example, streams consisting of pulses with feature combinations (A_1, B_1) and (A_2, B_2) are undistinguishable from ones consisting of pulses with feature combinations (A_1, B_2) and (A_2, B_1) if the features are considered separately. And long-term patterns are as important as statistical features for processing streams consisting of functional pulse tuples. Typical examples are streams of advanced electronic systems, such as imaging radar [14].

Some theoretically intense methods, such as Kalman filter based methods [15], [16], hidden Markov model based methods [17], [18], and multiple hypothesis tracking based methods [19], have also been proposed to categorize pulses of different emitters from mixed streams. They treat the stream analysis problem like a signal processing one, and introduce techniques in the latter community to solve the former problem. The algorithmic processes of these methods work only when some preassumptions are met for the streams, and they are a bit too complicated to be packaged for practical usages.

In this paper, we introduce recurrent neural networks (RNNs) [20] to address the problems of classification, denoising, and deinterleaving of pulse streams. The usage of neural networks in pulse stream processing dates back to early 1990s [21]. In previous literatures, shallow networks were reported to extract features of each pulse very well and succeed to categorize pulses according to separated or joint features [21]–[24]. However, the problem of pulse categorization is rather simple when compared with the problems of classification, denoising, and deinterleaving, and can be well solved by other numerical methods [6], [11]–[13]. Recent developments in the area of machine learning indicate that deep neural networks have much enhanced abilities of representation than shallow ones [25], and RNNs have been used to gain satisfying results on many sequence processing problems, such as machine translation [26], [27] and stock price prediction [28].

The RNN is used in this paper to extract long-term patterns (patterns that last for more than two successive pulses) from previously collected streams via supervised learning. A classification RNN is then established to map these patterns to emitter class indexes, and a group of forward/backward prediction RNNs are established to understand the current context of the pulses and predict features of upcoming pulses, so as to deal with denoising and deinterleaving problems. A new representation of pulse features is presented to prepare streams for RNN processing, and detailed explanations are made on how the RNN outputs contribute to the solution of the oriented problems. The training and using of the RNNs for classification, denoising, and deinterleaving are end-to-end. All the network parameters are tuned automatically during training based on inputted streams and outputted ground truths (class indexes of streams and features of upcoming pulses), and the trained networks output necessary information when testing streams are inputted. No expert knowledge is required during processes of training and testing of the RNNs. Simulation results also show that the RNNs are able to learn abstract (instead of determined) patterns in training streams, such as flexible scopes of constant pris and dynamic modes of stagger pris, and embody them to deterministic values according to local contexts. Statistical simulation results demonstrate the satisfying performances of the proposed methods on classification, denoising, and deinterleaving, despite of demanding settings of short streams, missing pulses, and interferential noises.

The rest of this paper is organized as follows. Section II presents a new representation of pulse streams and provide clues for potential applications of the RNN in stream processing. Then, the three problems of classification, denoising, and deinterleaving are addressed in details in Section III–V, respectively. Simulations are carried out in Section VI to demonstrate the performances of the proposed methods. Section VII makes some complementary discussions on aspects of the research that are not covered in the text. Section VIII concludes the whole paper.

II. PROBLEM FORMULATION

In most previous literatures, pulse streams have been described with multiple numerical features, including frequency, pw, time-of-arrival, etc. [1]. Such a representation well fits the requirements of statistical methods. But numerical values can hardly be understood by machines, they should be digitized and regularized to facilitate their usage in machine learning models. We present a new representation of pulse streams in this section, and analyze preliminarily why RNN techniques can be introduced to solve the tasks of classification, denoising, and deinterleaving of pulse streams.

A. Representation of Pulse Streams

Sequential patterns are intrinsic characteristics that distinguish pulse streams from random noise trains, and also from pulse streams of other emitters. These patterns contain not only statistical features of each pulse, such as frequency, pw, and angle-of-arrival, but also how pulses emerge along the time axis, which derives a new feature of pri [1]. After categorizing pulses according to preliminary statistical features, pri can be exploited as a major feature to separate pulses-of-interest from noises and aliasing pulses of other emitters [6], [13].

In this paper, we mainly exploit the pri feature to handle the tasks of classification, denoising, and deinterleaving of pulse streams, and also take pw into consideration to show how the other features can be used jointly with pri. In order to prepare for stream processing afterward, we convert the widely used numerical representations of pulse streams, i.e., $\xrightarrow{\mathrm{pri}_1} \mathrm{pw}_1 \xrightarrow{\mathrm{pri}_2} \cdots \xrightarrow{\mathrm{pri}_n} \mathrm{pw}_n \xrightarrow{\mathrm{pri}_{n+1}} \cdots$, to discrete event sequences as $\{\mathrm{pri}_1, \mathrm{pw}_1\}, \ldots, \{\mathrm{pri}_n, \mathrm{pw}_n\}, \ldots$. In both representations, we append an extra pri of 0 before the first pulse to facilitate stream description and processing. Sequential patterns of consecutive pulses can then be represented by a series of feature combinations $\{\mathrm{pri}_n, \mathrm{pw}_n\}$, where pri is defined as the interval between the current data sample and the previous one without distinguishing pulses and noises. Each of the feature combinations contains not only information about the characteristics of the pulse itself, but also the preceding context close to it. By processing feature combinations of successive pulses effectively, the sequential patterns of long pulse streams can be extracted and further exploited in testing streams.

The feature combinations are then digitized to obtain regularized formulations. We introduce two large enough upperbounds for pri and pw, and denote them by D_{pri} and D_{pw}, respectively. Outliers larger than the bounds are set to 0. Valid feature values in scopes of $[0, D_{\mathrm{pri}}]$ and $[0, D_{\mathrm{pw}}]$ are digitized linearly with respect to units of d_{pri} and d_{pw}, i.e.,

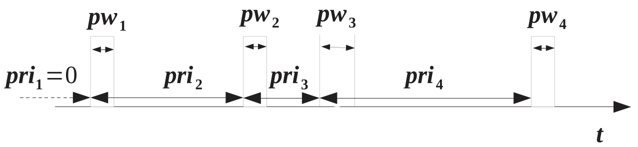


Fig. 1. Pulse stream example with three pulses and one noise.

$\text{pri}_{\text{digital}} = \lfloor \text{pri}_{\text{numeric}}/d_{\text{pri}} \rfloor$ and $\text{pw}_{\text{digital}} = \lfloor \text{pw}_{\text{numeric}}/d_{\text{pw}} \rfloor$, where subscripts $(\cdot)_{\text{numeric}}$ and $(\cdot)_{\text{digital}}$ are used to indicate numerical and digitized versions of features, $\lfloor \alpha \rfloor$ represents the largest integer not larger than α. After digitizing, pri and pw are represented by digits within scopes of $[0, \lfloor D_{\text{pri}}/d_{\text{pri}} \rfloor]$ and $[0, \lfloor D_{\text{pw}}/d_{\text{pw}} \rfloor]$ with tolerable quantization errors. As we will mainly use digitized forms of the features in this paper, the subscripts of $(\cdot)_{\text{numeric}}$ and $(\cdot)_{\text{digital}}$ will be omitted for conciseness and digitized features will be referred to by default unless otherwise stated.

In the following, we provide an example to explain the new representation of pulse streams, as is shown in Fig. 1. Four pulses are observed in total, with the three represented by rectangles come from an emitter, and the other one represented by a circle being noise. The emitter pulses have a constant pri of 800 μs. An emitter pulse is lost between the two emitter pulses with their pws labelled as pw_2 and pw_4. The noise pulse is 500 μs apart from the preceding emitter pulse. All four pulses have the same pw of 2 μs. Observation inaccuracies are not considered for the pris and pws to simplify description.

The traditional representation of the pulse stream is

$$\xrightarrow{0\ \mu\text{s}} 2\ \mu\text{s} \xrightarrow{800\ \mu\text{s}} 2\ \mu\text{s} \xrightarrow{500\ \mu\text{s}} 2\ \mu\text{s} \xrightarrow{1100\ \mu\text{s}} 2\ \mu\text{s}.$$

The feature combination sequence is

$$\{0\ \mu\text{s}, 2\ \mu\text{s}\}, \{800\ \mu\text{s}, 2\ \mu\text{s}\}, \{500\ \mu\text{s}, 2\ \mu\text{s}\},$$
$$\{1100\ \mu\text{s}, 2\ \mu\text{s}\}.$$

After that, if we choose digitizing units as $d_{\text{pri}} = 5\ \mu$s and $d_{\text{pw}} = 0.2\ \mu$s, the digitized representation of the pulse stream is rewritten as

$$\{0, 10\}, \{160, 10\}, \{100, 10\}, \{220, 10\}.$$

Each of the digitized pris and pws can be represented by a one-hot vector, with the location of its only nonzero element of 1 indicating the value of the digitized features. For example, if pri is upperbounded by $D_{\text{pri}} = 5000\ \mu$s and digitized with a unit of $d_{\text{pri}} = 5\ \mu$s, then the one-hot representation of 6.7 μs is $[0, 1, 0, 0, \ldots, 0]^T \in \mathcal{R}^{1001\times 1}$. Similarly, if pw is upperbounded by $D_{\text{pw}} = 4\ \mu$s and digitized with a unit of $d_{\text{pw}} = 0.2\ \mu$s, the one-hot representation of 0.15 μs is $[1, 0, 0, 0, \ldots, 0]^T \in \mathcal{R}^{21\times 1}$. One-hot features can be processed more easily by machine learning techniques than their numerical counterparts.

However, one-hot features are much too sparse and may make the learning process unstable. Researchers in the machine learning community have developed embedding ideas to condense the dimension of the features to stabilize the learning process [29]. This idea has gained great successes in areas of natural language processing [29], recommendation [30], and so on. According to the embedding technique, the one-hot pri and pw features can be transformed as follows:

$$\mathbf{e}_{\text{pri}} = \mathbf{E}^{(\text{pri})}\mathbf{g}_{\text{pri}} \tag{1}$$

and

$$\mathbf{e}_{\text{pw}} = \mathbf{E}^{(\text{pw})}\mathbf{g}_{\text{pw}} \tag{2}$$

where $\mathbf{g}_{\text{pri}} \in \mathbb{R}^{L_1\times 1}$ and $\mathbf{g}_{\text{pw}} \in \mathbb{R}^{L_2\times 1}$ are one-hot pri and pw vectors, $\mathbf{E}^{(\text{pri})} \in \mathbb{R}^{l_1\times L_1}$ and $\mathbf{E}^{(\text{pw})} \in \mathbb{R}^{l_2\times L_2}$ are embedding matrices for the two features with $l_1 \ll L_1$ and $l_2 \ll L_2$, $\mathbf{e}_{\text{pri}} \in \mathbb{R}^{l_1\times 1}$, and $\mathbf{e}_{\text{pw}} \in \mathbb{R}^{l_2\times 1}$ are embedded vectors. The embedding matrices should be initialized properly and trained via supervised learning.

