IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 August 2023, accepted 18 August 2023, date of publication 22 August 2023, date of current version 30 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3307429

RESEARCH ARTICLE

CRSF: An Intrusion Detection Framework for Industrial Internet of Things Based on Pretrained CNN2D-RNN and SVM

SHIMING LI[1], GUANGZHAO CHAI[1], YUHE WANG[1], GUOHUI ZHOU[1], ZHENXING LI[1], DAN YU[1], AND RENCAI GAO[2]

[1]College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China
[2]College of Computer Science, Baicheng Normal University, Baicheng 137000, China

Corresponding authors: Yuhe Wang (cs2008wyh@163.com) and Rencai Gao (rencai_gao@163.com)

This work was supported in part by the Provincial Universities Basic Business Expense Scientific Research Projects of Heilongjiang Province under Grant 2021-KYYWF-0179, in part by the Science and Technology Project of Henan Province under Grant 212102310991, in part by the Key Scientific Research Project of Henan Province under Grant 21A413001, and in part by the Postgraduate Innovation Project of Harbin Normal University under Grant HSDSSCX2023-10.

ABSTRACT The traditional support vector machine (SVM) requires manual feature extraction to improve classification performance and relies on the expressive power of manually extracted features. However, this characteristic poses limitations in complex Industrial Internet of Things (IIoT) environments. Traditional manual feature extraction may fail to capture all relevant information, thereby restricting the application effectiveness of SVM in IIoT settings. CNN-RNN, as a deep learning network capable of simultaneously extracting spatial and temporal features, can alleviate researchers' burden. In this paper, we propose a novel intrusion detection system (IDS) framework based on anomalies, called CRSF. The framework's pre-training part employs a dimension transformation function to process input data into two-dimensional images. Two-dimensional convolutional kernels are then employed to extract spatial features, and the feature sequences are passed to an RNN to capture richer temporal features. After sufficient pre-training, SVM is used as a classifier to map the pre-training data from the feature space to a high-dimensional space and learn nonlinear decision boundaries, enabling the framework to accurately differentiate feature representations of different classes. Simulation experiments on the TON_IoT-Datasets demonstrate the effectiveness of the CRSF framework in intrusion detection. When using the ''linear'' kernel function in SVM, the framework achieves an accuracy, F1-score, and AUC of 0.9959, 0.9959, and 0.9977, respectively, indicating its capability and superiority in intrusion detection.

INDEX TERMS Industrial Internet of Things, convolutional neural network, recurrent neural network, support vector machine, intrusion detection.

I. INTRODUCTION

The Industrial Internet of Things (IIoT) [1] utilizes open Internet technologies to achieve integration, intelligent management, and interaction among industrial equipment, systems, and data. However, the high openness of the Internet often makes IIoT networks vulnerable to attacks, posing serious threats to the security and operation of IIoT systems.

The associate editor coordinating the review of this manuscript and approving it for publication was Jon Atli Benediktsson.

To enhance the security of IIoT systems, many traditional network security techniques have been applied, and the intrusion detection system (IDS) has gained attention as an important security technology in the field of IIoT. IDS is a network security tool used to monitor and identify potential intrusion behaviors in computer networks [2]. It helps protect computer networks from malicious activities and attacks. IDS can be divided into signature-based IDS (SIDS) and anomaly based IDS (AIDS), where SIDS can only identify known intrusions, while AIDS can detect zero-day attacks.

Machine learning techniques (ML) have been widely applied in the field of AIDS. IDS using ML methods [3], [4] can achieve fine-grained and high-precision automated detection. Support vector machine (SVM), as a typical machine learning algorithm, is commonly used for classification and regression tasks. SVMs have advantages such as effective handling of high-dimensional data, strong generalization ability, interpretability, and precise classification capability [5]. However, SVM requires manual feature extraction, expert knowledge, and extensive engineering effort. A practical solution is to utilize deep learning networks for automatic feature extraction [6], [7]. Deep learning networks significantly reduce the overall workload of feature extraction and can achieve excellent performance without requiring extensive expert knowledge. Convolutional neural networks (CNNs) [8], [9], as classical deep learning algorithms, can extract spatial features and compress feature dimensions through convolutional kernels and pooling operations. Recurrent neural networks (RNNs) [10], [11] have the ability to process time series data and extract temporal features. The combination of both (CNN-RNN) [12], [13] enables the extraction of spatial and temporal features simultaneously. The CNN part can employ parameter sharing [14] to reduce model parameters, improve computational efficiency, and accelerate feature extraction. Through an end-to-end training process, the CNN-RNN model can automatically learn abstract feature representations from the data, reducing the reliance on manual feature extraction.