Embedding matrices in (1) and (2) act like look-up tables. When a one-hot feature is given, one column of the matrix is selected according to the location of the nonzero vector element to represent the feature. The embedded features are then fed to the neural networks as a train of inputs. Well designed neural networks are required to extract inner patterns within successive pulses, so as to identify different emitters, and be aware of pulse contexts to distinguish pulses from outliers.

B. How RNN Fits the Processing Tasks

Deep learning techniques have been developing fast in the past decade, and systems based on the techniques have been reported to reach or even surpass the level of human beings [31], [32]. Two kinds of neural networks have been widely used for deep learning, i.e., convolutional neural networks and RNNs [25], [33]. The RNNs are designed for processing sequential data and have gained great successes in areas, such as machine translation and finance [26]–[28]. They process sequential samples one by one to extract information contained in streams [26], [27], or predict upcoming samples based on previous ones [28]. Some of the successful applications of RNN are similar to the pulse stream processing problems in this paper.

Emitters radiate pulse streams to implement certain functions, which behave like human beings speaking out sentences to express themselves. Sentences have been processed with RNN for sentiment analysis in recent years to judge whether the attitude of the speaker is positive, negative, or neutral [34]. The networks read the embedded words one by one, and extract information contained in key words and phrases to distinguish between different sentiments. Such networks are also expected to be able to process pulse features sequentially, and extract distinguishing patterns between pulse streams of different emitters. There are preliminary patterns, such as particular values of pri and pw, and also complicated patterns, such as dynamic modes of successive pris. By extracting distinguishable patterns from pulse streams, the trained RNN may succeed to solve the problem of stream classification.

Another widespread application area of RNN is sequence prediction [28], [35]. RNNs have been used to predict upcoming data samples to forecast stock prices [28]

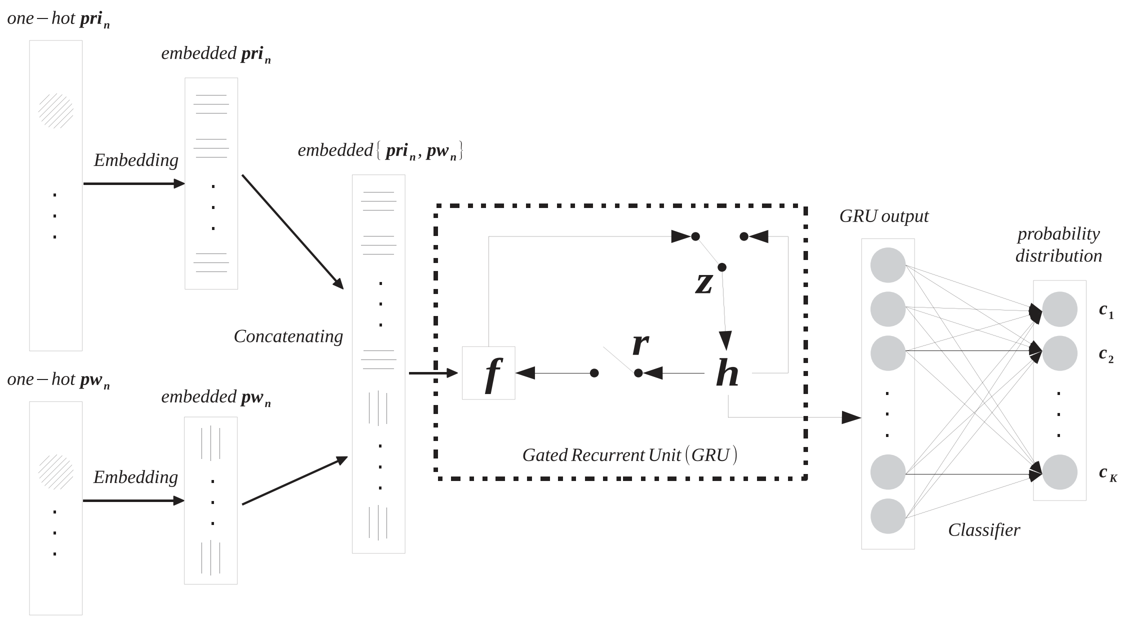


Fig. 2. Structure of the RNN for classification. Digitized one-hot features are embedded and concatenated to obtain an input vector **f** of the GRU unit; the GRU unit processes the input vector and output state vector **h**; a fully connected layer transforms the GRU output to a probability distribution vector.

and protein structures [35]. They extract stream patterns from previously collected data, understand current context based on preceding samples, and then succeed to predict upcoming samples. Pulse streams are generally regularized data trains, consisting of constant, periodic, or combinational feature patterns. Features of pulses in the near future are predictable based on these of the past ones. However, practical pulse streams may be contaminated by noises or interferential pulses, which cause negative effects to pattern identification and prediction. Therefore, the RNN will be introduced in this paper to predict features of upcoming pulses based on that of the past ones, and distinguish between expected pulses and outliers, so as to realize stream denoising and deinterleaving.

There have been long-lasting researches on RNN [33], [36], and great progresses have been made in the machine learning community since the proposal of long short-term memory (LSTM) in 1997 [37]. LSTM introduces a forgetting mechanism into the original RNN framework, which well solves the problem of vanishing gradients and succeeds to extract long-term patterns. The progresses contribute a lot to the widespread applications of RNN [25], [33], [38]. Recently, a gated recurrent unit (GRU) is proposed as a substitution of LSTM [39], [40]. GRU has fewer tunable parameters than LSTM, and gains comparable performances in many applications. In this paper, we establish GRU-based RNNs to solve the problems of classification, denoising, and deinterleaving of pulse streams.

III. RECURRENT LEARNING FOR CLASSIfiCATION

In this section, we present our ideas for classifying pulse streams with RNN. Detailed discussions will be provided on the structure of GRU, how the embedded features are fed into RNN for pattern learning, and how the trained networks are exploited for testing.

A. Structure of RNN for Classification

A sketch of the classification RNN is shown in Fig. 2. Digitized pris and pws at time instants of each pulse are first represented with one-hot vectors, and then embedded to lower dimensional features according to (1) and (2). The two embedded features are then concatenated to form a combined feature of $\mathbf{x}_n = [\mathbf{e}_{\mathrm{pri},n}^T, \mathbf{e}_{\mathrm{pw},n}^T]^T$, which is the input to the GRU module. This module extract sequential patterns contained in pulse streams, and store them in the state of GRU, which is denoted by $\mathbf{h}_n$ and also treated as the output of this module. Finally, a fully connected layer is appended to the GRU module to map its state vector to a probability distribution along different classes.

Detailed algorithmic procedures of the GRU module are described in (3)–(6) [39], [40]. Equation (3) describes the update gate of GRU. In this equation, $\mathbf{x}_n$ represents the concatenation of embedded pri and pw at time instant n, $\mathbf{h}_{n-1}$ represents the GRU state at time instant $n-1$ and $\mathbf{h}_0 = \mathbf{0}$, with each time instant corresponding to the arrival of a data sample (may be a pulse or noise). The two vectors are transformed linearly with respect to matrices $\mathbf{W}^{(u)}$ and $\mathbf{U}^{(u)}$, and then added with a bias vector $\mathbf{b}^{(u)}$ to obtain the final update vector $\mathbf{z}_n$ via a logistic sigmoid function $\sigma(\cdot)$. In (4), a reset vector is obtained in a similar way with the same input vectors but different tunable parameters. Another group of tunable parameters of $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are introduced to map inputs $\mathbf{x}_n$ and $\mathbf{h}_{n-1}$ to memory vector $\mathbf{f}_n$ using the new reset vector $\mathbf{r}_n$ and a hyperbolic tangent function tanh$(\cdot)$. In (6), the GRU state is updated from $\mathbf{h}_{n-1}$ to $\mathbf{h}_n$ based on the inputs and newly obtained update and reset vectors, and $\mathbf{1}$ is an all-one vector. In the equations, $\odot$ stands for element-wise multiplication, and the dimensions of tunable matrices and vectors can be derived from context

$$\mathbf{z}_n = \sigma(\mathbf{W}^{(u)}\mathbf{x}_n + \mathbf{U}^{(u)}\mathbf{h}_{n-1} + \mathbf{b}^{(u)}) \tag{3}$$

$$\mathbf{r}_n = \sigma(\mathbf{W}^{(r)}\mathbf{x}_n + \mathbf{U}^{(r)}\mathbf{h}_{n-1} + \mathbf{b}^{(r)}) \tag{4}$$

$$\mathbf{f}_n = \tanh(\mathbf{W}\mathbf{x}_n + \mathbf{r}_n \odot \mathbf{U}\mathbf{h}_{n-1} + \mathbf{b}) \tag{5}$$

$$\mathbf{h}_n = \mathbf{z}_n \odot \mathbf{f}_n + (\mathbf{1} - \mathbf{z}_n) \odot \mathbf{h}_{n-1}. \tag{6}$$

Denote the dimension of the GRU state vector by l, i.e., $\mathbf{h}_n \in \mathbb{R}^{l\times 1}$, and the number of candidate classes by K, the GRU state is finally mapped to a probability distribution on classes $c_1, c_2, \ldots, c_K$ via a fully connected layer. The mathematical model of this layer is

$$\hat{\mathbf{p}} = s(\mathbf{W}^{(o)}\mathbf{h}_n + \mathbf{b}^{(o)}) \tag{7}$$

where $\mathbf{W}^{(o)} \in \mathbb{R}^{K\times l}$ is a weight matrix, $\mathbf{b}^{(o)} \in \mathbb{R}^{l\times 1}$ is a bias vector, $s(\cdot)$ is the softmax function.

Each element of the output vector $\hat{\mathbf{p}}$ indicates the probability that the pulse stream belongs to a certain class. The class with the largest probability is then chosen as the classification result.

B. Training of RNN Classifier

Many parametric weight matrices and bias vectors are introduced for the implementation of RNN-based classification. They should be tuned to output correct class labels for pulse streams. An efficient way to tune the parameters is supervised learning. That is, a set of pulse streams tagged with true class labels is fed into the RNN, a probability distribution will be computed based on current network parameters. The estimated probability distribution is then compared with the given label to calculate a loss, which measures the deviation between the estimated and targeted probability distributions. Finally, the parameters are tuned to decrease the loss to obtain better classification results. After several rounds of parameter tuning, the trained RNN is expected to perform satisfyingly in classifying test streams.

Before training starts, the GRU state $\mathbf{h}_0$ is initialized to $\mathbf{0}$, and all the weight matrices and bias vectors are initialized randomly according to Gaussian distributions with variance 0.1. Each time when a new data sample is observed, the corresponding pw and pri are digitized and transformed to one-hot vectors. The one-hot vectors are fed into the RNN, and then embedded and concatenated to form an input vector $\mathbf{x}_n$ of the GRU module. The GRU module processes the input vector according to (3)–(6) and finally output a state vector $\mathbf{h}_n$, which is updated recurrently when new data samples arrive. When the last sample of a stream has been processed, the final state vector is inputted to the fully connected layer to obtain a probability distribution estimate $\hat{\mathbf{p}} = [\hat{p}_1, \ldots, \hat{p}_K]^T \in \mathbb{R}^{K\times 1}$ on the K classes.

The ground truth of the probability distribution estimate associated with a stream is $\mathbf{p} = [0, \ldots, 0, 1, 0, \ldots, 0]^T \in \mathbb{R}^{K\times 1}$, with the location of the only nonzero element indicating the true label. The estimate of the probability distribution vector may deviate from its truth for both zero and nonzero elements, i.e., the estimated probability for the true class is smaller than 1, and these for the other classes are larger than 0. A loss function is defined following the binary cross-entropy criterion to evaluate the deviation of the estimated probability vector from its groud truth, i.e.,

$$\text{loss} = -\Sigma_{k=1}^{K}[p_k \log(\hat{p}_k) + (1 - p_k)\log(1 - \hat{p}_k)]. \tag{8}$$

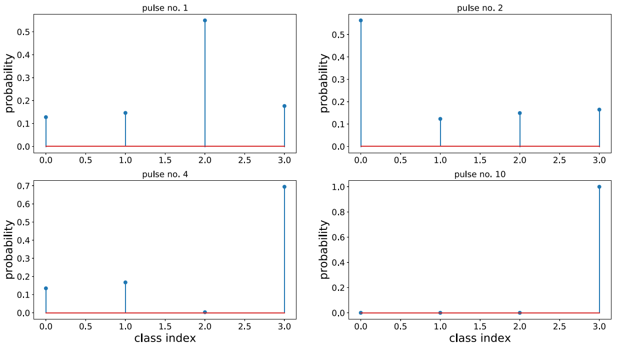


Fig. 3. Probability distribution estimates of the classifier RNN at different time instants when tested on a pulse stream with stagger pris.

The loss function reaches the only minimum of 0 when $\hat{\mathbf{p}} = \mathbf{p}$, and has positive values otherwise. The farther that $\hat{\mathbf{p}}$ deviates from $\mathbf{p}$, the larger the loss will be. The training process tunes RNN parameters to minimize the loss, and gradually modifies the RNN to become a better classifier.

Backpropagation is the most widely used method for minimizing loss functions in neural networks [41], [42]. It calculates the derivations of the loss function with respect to each tunable parameter [43], including elements of embedding matrices, weight matrics, bias vectors, and then modify the parameters along the opposite direction of the derivations, i.e.,

$$\alpha_{\text{new}} = \alpha_{\text{old}} - \eta\frac{\partial \text{loss}}{\partial \alpha} \tag{9}$$

where α represents any one of the tunable parameters, η is a self-defined positive learning rate smaller than 1. Many literatures have made in-depth discussions on the details of backpropagation of RNN [43], [44], and most machine learning platforms, such as Pytorch [45] and Tensorflow [46], provide callable instructions for computing the gradients automatically. Therefore, we skip over details of the backpropagation process here and refer readers interested in it to these literatures and platform documents for detailed explanations.

Parameter settings and dataset descriptions for training and testing the classifier RNN are delayed to Section VI. Missing pulses and observation noises are included in the datasets to better simulate practical pulse streams. As an example to show how RNN behaves in classification, we feed a noise-free stream with stagger pris to the RNN for testing. The streams of this emitter have undistinguishable statistical features from these of the other emitters, which have constant pris, but have different long-term patters that last for several successive pulses. Fig. 3 shows dynamic probability distributions of the trained RNN after receiving the first, second, fourth, and tenth pulses.

When only one or two pulses have been received, the RNN can not collect enough information to distinguish between different classes; thus, all the four classes have significantly nonzero probabilities. After receiving four pulses, the probability of the third class vanishes to 0 as it has a different pw from the received pulses, and the fourth class now has a much larger probability than the first and second classes, which have undistinguishable statistical features from the fourth class. This phenomenon indicates that the RNN has succeeded to gathered temporal patterns from very limited successive pulses to complement statistical features. When as many as ten pulses are processed, the probability associated with the fourth class overwhelms these associated with the other ones, and the outputted probability vector approaches the ground truth of $[0, 0, 0, 1]^T$ very well.

IV. FEATURE PREDICTION FOR STREAM DENOISING

In practical applications, pulse streams may be contaminated by interferential pulses from other emitters or random noises, which are called observation noises or outliers in this paper. Measurement inaccuracies in the features will not be considered except in the simulations in Section VI, and the term “noise” is used to indicate outliers exclusively to avoid confusions unless otherwise stated.

In this section, we assume that pulse streams have been classified correctly beforehand using the RNN presented in Section III, and establish another RNN to realize feature prediction of upcoming features and deal with the problem of stream denoising.

A. Structure of RNN for Feature Prediction

Statistical features such as pw can be exploited beforehand to filter out outliers that diverge largely in appearance from pulses-of-interest, and the remained outliers are generally undistinguishable from pulses directly. A feasible way to separate pulses from outliers is to make use of contextual features, e.g., pri, according to a criterion that whether the context centered at a certain data sample obeys the pattern of an emitter’s pulse streams.

The feature prediction RNN has a structure shown in Fig. 4. The digitized one-hot features are first embedded, concatenated, and recurrently processed in the same way as that in the classifier RNN shown in Fig. 2. The GRU output state is then inputted to a fully connected layer to predict features of the upcoming pulse separately. We take pri and pw as pulse features in this paper, and mathematical models of the output layer are

$$\hat{\mathbf{p}}_n^{(\mathrm{pri})} = s(\mathbf{W}^{(\mathrm{pri})}\mathbf{h}_n + \mathbf{b}^{(\mathrm{pri})}) \tag{10}$$

$$\hat{\mathbf{p}}_n^{(\mathrm{pw})} = s(\mathbf{W}^{(\mathrm{pw})}\mathbf{h}_n + \mathbf{b}^{(\mathrm{pw})}) \tag{11}$$

where superscripts $(\cdot)^{(\mathrm{pri})}$ and $(\cdot)^{(\mathrm{pw})}$ indicate variables associated with features of pri and pw, respectively, subscript $(\cdot)_n$ indicates variables corresponding to the nth time instant, and $s(\cdot)$ stands for the softmax function.

The final outputs associated with the features are normalized vectors, and they have the same dimensions as the inputted one-hot feature vectors. Each element of the output vector indicates the probability of the next pulse feature taking a certain digitized value. Take the pulse stream in Fig. 1, for example. Suppose that the prediction RNN is able to distinguish the pulse features and temporal patterns of this stream based on the first two pulses, i.e., constant pw of 2 μs and constant pri of 800 μs. Then, bias-free pri and pw prediction vectors should be $\mathbf{p}_3^{(\mathrm{pri})} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$, with 1’s index being $161, 321, 481, \ldots$ to indicate pri values of 800 μs, 1600 μs, 2400 μs, ... by taking missing pulses into account, and $\mathbf{p}_3^{(\mathrm{pw})} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$, with 1’s index being 11 to indicate a pw value of 2 μs. According to the predictions, the noise that departs from the second emitter pulse by 500 μs does not have an expected time-of-arrival, thus, is distinguished as noise and skipped over to the next data, which obeys the predictions and is processed as a pulse for further predictions.

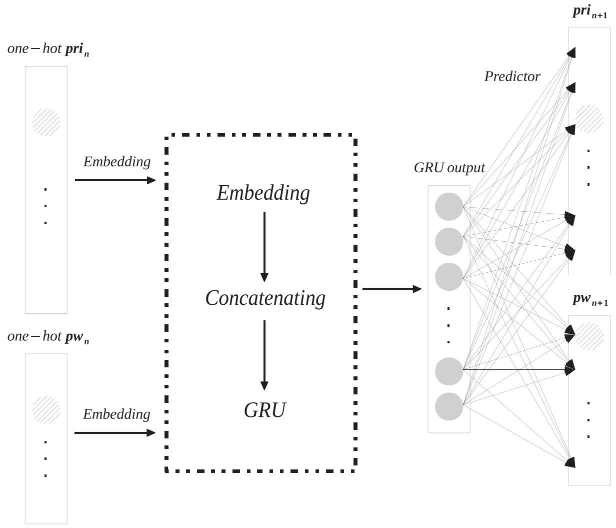


Fig. 4. Structure of the RNN for next-pulse feature prediction. The network has the same structure as the classifier RNN, except that the final outputs divide into multiple vectors associated with different pulse features.

B. Training of RNN Predictors

The prediction RNN is trained with labeled pulse streams beforehand to predict features of upcoming pulses. Our training strategy of the prediction RNN is shown in Fig. 5(a), where rectangles stand for pulses and circles for noises, dashed, and arrowed lines point to pulses to be predicted at each time instant. Pulse streams belonging to known classes are simulated or collected with electrical systems, and noises are added to the streams artificially to enhance the robustness of trained RNNs. Each of the data sample in the noise-contaminated streams are tagged with a pulse or noise label. The streams are fragmentary as some of the pulses are missed with a certain probability, and the added noises interrupt the pri features.

During RNN training, data samples in the stream are processed sequentially without discriminating pulses and noises. However, only pulse features are taken as ground truths for RNN prediction. Moreover, both forward and