To fully leverage the advantages of SVM in accurately finding decision boundaries and CNN-RNN in automatically extracting spatiotemporal features, this paper proposes a novel AIDS framework called CRSF. Its pre-training part uses a dimension transformation function to batchwise increase the dimensionality of intrusion samples. A CNN with two-dimensional convolutional kernels (CNN2D) is used to process the data and extract spatial features, which are then passed to RNN for extracting temporal features. After sufficient pre-training, the pre-training data are finally fed to SVM for training to achieve the classification objective. The benefits of dimensionality expansion and feature extraction using CNN2D lie in the process of transforming one-dimensional data into two-dimensional image data. In the two-dimensional image, each pixel can be regarded as a feature value, and each row of data represents different time steps or sampling points. By transforming the data into this higher-dimensional form, CNN2D can extract more spatial features and patterns from the image, thereby improving the accuracy of the model. The CRSF framework combines the advantages of SVM and the softmax classifier. The softmax classifier is adequately trained in the pre-training stage and can provide good initial classification results, while the SVM classifier can make more fine-grained classification decisions based on these features. By combining the results of the two classifiers, more accurate classification results can be obtained, enhancing the discriminability and classification accuracy of the overall framework between different categories.

The main contributions of this paper are as follows:

1. A novel network framework for industrial IoT intrusion detection is proposed, which includes a pre-training part that automatically extracts spatial and temporal features from input data, reducing the reliance on manual feature engineering for SVM.
2. The preprocessing part of the new framework utilizes the softmax function for initial classification, and after training, the initialized classification results are further trained by the SVM component. This design combines the classification results of softmax and SVM, leading to improved accuracy.
3. The preprocessing part of the new framework transforms input data into two-dimensional data to enable the extraction of more spatial features and patterns. Comparative experiments are conducted on the dimensions of the input data, and the results demonstrate that this approach further enhances the accuracy of the framework.
4. comparative experiments were conducted between the proposed framework and several individual models on the TON_IoT-Datasets. Furthermore, comparisons were made against state-of-the-art models, providing evidence of the advanced nature of the framework.

The remaining contents of this paper are as follows: Section II discusses the relevant background and techniques, Section III provides a detailed description of the structure and implementation process of CRSF, Section IV explains the model selection and parameters for the comparative experiments and analyses the experimental results, and Section V summarizes the role of the CRSF framework and the contributions of this paper and discusses our future plans.

II. RELATED WORK

A. IDS BASED ON MACHINE LEARNING

IDS based on machine learning is a technique that utilizes machine learning algorithms and models to detect abnormal activities and security threats in a network. Compared to traditional rule-based IDSs, machine learning-based IDSs can automatically learn and adapt to new threat patterns, offering better adaptability and generalization capability. SVM is a representative machine learning algorithm used for classification and regression tasks. SVM exhibits strong generalization ability and noise resistance. It finds an optimal hyperplane to differentiate samples of different classes and has achieved good results in various machine learning tasks. Almaiah M A proposed an intrusion detection system model based on the principal component analysis feature selection technique and different SVM kernel classifiers [16], emphasizing the role of SVM with the Gaussian kernel function in the field of intrusion detection. Yakub Kayode Saheed proposed the K-means-GASVM model [17], where they performed feature selection through k-means clustering on normalized data and used genetic algorithms for wrapper feature selection. Finally, SVM was employed for classification, improving the overall model's robustness and reducing computational costs. Agarap A F M introduced a structure that

combines a gated recurrent unit (GRU) and SVM for intrusion detection [18]. They performed simulation experiments using the ''Kyoto University honeypot systems' network traffic data'' dataset, and the experimental results showed that this hybrid structure outperformed the traditional GRU model in terms of accuracy and time.