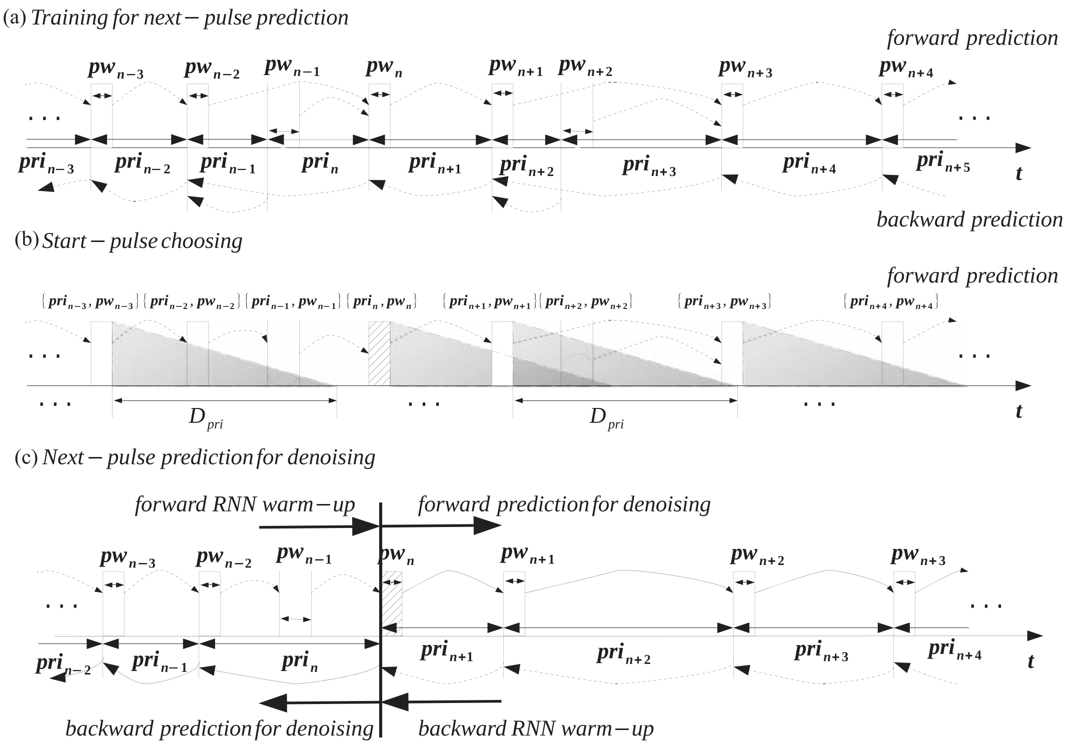

(a) Training for next − pulse prediction
forward prediction
pw_{n-3} pw_{n-2} pw_{n-1} pw_n pw_{n+1} pw_{n+2} pw_{n+3} pw_{n+4}
...
pri_{n-3} pri_{n-2} pri_{n-1} pri_n pri_{n+1} pri_{n+2} pri_{n+3} pri_{n+4} pri_{n+5}
t
backward prediction
(b) Start − pulse choosing
forward prediction
$\{pri_{n-3}, pw_{n-3}\}$ $\{pri_{n-2}, pw_{n-2}\}$ $\{pri_{n-1}, pw_{n-1}\}$ $\{pri_n, pw_n\}$ $\{pri_{n+1}, pw_{n+1}\}$ $\{pri_{n+2}, pw_{n+2}\}$ $\{pri_{n+3}, pw_{n+3}\}$ $\{pri_{n+4}, pw_{n+4}\}$
D_{pri}
D_{pri}
t
(c) Next − pulse prediction for denoising
forward RNN warm − up
forward prediction for denoising
pw_{n-3} pw_{n-2} pw_{n-1} pw_n pw_{n+1} pw_{n+2} pw_{n+3}
pri_{n-2} pri_{n-1} pri_n pri_{n+1} pri_{n+2} pri_{n+3} pri_{n+4}
t
backward prediction for denoising
backward RNN warm − up

Fig. 5. Sketch map for training and using prediction RNN. (a) Forward and backward RNN are trained by processing pulses up to a certain time instant and taking next-pulse features as ground truths. (b) Pulse, together with its context, that best obeys the learned sequential patterns is chosen as a start-pulse for denoising. (c) Forward and backward denoising are applied after warming up.

backward RNNs are trained for each class to mine and exploit bidirectional patterns in pulse streams. We take the four-sample stream in Fig. 1 for example to explain more concretely the training process. The stream samples have tags PPNP, where P stands for pulse and N for noise. The numerical representation of the stream is $\xrightarrow{\text{pri}_1} \text{pw}_1 \xrightarrow{\text{pri}_2} \text{pw}_2 \xrightarrow{\text{pri}_3} \text{pw}_3 \xrightarrow{\text{pri}_4} \text{pw}_4$, and the event-like representation is $\{\{\text{pri}_1, \text{pw}_1\}, \{\text{pri}_2, \text{pw}_2\}, \{\text{pri}_3, \text{pw}_3\}, \{\text{pri}_4, \text{pw}_4\}\}$ with $\text{pri}_1 = 0$. The feature combinations are processed by a forward RNN one-by-one, and the corresponding ground truths at the four time instants are $\{\{\text{pri}_2, \text{pw}_2\}, \{\text{pri}_3 + \text{pri}_4, \text{pw}_4\}, \{\text{pri}_4, \text{pw}_4\}, \{-, -\}\}$, where the noise is skipped over in the expected prediction outputs and the last processing step has no outputs. The inversed formulation of the stream is $\{\{\text{pri}_1, \text{pw}_4\}, \{\text{pri}_4, \text{pw}_3\}, \{\text{pri}_3, \text{pw}_2\}, \{\text{pri}_2, \text{pw}_1\}\}$ with $\text{pri}_1 = 0$, and the ground truths of prediction are $\{\{\text{pri}_4 + \text{pri}_3, \text{pw}_2\}, \{\text{pri}_3, \text{pw}_2\}, \{\text{pri}_2, \text{pw}_1\}, \{-, -\}\}$.

Denote the outputted feature vectors of the prediction RNN by $\hat{\mathbf{p}}_n^{(\text{pri})}$ and $\hat{\mathbf{p}}_n^{(\text{pw})}$, and the corresponding ground truths by $\mathbf{p}_n^{(\text{pri})}$ and $\mathbf{p}_n^{(\text{pw})}$, which are one-hot formulations of the upcoming pulse features. Then, the prediction loss at time instant n is

$$\text{loss}_n = \eta_1 \text{loss}_n^{(\text{pri})} + \eta_2 \text{loss}_n^{(\text{pw})} \tag{12}$$

where $\text{loss}_n^{(\text{pri})}$ and $\text{loss}_n^{(\text{pw})}$ are cross-entropy loss functions between $(\hat{\mathbf{p}}_n^{(\text{pri})}, \mathbf{p}_n^{(\text{pri})})$ and $(\hat{\mathbf{p}}_n^{(\text{pw})}, \mathbf{p}_n^{(\text{pw})})$ defined in a similar way as that in (8), and η_1 and η_2 are weight coefficients indicating different importances of pri and pw prediction accuracies. In this paper, we set $\eta_1 = \eta_2 = 1$ for simplicity.

The total loss of the RNN for predicting a whole noise-contaminated pulse stream is calculated by cumulating losses at each time instant, i.e.,

$$\text{Loss} = \Sigma_{n=1}^{N} \text{loss}_n \tag{13}$$

where N represents the length of the stream. It should be noted that some of loss_n are set to 0 when n is close to N and there are no upcoming pulses. The total loss is then used to tune RNN parameters via backpropagation as that in (9).

Another important issue in feature prediction is the scope of pri and pw, i.e., D_{pri} and D_{pw} in Section II-A. D_{pw} can be set according to the statistical mean and standard deviation (STD) of pws in a certain class, and the setting of D_{pri} is relatively complicated. More concretely, D_{pri} should be large enough for the RNN to predict the time instant of the next pulse in despite of missing pulses, and meanwhile, too large pri s may be very difficult to predict as they seldom emerges in training datasets. Detailed settings of D_{pw} and D_{pri} will be made clear in Section VI, and we mention them here to complete the introduction for testing the prediction RNN.

C. Using RNN Predictors for Denoising

In the classification task, the RNN is trained and tested in the same way by processing each pulse stream from start to end, and outputs a classification result when all the pulses have been processed. However, denoising via next-pulse prediction is a sequential task, only pulse features (with noises skipped over) are taken as ground truths of RNN output during training. The testing process is dynamic and more complicated. Data samples of a to-be-denoised stream should be distinguished one-by-one in a temporal order to determine whether they are pulses or noises. Pulses will be remained for future processing and predicting, and noises will be skipped over to avoid causing interferences to temporal patterns of successive pulses.

A brief sketch of using the trained prediction RNN for denoising is shown in Fig. 5(b) and (c). The pulse stream is first processed with the forward RNN sample-by-sample, since no prior information can be exploited to distinguish pulses from noises. A confidence degree is computed for each sample as follows:

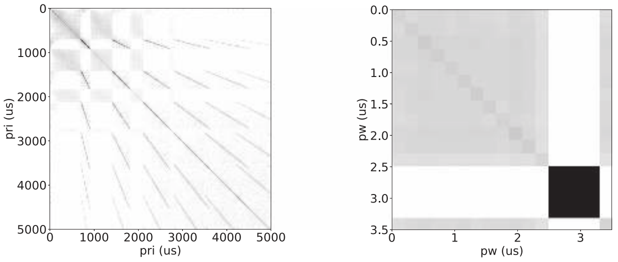
$$\hat{\mathbf{P}}_n = \Sigma_{q=1}^{Q} \hat{p}^{(\mathrm{pri})}_{i_q} \hat{p}^{(\mathrm{pw})}_{j_q} \tag{14}$$

where Q represents the number of data samples with temporal distances smaller than D_{pri} from the current sample, i_q and j_q are indexes of digitized temporal distance and width of the qth upcoming sample in $\hat{\mathbf{p}}^{(\mathrm{pri})}$ and $\hat{\mathbf{p}}^{(\mathrm{pw})}$. For example, in Fig. 5(b), there are two samples with temporal distances smaller than D_{pri} with respect to the pulse indexed by $n-3$, thus $Q = 2$, i_1 and i_2 equal to digitized values of pri_{n-2} and $\mathrm{pri}_{n-2} + \mathrm{pri}_{n-1}$, and j_1 and j_2 equal to digitized values of pw_{n-2} and pw_{n-1}.

The confidence degree indicates how much the context of each sample obeys the temporal pattern of emitters in a certain class, the better the pattern is obeyed, the larger the degree will be. Two major factors contribute to the degree. One is whether the prediction RNN is well warmed-up based on previous data samples, the other is how much the subsequent samples within a temporal scope of $[0, D_{\mathrm{pri}}]$ is consistent with the predicted features in $\hat{\mathbf{p}}^{(\mathrm{pri})}$ and $\hat{\mathbf{p}}^{(\mathrm{pw})}$. Noises generally have small confidence degrees, since the RNN state may be significantly interfered by processing a new {pri, pw} combination of a noise, which does not obey the stream pattern, and features of the pulses following up can hardly be predicted as they do not have regularized pri s with respect to the noise. The larger the confidence degree is, the more probable the data sample is actually a pulse, and the better the features of upcoming pulses can be predicted. Therefore, we tag the sample with the largest confidence degree as a pulse, and choose it as the start-point of the forward/backward denoising processes.

After selecting the starting pulse, we first warm up the forward prediction RNN by processing the substream before the selected pulse, as is shown in Fig. 5(c). Then, in the rest of the stream, we update pri and pw probability distributions of $\hat{\mathbf{p}}^{(\mathrm{pri})}$ and $\hat{\mathbf{p}}^{(\mathrm{pw})}$ after processing each pulse, and distinguish each upcoming pulse by evaluating how much its features match the predictions. All samples with temporal distances smaller than D_{pri} from the current pulse are evaluated according to a matching score defined as

$$\beta = \hat{p}^{(\mathrm{pri})}_{i} \hat{p}^{(\mathrm{pw})}_{j} \tag{15}$$



(a) (b)

Fig. 6. Row-wise coherence of the weight matrices in the output layer of the denoising RNN. (a) $\mathbf{W}^{(\mathrm{pri})}$. (b) $\mathbf{W}^{(\mathrm{pw})}$.

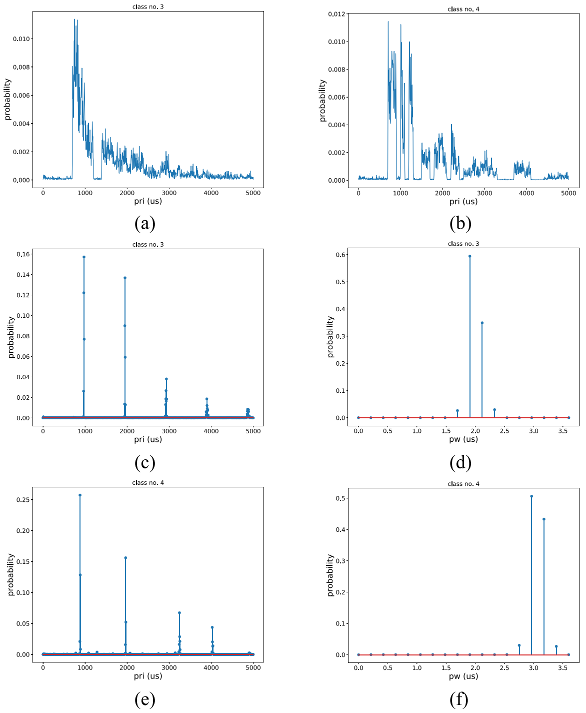
where i and j are indexes of the sample's digitized temporal distance and pw in $\hat{\mathbf{p}}^{(\mathrm{pri})}$ and $\hat{\mathbf{p}}^{(\mathrm{pw})}$. The sample with the largest matching score is chosen as the next pulse, and all the other samples between the current and the chosen pulses are tagged as noises.

When the forward denoising process has been completed, we exploit the trained backward RNN to denoise the preceding substream of the starting pulse. The tagged noises are skipped over during the warm-up procedure of the backward RNN, and the prediction for denoising procedure begins at the initially chosen starting-pulse, as is shown in Fig. 5(c).

A major concern may arise on the ability of the denoising RNN for predicting diversed feature values. Take pri for example, the time instant of the next pulse may be pri, 2pri, ... in a stream with constant pri due to pulse missing. As pri is significantly nonzero, these feature candidates are much diverged in value, then how can the prediction result $\hat{\mathbf{p}}^{(\mathrm{pri})}$ be able to indicate all the possibilities of the next pri?

Actually, all information about the possibilities of candidate feature values are contained in the GRU output $\mathbf{h}_n$, which is then mapped onto the feature spectra $\hat{\mathbf{p}}^{(\mathrm{pri})}_n$ and $\hat{\mathbf{p}}^{(\mathrm{pw})}_n$ using mapping matrices $\mathbf{W}^{(\mathrm{pri})}$ and $\mathbf{W}^{(\mathrm{pw})}$. The similarity of different feature values in the prediction space can be indicated by the coherence between different rows of $\mathbf{W}^{(\mathrm{pri})}$ and $\mathbf{W}^{(\mathrm{pw})}$. In Fig. 6, we plot grey images of the coherence matrices associated with a denoising RNN trained for an emitter with constant pri between 700 μs and 900 μs and perturbed pw with a mean of 3 μs. It can be concluded from the matrices that rows of $\mathbf{W}^{(\mathrm{pri})}$ corresponding to harmonics of a certain pri within [700 μs, 900 μs] are exclusively correlated with each other, which means that if the GRU module is able to learn the constant pri pattern from preceding pulses, the output layer will map the GRU state to harmonics of the stream's basic pri. The pw cohenrence matrix also indicate that values close to 3 μs are strongly recommendated by the RNN. Therefore, if the RNN is able to learn stream patterns from preceding pulses correctly, the output layer will map the state vector to numerically diversed but contextually close feature candidates.

In Fig. 7, we provide more concrete results to describe the behaviors of the denoising RNN in feature prediction. Two forward denoising RNNs are trained for two emitters in Section VI, one has a constant but unknown pri



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 7. Behaviors of forward denoising RNN in predicting next-pulse features. (a) pri prediction spectrum of the second pulse in a constant pri stream. (b) pri prediction spectrum of the second pulse in a stagger pri stream. (c) pri prediction spectrum after processing ten pulses in a stream with constant pri of 975 μs. (d) pw prediction spectrum after processing ten pulses in a stream with perturbed pw having a mean of 2 μs. (e) pri prediction spectrum after processing ten pulses in a stream with stagger pri of 780 μs/880 μs/1080 μs/1280 μs. (f) pw prediction spectrum after processing ten pulses in a stream with perturbed pw having a mean of 3 μs.

within the scope of [700 μs, 1200 μs], its pw has a mean of 2 μs and STD of 0.1 μs; the other has stagger pris $\tau/(\tau + 100\mu\text{s})/(\tau + 300\,\mu\text{s})/(\tau + 500\,\mu\text{s})$ with τ selected randomly within [700 μs, 800 μs] for each stream, its pw has a mean of 3 μs and a STD of 0.1 μs. Then, two noise-free complete pulse streams with a constant pri of 975 μs and stagger pris with $\tau = 780$ μs are inputted to the two RNNs, respectively, for feature prediction. Detailed settings of the two emitters can be found in Section VI labelled as emitters of Class 3 and 4.

Fig. 7(a) and (b) shows the $\hat{\mathbf{p}}^{(\text{pri})}$ s of the two RNNs after receiving the first pulse. Although no information about the pri is contained in the first data sample, the predicted pri spectra are still able to indicate pri scopes of the next pulse. The second pulse of the stream with constant pri is expected to emerge after 700–1200 μs with high probabilities, and if this pulse is missed, the temporal distance will be 1400–2400 μs with lower probabilities. This scope overlaps with the distance scope of [2100 μs, 3600 μs] when two successive pulses are both missed. The next pri of the stagger stream has more special characteristics, as is shown in Fig. 7(b). Each of the single pris in the stagger pri sequence of 780 μs/880 μs/1080 μs/1280 μs and the sum of any two or more successive pris are indicated by the probability distribution.

When as many as ten pulses have been received and processed, the prediction spectra of the next pri and pw are shown in Fig. 7(c)–(f). The predicted pw [see Fig. 7(d)] and pri [see Fig. 7(c)] of the first stream concentrate around 2 μs and on harmonics of the basic pri of 975 μs, and slight deviations exist due to measurement noises in the training dataset. For the stream with stagger pris, the tenth pulse with a pri of 780 μs has just been processed, and the upcoming pris should be 880 μs, 1080 μs, 1280 μs, 780 μs, ... in a temporal order, therefore, the time instant of the next pulse should be 880 μs, 1960 μs, 3240 μs, 4020 μs, etc. Candidates within [0, 5000 μs] have all been indicated by the predicted pri spectrum in Fig. 7(e). The predicted pw concentrates around 3 μs in Fig. 7(f).

The results in Figs. 6 and 7 imply that the prediction RNNs are able to mine and store abstract temporal patterns in streams of a certain class during training, and understand the current context and embody originally dim patterns according to detailed features in the testing stream, such as refining pri values within a certain scope or determining a certain stagger mode. Moreover, different probabilities of feature estimates also indicate diverged chances of upcoming pris and pws. If pulses of a stream are missed with the same probability ρ_m, then the cumulant probability of missing q successive pulses is $\rho_m^q(1 - \rho_m)$, which decays with q. Therefore, probabilities of predicted pris according to higher order harmonics decrease in Fig. 7(a)–(c) and (e).

V. DEINTERLEAVING VIA ITERATIVE DENOISING

Interleaved pulse streams contain multiple substreams radiated by different emitters. From the perspective of each emitter, the interleaved stream can be deemed as a noise-contaminated one, and may be processed using the denoising RNNs. Actually, the denoising RNNs do help to solve the deinterleaving problem, but two major differences between the denoising and the deinterleaving problems should be considered during this process. First, it is difficult to classify interleaved streams using the classification RNN proposed in Section III before deinterleaving them. That is because the classification RNN is designed to output a single class label, and is unable to analyze the substream components in detail and output multiple labels for them. Second, a deinterleaving task requires extracting each of the substreams associated with the interleaved emitters, instead of focusing on a certain substream and eliminating the rest ones.

Due to these reasons, we skip over the classification procedure before deinterleaving, and integrate all prediction RNNs corresponding to each candidate class to propose a deinterleaving sketch, as is shown in Fig. 8. Substreams belonging to each emitter are extracted from the original stream in an iterative manner, and the remained stream is processed according to the same procedures until very few pulses are left behind.

Each iteration of stream deinterleaving follows a similar procedure as denoising. But as the stream class is not known beforehand, all the trained forward denoising RNNs

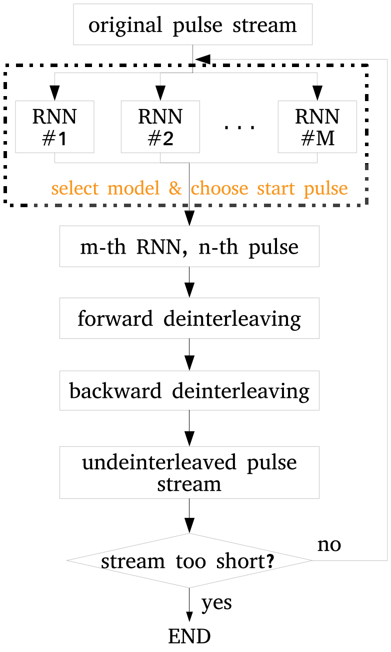


Fig. 8. Integrated deinterleaving sketch based on trained denoising RNNs of all classes.

are used to process the interleaved stream and choose a starting-point according to (14). Then, the largest confidence degree associated with the mth denoising RNN and the nth data sample is chosen for starting the current deinterleaving iteration. Forward and backward prediction based deinterleaving procedures are then implemented in the same way as that shown in Fig. 5(c), and a substream can be extracted when the backward deinterleaving process is finished. The deinterleaved pulses are deleted from the original stream, and the remained stream is inputted for a new deinterleaving iteration.

In Fig. 9, we provide a descriptive sketch of the deinterleaving procedures by showing detailed steps for deinterleaving a stream consisting of rectangles and triangles. Two groups of forward and backward denoising RNNs are assumed to have been trained beforehand for the rectangle streams and triangle streams, respectively, and we denote them by REC-RNN and TRI-RNN. In step (a), both forward REC-RNN and TRI-RNN are used to process the stream sequentially, and confidence degrees are calculated for each data sample according to (14). Suppose that the line-filled rectangle has the largest confidence degree according to the REC-RNN, the REC-RNN, and the line-filled rectangle are chosen for starting this deinterleaving iteration. Forward and backward denoising procedures are implemented in turn to extract rectangles from the original stream. This deinterleaving iteration may be not perfectly accurate, and some of the rectangles are left behind in the remained stream when this iteration is finished. Then, the remained stream is processed in a second iteration. By choosing the TRI-RNN model and a proper starting pulse, the triangle substream can be extracted via forward and backward deinterleaving procedures. When the second iteration finishes, only a few rectangles and triangles are left behind. The remained stream is shorter than a user-defined threshold and the deinterleaving procedure is terminated.