B. IDS BASED ON DEEP LEARNING

After the emergence of deep learning, its ability to automatically learn features from large amounts of data [19], [20] has greatly reduced the difficulty of feature engineering. Deep learning models, such as neural networks, often have a large number of parameters and can express complex nonlinear relationships, making them suitable for intrusion detection in industrial IoT. For example, research by Vinayakumar et al. showed that classifiers generated by deep neural networks outperformed machine learning models in terms of accuracy and training time [21]. ZIL E. HUMA proposed a hybrid deep random neural network (DRaNN) by combining deep random neural networks with dropout regularization and multilayer perceptrons [22], demonstrating the advantages of deep neural network models in the field of intrusion detection in industrial IoT.

1) IDS BASED ON CNN

In the field of deep learning, the advantages of the CNN algorithm [23], [24] have been widely applied in the domain of IDS for industrial IoT. The convolution operation of CNN reduces the number of model parameters and alleviates the trade-off between high real-time requirements and low training efficiency. Liu and Zhang proposed a multiclass network intrusion detection model based on convolutional neural networks [25]. This model transformed one-dimensional raw data into two-dimensional data and utilized CNN for feature extraction, achieving good results through softmax classification. Li et al. and their colleagues conducted an initial performance evaluation of CNN prior to adversarial training, demonstrating the high performance of CNN as an IDS [26]. Song et al. treated collected data packets as image data and performed corresponding preprocessing. They then classified the processed results using CNN [27], demonstrating the feasibility of this approach. However, the aforementioned studies did not provide comparative experiments between data dimensionality before and after transformation. Although dimensionality expansion can improve the performance of the model in certain cases, its impact on specific tasks and datasets can vary. Therefore, conducting comparative experiments between nonexpanded and expanded data is a meaningful approach to accurately evaluate the effect of dimensionality expansion. Li et al. proposed a deep learning intrusion detection method based on the fusion of multiple convolutional neural networks (multi-CNN) [28]. The authors divided the data into four parts based on correlation and processed them into grayscale images, which were then passed through multiple CNNs for feature extraction. Finally, they demonstrated that this model outperformed traditional machine learning methods.

2) IDS BASED ON RNN

To further improve the performance of intrusion detection models, researchers have started exploring the application of recurrent neural networks (RNNs) in this field. RNN, as a type of neural network with sequential modeling capability, has been widely used in intrusion detection. Unlike CNN, RNN can handle sequential data and capture the temporal relationships of the data by remembering previous information. Therefore, RNN can be leveraged to enhance the detection ability of intrusion detection models, especially for intrusion behaviors with temporal characteristics. Ullah et al. proposed a novel deep learning model based on RNNs [29]. This model combines LSTM, BiLSTM, and GRU methods to achieve high classification accuracy. Feng et al. introduced a softmax classifier based on a stacked long short-term memory (LSTM) network for time series anomaly detection [30]. This structure is not only easy to train but also capable of predicting unseen attacks. However, it requires a large dataset to allow the model to train effectively, and its effectiveness is not significant when dealing with small-sample data. In fact, many researchers have applied RNN and its derivative models in the field of intrusion detection for industrial IoT [31], [32], [33]. These research findings indicate that RNN can effectively handle sequential data and capture temporal dependencies within the data.

3) IDS BASED ON CNN-RNN

However, using RNN or CNN alone may not fully exploit and utilize the temporal and spatial features in the data. To improve the accuracy of intrusion detection, researchers have combined CNN and RNN to build more powerful and accurate intrusion detection models. For example, Ashima Chawla proposed a structure that combines stacked one-dimensional CNNs with gated recurrent units (GRUs) [34]. This structure utilizes CNN layers to capture the local structural correlations in the sequence and improves performance through parallel execution. The RNN (GRU) layers are used to learn sequence dependencies from higher-level features. They conducted simulation experiments using the ADFA dataset to evaluate the performance of the proposed model. The experimental results demonstrated that this model outperforms the long short-term memory (LSTM) network in terms of training time reduction and classification performance. Vinayakumar et al. comprehensively analyzed the topology, network parameters, and network structure of architectures constructed with one-dimensional convolutional neural networks (CNN1D) and recurrent neural networks (RNN) and its variants [34]. They conducted simulation experiments on the KDDCup 99 dataset, and the experimental results demonstrated that architectures composed of CNN and RNN, along with their variants, exhibited superior performance compared to classical machine learning classifiers.