VI. Simulations

In this section, we carry out simulations to demonstrate the performances of the RNN-based classification, denoising, and deinterleaving methods. The simulation settings are much different from the ones addressed in previous literatures. The streams considered in this paper are generally very short, incomplete and noise-contaminated, and pulses of different emitters can hardly be distinguished using statistical features. Most of existing methods fail in such circumstances and cannot be used as baseline methods for performance comparison. Therefore, we concentrate mainly on the behaviors of the proposed methods in this section.

A. Simulation Settings

Five classes of pulse streams are considered in this section, with their attributes listed in Table I. Stream features of different classes overlap in a statistical perspective, and can hardly be categorized straightforwardly.

Gaussian distributed deviations (STD) are added to pri and pw observations to simulate measurement noises, and the pris of emitters belonging to the first four classes are set within a certain scope to increase the difficulty of the processing tasks. The fifth emitter class has stagger pw and pri features that can hardly be distinguished from each of the other classes directly, so as to further increase the difficulty of classification and deinterleaving. In each of the training and testing streams, a certain τ is selected and kept unchanged throughout the stream.

Missing pulses and observation noises are included in training and testing streams of the classification and denoising tasks. Each pulse is assumed to miss with a certain probability ρ_m, and noises are added between each two adjacent pulses with their number obeying a poisson distribution with a mean of $\rho_n(1-\rho_m)$. In this way, the ratio of noise number to pulse number is guaranteed to be ρ_n in average in the streams. Noise pws obey the same distribution as pulse pws in the same stream, and their time instants are selected randomly. In interleaved streams, no noises are included, and substreams belonging to different emitters are generated with the same missing probability ρ_m and then interleaved with respect to an initial time shift randomly chosen within $[0, D_{\text{pri}}]$. The number of samples (including both pulses and noises) in each stream in denoising tasks and also that of pulses in each substream in deinterleaving tasks are set randomly within [20, 25]. Stream lengths in relatively easier classification tasks are shortened to the scope of [10, 15]. During RNN training for both classification and prediction, ρ_m and ρ_n are fixed at 0.5 and 0.2, respectively. But different values of ρ_m and ρ_n are chosen during testing, so as to show the generalization ability of the trained networks. A deinterleaving process is terminated when the remained stream is shorter than 10 or the iteration number exceeds 5.

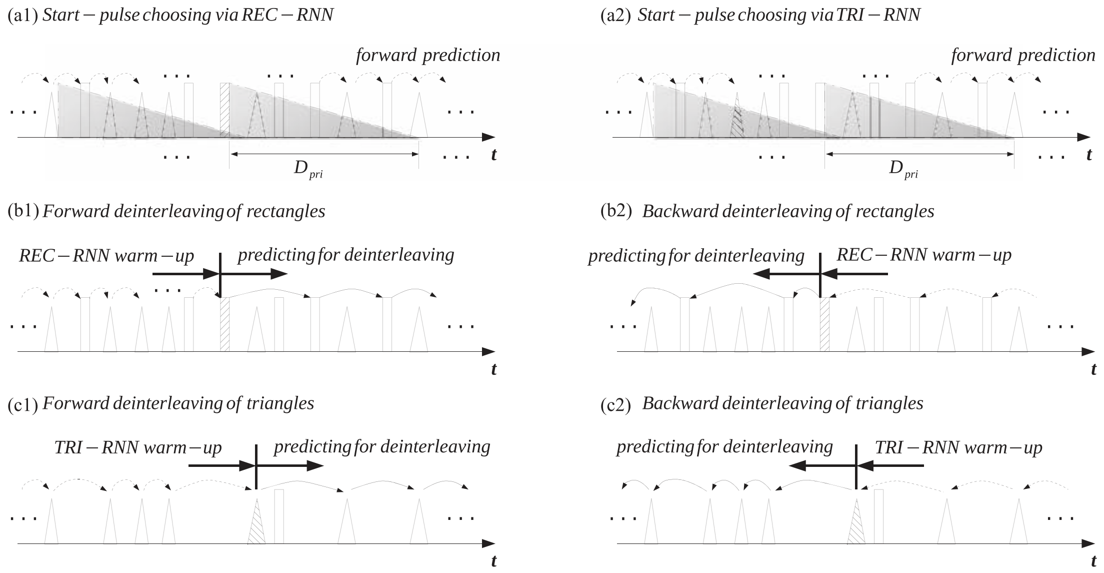
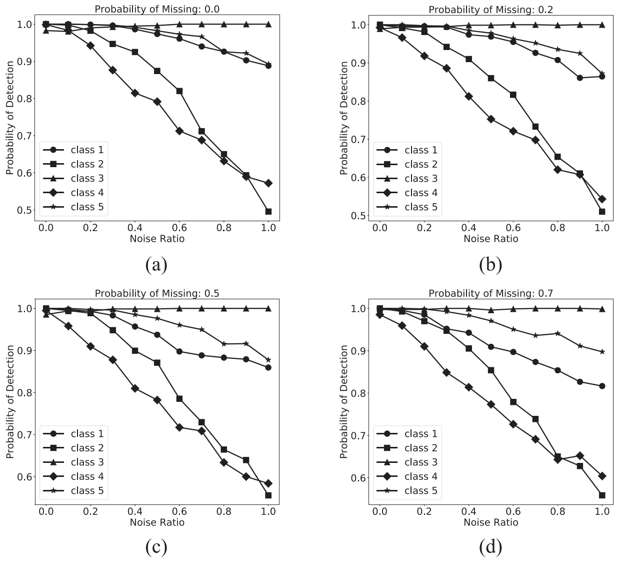

(a1) Start − pulse choosing via REC − RNN
forward prediction
· · ·
D_{pri}
t
(a2) Start − pulse choosing via TRI − RNN
forward prediction
· · ·
D_{pri}
t
(b1) Forward deinterleaving of rectangles
REC − RNN warm − up
predicting for deinterleaving
t
(b2) Backward deinterleaving of rectangles
predicting for deinterleaving
REC − RNN warm − up
t
(c1) Forward deinterleaving of triangles
TRI − RNN warm − up
predicting for deinterleaving
t
(c2) Backward deinterleaving of triangles
predicting for deinterleaving
TRI − RNN warm − up
t

Fig. 9. Visual sketch of deinterleaving a stream consisting of rectangles and triangles. (a) Choose denoising RNN and starting pulse by comparing all candidate RNN and pulse choices. (b) Deinterleaving rectangles via forward and backward denoising. (c) Deinterleaving triangles via forward and backward denoising.

TABLE I
Attributes of Simulated Stream Classes

	pw type	*pw* mean	*pw* STD	*pri* type	*pri* mean	*pri* STD
class 1	constant	3us	0.1us	constant	$\tau \in [700us, 900us]$	2us
class 2	constant	3us	0.1us	constant	$\tau \in [1000us, 1200us]$	2us
class 3	constant	2us	0.1us	constant	$\tau \in [700us, 1200us]$	2us
class 4	constant	3us	0.1us	stagger	$\tau/(\tau+100)/(\tau+300)/(\tau+500)$ with $\tau \in [700us, 800us]$	2us
class 5	stagger	2us/2us/3us/3us	0.1us	stagger	$\tau/(\tau+100)/(\tau+300)/(\tau+500)$ with $\tau = 800us$	2us

A classification RNN is trained for distinguishing each stream among the five classes in Table I, and a group of forward/backward prediction RNNs are trained for each of the classes. Ten thousands of streams are generated to train each of the RNNs, while the forward and backward prediction RNN of the same class share the same group of training datasets. The RNNs are trained on the platform of Pytorch [45] with a batch size of 64 and a learning rate of $\eta = 0.05$. Each of the batch is selected randomly from the corresponding dataset, and 5000 batches are selected in total to train each RNN. The pri feature is upperbounded by $D_{\text{pri}} = 5000\ \mu$s, and digitized with a unit of $d_{\text{pri}} = 5\ \mu$s, and the pw feature is upperbounded by $D_{\text{pw}} = 3.5\ \mu$s, and digitized with a unit of $d_{\text{pw}} = 0.2\ \mu$s. Thus, the dimensions of the one-hot pri and pw features are $L_1 = 1001$ and $L_2 = 18$. The two features are embedded to vectors with dimensions $l_1 = 120$ and $l_2 = 8$ according to (1) and (2), respectively, and the dimension of the GRU input $\mathbf{x}$ has a dimension of 128. The dimension of the GRU output vector $\mathbf{h}$ is also set to be $l = 128$.



(a) (b) (c) (d)

Fig. 10. Classification performance of streams belonging to the four classes for different ρ_m s when ρ_n varies from 0 to 1. (a) $\rho_m = 0.0$. (b) $\rho_m = 0.2$. (c) $\rho_m = 0.5$. (d) $\rho_m = 0.7$.

B. Classification

Five thousands of testing streams are generated in total, with each stream belonging to one of the five classes with randomly chosen τ according to Table I. Four pulse missing probabilities of $\rho_m = 0.0, 0.2, 0.5, 0.7$ are set for the streams, and noise-to-pulse ratio ρ_n varies from 0.0 to 1.0. The classification performance is evaluated in terms of detection probabilities, i.e., the ratio of correct classification number to total stream number for each class. Fig. 10

shows detection probabilities of all the five classes under different settings.

It can be concluded from Fig. 10 that streams belonging to the third class are classified with probability 1 in all environments. That is because their pw are totally 2 μs, while the pws of the other classes are totally or partially 3 μs. The pw feature distinguishes streams belonging to the third class well from these belonging to the other classes. Such a result indicates that different features provide complementary information in solving steam processing tasks. The classification performance of the fifth class deteriorates slightly from that of the third class, that is mainly because the pw feature now partially overlaps with that of the other three classes, and the difference will be more difficult to track in cases of large missing probabilities and noise ratios. Detection probabilities of streams in classes 2 and 4 decrease with ρ_n in a similar manner for different ρ_m s. When ρ_n is as large as 1.0, the classification probabilities is still higher than 50%. The reason for the performance deteriorations is that observation noises emerge between pulses randomly, causing great damages to originally regularized pri patterns, and making them undistinguishable from streams in the other classes. The classification probabilities of streams in class 1 are very high when ρ_m is small, and only decrease slightly with increasing ρ_n. However, for larger ρ_m s, the probabilities decrease more quickly with ρ_n. That is because only large pris exist in streams with high missing probabilities, and the smaller pris, which distinguish class 1 from classes 2, 4, and 5, disappear partially in the streams.

C. Denoising

A sketch of training and testing the forward and backward denoising RNNs for each class is shown in Fig. 5. A slight modification is made in the simulations to take measurement inaccuracies of pri and pw into consideration, i.e., a small neighborhood of 5 units are included to calculate the confidence degree of each sample for prediction, and $\hat{p}_i^{(\mathrm{pri})}$ and $\hat{p}_j^{(\mathrm{pw})}$ are replaced by $\Sigma_{\Delta=-2}^{2}\hat{p}_{i+\Delta}^{(\mathrm{pri})}$ and $\Sigma_{\Delta=-2}^{2}\hat{p}_{j+\Delta}^{(\mathrm{pw})}$, respectively, in (14) and (15). Three hundreds of noise-contaminated streams are generated for each class to evaluate the performance of the method in each environment.

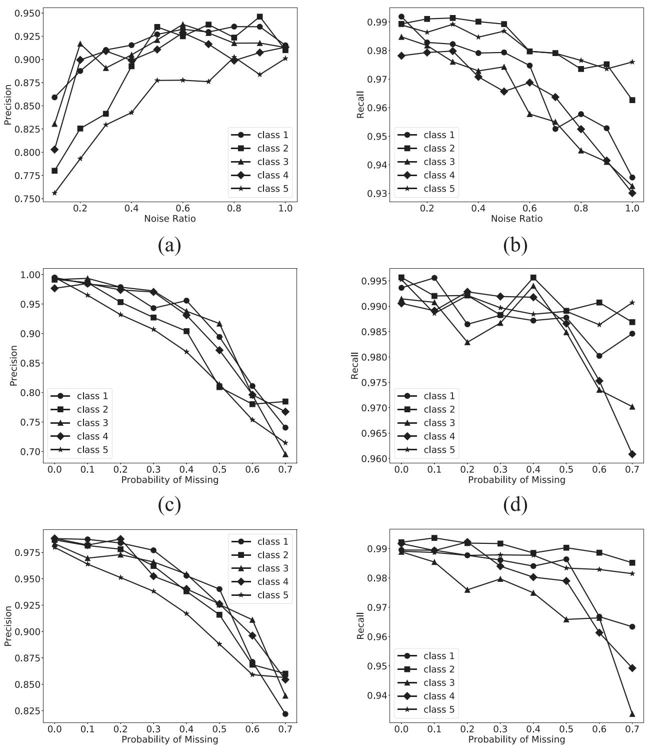
As the aim of denoising is to distinguish observation noises from pulses, we count the numbers of three different kinds of pulses for the performance evaluation:

1) true positive (TP): noises being detected as noises;
2) false positive (FP): pulses being detected as noises;
3) false negative (FN): noises being detected as pulses.

And two criteria of precision and recall are calculated to evaluate denoising performances [47]

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{17}$$



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 11. Denoising performance of streams belonging to the four classes for different ρ_m s and ρ_n s. (a) and (b) Show precision and recall with fixed $\rho_m = 0.5$ and varying ρ_n from 0.1 to 1. (c) and (d) Show precision and recall with fixed $\rho_n = 0.2$ and varying ρ_m from 0 to 0.7. (e) and (f) Show precision and recall with fixed $\rho_n = 0.5$ and varying ρ_m from 0 to 0.7.

The criterion of precision indicates the ratio of actual noises in the collection of samples being detected as noises, and recall indicates how many of the observation noises are correctly detected as noises.

In the simulations, we fix one of ρ_m and ρ_n and vary the other one within a certain scope. The obtained statistical precision and recall are shown in Fig. 11. In Fig. 11(a) and (b), ρ_m is fixed at 0.5 and ρ_n varies from 0.1 to 1, and precision increases with ρ_n while recall decreases. That is because the number of FP does not change largely for different ρ_n s, and precision increases with the number of TP when ρ_n increases and more noises are contained. But for larger ρ_n s, noises have more chances to emerge at neighborhoods of missed pulses and will be misdetected as pulses, so recall decreases slightly with ρ_n. However, recall is still larger than 90% when ρ_n is as larger as 1, which indicates that the prediction RNNs have satisfying abilities in distinguishing pulses from noises even in highly noisy environments.

We then fix ρ_n at 0.2 and 0.5 and vary ρ_m to obtain precision and recall in Fig. 11(c) and (d) and Fig. 11(e) and (f), respectively. The recalls are close to 1 when the streams are complete, i.e., $\rho_m = 0$, and are still larger than 0.96 and 0.93 for $\rho_n = 0.2$ and $\rho_n = 0.5$ when ρ_m is as large as 0.7. The results again demonstrate the ability of

TABLE II
Probabilities of Correct Deinterleaving

Probability of Missing: 0.3					
	c_1	c_2	c_3	c_4	c_5
c_1	0.920	0.977	0.981	0.965	0.922
c_2		0.920	0.983	0.960	0.931
c_3			0.954	0.984	0.936
c_4				0.933	0.921
c_5					0.864

Probability of Missing: 0.5					
	c_1	c_2	c_3	c_4	c_5
c_1	0.902	0.966	0.980	0.955	0.913
c_2		0.913	0.981	0.959	0.925
c_3			0.934	0.970	0.917
c_4				0.873	0.910
c_5					0.839

Probability of Missing: 0.7					
	c_1	c_2	c_3	c_4	c_5
c_1	0.865	0.941	0.966	0.927	0.904
c_2		0.892	0.954	0.924	0.915
c_3			0.902	0.959	0.904
c_4				0.840	0.899
c_5					0.810

the prediction RNNs in distinguishing pulses from noises. However, precision decreases faster with respect to ρ_m than recall, that is mainly because more pulses are not detected during prediction due to heavier interferences in RNN states caused by noises. Moreover, we obtained larger precisions for $\rho_n = 0.5$ than $\rho_n = 0.2$, just because TP is larger in the former case.

D. Deinterleaving

Interleaved streams are deinterleaved with the trained denoising RNNs following the sketch shown in Fig. 8. Pulses in aliasing streams may be deinterleaved correctly or incorrectly, or left behind when the deinterleaving process is terminated. The performance of deinterleaving is evaluated according to the ratio of correctly deinterleaved pulses to the length of the original stream.

The missing probabilities of the substreams are set to be $\rho_m = 0.3, 0.5$, and 0.7. For each ρ_m and each combination of two streams generated from the class set $\{c_1, c_2, c_3, c_4, c_5\}$, 300 streams are generated for deinterleaving, and the obtained statistical probabilities of correct deinterleaving are shown in Table II.

Most of the deinterleaving probabilities are larger than 92% when $\rho_m = 0.3$, and do not decrease significantly when ρ_m increases from 0.3 to 0.7, which indicates that the prediction RNNs perform satisfyingly even when the streams are highly incomplete and interfered heavily by other emitter streams. Aliasing streams can be deinterleaved more easily when one of the substream belongs to class 3, as pulses of streams belonging to this class have a more distinguishable pw feature, and its pri feature is simple. Interleaved streams corresponding to the class combination of $\{c_1, c_2\}$ also have very large deinterleaving probabilities for different ρ_m s, that is because streams belonging to the two classes both have easily recognizable constant pri patterns. However, as substreams of class 4 have more complicated stagger pris, and the pri values overlap with that of the other classes, the interleaved streams with a substream generated from class 4 have slightly smaller deinterleaving probabilities than streams generated from class combinations of $\{c_1, c_2, c_3\}$. Substreams generated from class 5 have coupled pw and pri patterns, both of which are stagger and change synchronously, the complex patterns make their pulses very difficult to track. As a result, when a substream of the interleaved streams is generated from class 5 and the other substream from one of the other classes, the deinterleaving probabilities further lowers but is still higher than 90% in most of the cases considered.

A more demanding task is deinterleaving streams consisting of substreams with similar or the same patterns, such as combinations of two substreams generated from the same class in Table I. The deinterleaving probabilities in such environments decrease by about 5% to 10% due to the hardly undistinguishable patterns of the two interleaved substreams. The deinterleaving performance becomes the worst when both substreams are generated from class 5, that is because the pw pattern is stagger and the pris are determined (instead of chosen randomly from a scope and, thus, can be diverse in the two substreams), making it more difficult to track along a certain substream and distinguish between the two substreams with the same patterns. In despite of the performance deteriorations, we can still obtain deinterleaving probabilities higher than 80% in all the simulations. As traditional deinterleaving methods, such as CDIF [6] and SDIF [13], are unable to deinterleave such substreams having the same statistical features, the proposed method has gained great performance margins over its existing counterparts.

VII. FURTHER DISCUSSIONS

This paper introduces RNNs to solve the problems of classification, denoising, and deinterleaving of pulse streams. The networks have been demonstrated to be able to "remember" statistical and local patterns of streams, and can also embody the originally abstract patterns in a stream after processing a few pulses. Actually, the proposed ideas can be extended in various aspects to gain performance enhancements, or solve new problems that have not been addressed in previous literatures. Due to space limitations, we only provide brief discussions on some of them in this section, and leave in-depth research for future work.

1) *Prediction for classification:* A major reason for the deterioration of the classification performances in Fig. 10 is that noises and missing pulses damage temporal patterns of streams. But the results in Fig. 11 and Table II demonstrate that the prediction RNNs are robust to these imperfections in streams. Moreover, the step-wise prediction results in Fig. 7 show that the difference in pre-

dicted patterns of streams corresponding to different classes are obvious. Performance improvements are expected to be obtained by using the prediction RNNs to solve the classification problem.

2) *Adaptation to streams with more complicated patterns*: Only streams with constant or stagger pws and pris are considered in the simulations in this paper mainly for reasons of enhancing readability of the text and visualization of the results. The results in Fig. 7 have partially shown that the trained prediction RNNs are able to store temporal patterns and predict multiple features of upcoming pulses. For some advanced emitters with more complicated patterns, the RNNs can also be trained to extract and store the manner of the joint variations of multiple features, and predict the features of upcoming pulses when the preceding ones are given. The predicted features can be exploited to realize denoising and deinterleaving of streams with more complicated patterns.

3) *Application in online processing*: In this paper, the problems of classification, denoising, and deinterleaving are solved in an offline manner, just for convenience of the performance evaluation. But actually, both the classification RNN and the forward prediction RNN process the streams sequentially. This processing manner adapts to online processing requirements very well (the backward prediction procedure can be abandoned here). In online processing tasks, pulses and noises are received one-by-one, the RNNs can be used to reveal stream attributes and identify local patterns in current contexts, and then classify the stream or predict features of upcoming pulses to realize classification, denoising, and deinterleaving on line.

VIII. CONCLUSION

In this paper, several RNN frameworks are established based on the GRU, so as to address problems of classification, denoising, and deinterleaving of pulse streams. The RNNs are trained via supervised learning and tested in various environments. Typical simulation results show that the RNNs are able to mine and abstract statistical and local patterns of streams with different feature types, and the mined patterns can be used easily to refine classification and prediction results when only a few pulses are available. Statistical performances show that the proposed methods solve the classification, denoising, and deinterleaving problems well in despite of demanding simulation settings, such as short stream lengths, large pulse missing probabilities, and noise ratios. Classification and denoising performances generally deteriorate with increasing pulse missing probabilities and noise ratios. But in most of the environments considered, the performances deteriorate not very significantly, which demonstrates the robustness of the proposed methods to different environments. Some possible extensions of the presented work have also been discussed briefly as clues of future research.

REFERENCES

[1] R. G. Wiley
ELINT: The Interception and Analysis of Radar Signals. Norwood, MA, USA: Artech House, 2006.

[2] J. Liu, J. P. Lee, L. Li, Z.-Q. Luo, and K. M. Wong
Online clustering algorithms for radar emitter classification
IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 8, pp. 1185–1196, Aug. 2005.

[3] F. Digne, A. Baussard, A. Khenchaf, C. Cornu, and D. Jahan
Classification of radar pulses in a naval warfare context using bezier curve modeling of the instantaneous frequency law
IEEE Trans. Aerosp. Electron. Syst., vol. 53, no. 3, pp. 1469–1480, Jun. 2017.

[4] T. R. Kishore and K. D. Rao
Automatic intrapulse modulation classification of advanced LPI radar waveforms
IEEE Trans. Aerosp. Electron. Syst., vol. 53, no. 2, pp. 901–914, Apr. 2017.

[5] S. Sirianunpiboon, G. Noone, and S. D. Howard
Robust and recursive radar pulse train parameter estimators
In *Proc. 4th Int. Symp. Signal Process. Appl.*, Gold Coast, Qld, Australia, 1996, pp. 475–478.

[6] H. Mardia
New techniques for the deinterleaving of repetitive sequences
Proc. Inst. Electr. Eng. Radar, Sonar Navigat., vol. 136, no. 4, pp. 149–154, 1989.

[7] H. Arslan
Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems. Berlin, Germany: Springer, 2007.

[8] D. C. Robbins, R. K. Sarin, E. J. Horvitz, and E. B. Cutrell
Advanced navigation techniques for portable devices
U.S. Patent 7 327 349, Feb. 5, 2008.

[9] M. A. Richards, J. Scheer, W. A. Holm, and W. L. Melvin
Principles of Modern Radar. Stevenage, U.K.: Scitech Publishing, 2010.

[10] P. S. Ray
A novel pulse TOA analysis technique for radar identification
IEEE Trans. Aerosp. Electron. Syst., vol. 34, no. 3, pp. 716–721, Jul. 1998.

[11] R. J. Orsi, J. B. Moore, and R. E. Mahony
Spectrum estimation of interleaved pulse trains
IEEE Trans. Signal Process., vol. 47, no. 6, pp. 1646–1653, Jun. 1999.

[12] K. Nishiguchi and M. Kobayashi
Improved algorithm for estimating pulse repetition intervals
IEEE Trans. Aerosp. Electron. Syst., vol. 36, no. 2, pp. 407–421, Apr. 2000.

[13] D. Milojević and B. Popović
Improved algorithm for the deinterleaving of radar pulses
Proc. Inst. Electr. Eng. Radar, Sonar Navigat., vol. 139, no. 1, pp. 98–104, 1992.

[14] R. Sullivan
Radar Foundations for Imaging and Advanced Concepts. Herts., U.K.: IET, 2004.

[15] J. B. Moore and V. Krishnamurthy
Deinterleaving pulse trains using discrete-time stochastic dynamic-linear models
IEEE Trans. Signal Process., vol. 42, no. 11, pp. 3092–3103, Nov. 1994.

[16] T. Conroy and J. B. Moore
The limits of extended kalman filtering for pulse train deinterleaving
IEEE Trans. Signal Process., vol. 46, no. 12, pp. 3326–3332, Dec. 1998.

[17] A. Logothetis and V. Krishnamurthy
An interval-amplitude algorithm for deinterleaving stochastic pulse train sources
IEEE Trans. Signal Process., vol. 46, no. 5, pp. 1344–1350, May 1998.

[18] N. Visnevski, S. Haykin, V. Krishnamurthy, F. A. Dilkes, and P. Lavoie
Hidden Markov models for radar pulse train analysis in electronic warfare
In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process..*, vol. 5, 2005, pp. 597–600.

[19] J. Liu, H. Meng, Y. Liu, and X. Wang
Deinterleaving pulse trains in unconventional circumstances using multiple hypothesis tracking algorithm
Signal Process., vol. 90, no. 8, pp. 2581–2593, 2010.

[20] J. B. Pollack
On connectionist models of natural language processing
Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois, Peoria, IL, USA, 1987.

[21] J. A. Anderson, M. T. Gately, P. A. Penz, and D. R. Collins
Radar signal categorization using a neural network
Proc. IEEE, vol. 78, no. 10, pp. 1646–1657, Oct. 1990.

[22] E. Granger, Y. Savaria, P. Lavoie, and M.-A. Cantin
A comparison of self-organizing neural networks for fast clustering of radar pulses
Signal Process., vol. 64, no. 3, pp. 249–269, 1998.

[23] A. Ata'a and S. Abdullah
Deinterleaving of radar signals and PRF identification algorithms
IET Radar, Sonar Navigat., vol. 1, no. 5, pp. 340–347, 2007.

[24] N. Petrov, I. Jordanov, and J. Roe
Radar emitter signals recognition and classification with feedforward networks
Procedia Comput. Sci., vol. 22, pp. 1192–1200, 2013.

[25] Y. LeCun, Y. Bengio, and G. Hinton
Deep learning
Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[26] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur
Recurrent neural network based language model
In *Proc. Interspeech*, 2010, vol. 2, p. 3.

[27] D. Bahdanau, K. Cho, and Y. Bengio
Neural machine translation by jointly learning to align and translate
in *Proc. Int. Conf. Learn. Represent.,* San Diego, CA, May 7–9, 2015.

[28] T.-J. Hsieh, H.-F. Hsiao, and W.-C. Yeh
Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm
Appl. Soft Comput., vol. 11, no. 2, pp. 2510–2525, 2011.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean
Efficient estimation of word representations in vector space
in *Proc. Int. Conf. Learn. Represent.*, Scottsdale, Arizona, May 2–4, 2013.

[30] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor
Recommender Systems Handbook. Berlin, Germany: Springer, 2015.

[31] D. Silver *et al.*
Mastering the game of go with deep neural networks and tree search
Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[32] D. Silver *et al.*
Mastering the game of go without human knowledge
Nature, vol. 550, no. 7676, pp. 354–359, 2017.

[33] J. Schmidhuber
Deep learning in neural networks: An overview
Neural Netw., vol. 61, pp. 85–117, 2015.

[34] D. Tang, B. Qin, and T. Liu
Document modeling with gated recurrent neural network for sentiment classification
In *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1422–1432.

[35] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi
Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles
Proteins: Structure, Function Bioinformatics, vol. 47, no. 2, pp. 228–235, 2002.

[36] I. Goodfellow, Y. Bengio, and A. Courville
Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

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

[38] F. A. Gers, J. Schmidhuber, and F. Cummins
Learning to forget: Continual prediction with LSTM
Neural Comput., vol. 12, no. 10, pp. 2451–2471, 2000.

[39] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio
On the properties of neural machine translation: Encoder-decoder approaches
In *Proc. 8th Workshop Syntax, Semantics Structure Statist. Translation*, 2014, pp. 103–111.

[40] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio
Empirical evaluation of gated recurrent neural networks on sequence modeling
in *Proc. NIPS 2014 Workshop Deep Learn.*, Dec. 2014.

[41] D. Williams and G. Hinton
Learning representations by back-propagating errors
Nature, vol. 323, no. 6088, pp. 533–538, 1986.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams
Learning representations by back-propagating errors
in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds., Cambridge, MA, USA: MIT Press, pp. 696–699, 1988.

[43] B. A. Pearlmutter
Gradient calculations for dynamic recurrent neural networks: A survey
IEEE Trans. Neural Netw., vol. 6, no. 5, pp. 1212–1228, Sep. 1995.

[44] J. Martens and I. Sutskever
Learning recurrent neural networks with hessian-free optimization
In *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 1033–1040.

[45] N. Ketkar
Introduction to Pytorch
in *Deep Learning with Python.* Berkeley, CA, USA: Apress, 2017, pp. 195–208.

[46] M. Abadi *et al.*
Tensorflow: Large-scale machine learning on heterogeneous distributed systems
in *Proc. 12th USENIX Symp. Operat. Syst. Design Implement.* Savannah, GA, Georgia, Nov. 2–4, 2016.

[47] D. M. Powers
Evaluation: From precision, recall and f-measure to ROC, informedness, markedness and correlation
J. Mach. Learn. Technol., vol. 2, no. 1, pp. 37–63, 2011.

Zhang-Meng Liu received the Ph.D. degree in statistical signal processing from National University of Defense Technology (NUDT) of China, Changsha, China, in 2012.

He is currently an Associate Professor with NUDT, working in the interdiscipline of electronics engineering and computer science, especially electronic data mining and machine learning for information processing. He was a visiting scholar with the Computer Science Department, University of Illinois at Chicago, from April 2017 to March 2018, researching on data mining and deep learning in the group led by Philip S. Yu. He has authored or coauthored more than 30 journal papers in various areas of array signal processing, passive localization, and pulse data processing.

Philip S. Yu (F'93) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1972, the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1976 and 1978, respectively, and the M.B.A. degree from New York University, New York, NY, USA, in 1982.

He is currently a Disthinguished Professor with the Department of Computer Science, University of Illinois at Chicago (UIC), Chicago, IL, USA, and also holds the Wexler Chair in Information and Technology. Before joining UIC, he was with the Software Tools and Techniques Department, IBM Thomas J. Watson Research Center, where he was a Manager. He has authored or coauthored more than 970 papers in refereed journals and conferences with more than 74 500 citations and an H-index of 127. He holds or has applied for more than 300 U.S. patents. His main research interests include big data, data mining (especially on graph/network mining), social network, privacy preserving data publishing, data stream, database systems, and Internet applications and technologies.

Dr. Yu is currently a Fellow of the ACM. He is the recipient of ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of big data, the IEEE Computer Society's 2013 Technical Achievement Award for "pioneering and fundamentally innovative contributions to scalable indexing, querying, searching, mining, and anonymization of big data," and the Research Contributions Award from the IEEE International Conference on Data Mining in 2003 for his pioneering contributions to the field of data mining. He was also the recipient of the IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. He is the Editor-in-Chief for the *ACM Transactions on Knowledge Discovery from Data*. He is on the steering committee of ACM Conference on Information and Knowledge Management and was a steering committee member of the IEEE Conference on Data Mining and the IEEE Conference on Data Engineering.