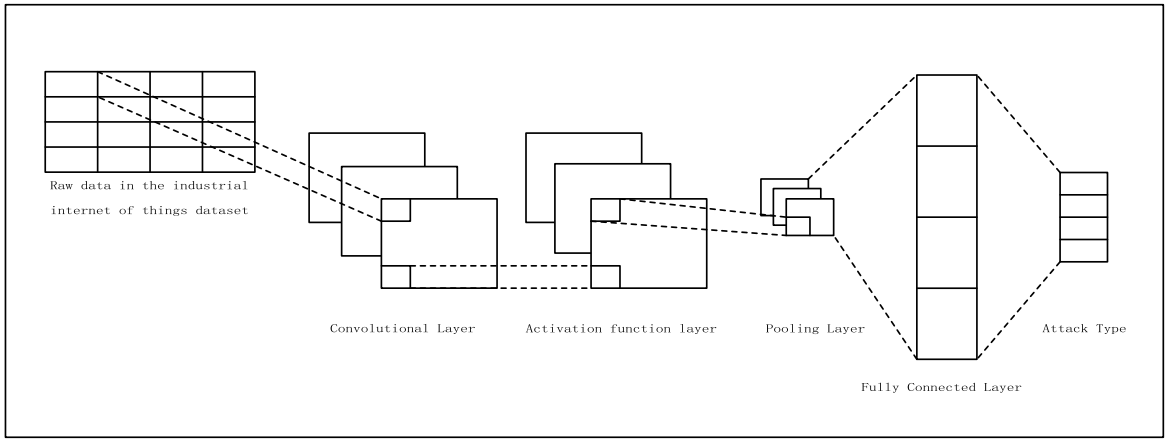

FIGURE 1. CNN architecture for intrusion detection.

C. RELATED TECHNOLOGIES

1) CNN TECHNOLOGY

CNN, a deep learning model [36], has achieved tremendous success in the field of intrusion detection due to its excellent feature extraction capability and parameter sharing mechanism. The architecture of a CNN consists of convolutional layers, pooling layers, activation functions, fully connected layers, and loss functions, as illustrated in Figure 1. The convolutional layers comprise multiple convolutional kernels, each responsible for extracting different features from the input data. Each kernel slides over the input data through convolutional operations and computes a series of feature maps. The pooling layers are utilized to reduce the spatial dimension of the feature maps and decrease the number of model parameters. The most commonly used pooling operation is max pooling, which selects the maximum value from the input region as the pooled result. Activation functions play a central role in mapping inputs to outputs in various neural network architectures. Nonlinear activation functions, such as the rectified linear unit (ReLU), are applied to the outputs of the convolutional layers to introduce nonlinear transformations and enhance the expressive power of the model.

Below are several commonly used activation functions in convolutional layers (formulas (1)∼(4)):

a: SIGMOID ACTIVATION FUNCTION

The sigmoid activation function maps the input to a range between 0 and 1 and can represent a probability distribution. It is often used in the output layer of binary classification problems. The mathematical expression of the sigmoid function is as follows:

$$f(x)_{sigm} = \frac{1}{1+e^{-x}} \tag{1}$$

b: RELU ACTIVATION FUNCTION

The ReLU activation function is a simple threshold function. It outputs 0 for input values less than or equal to 0, and it outputs the input value itself for input values greater than 0. The mathematical expression of ReLU is as follows:

$$f(x)_{ReLU} = max(0, x) \tag{2}$$

c: LEAKY RELU ACTIVATION FUNCTION

The Leaky ReLU activation function is a modified version of the traditional ReLU function that introduces a small nonzero slope for input values less than 0. Unlike the ReLU function, which outputs 0 for inputs less than or equal to 0, the Leaky ReLU retains a nonzero value for the negative part of the input, preventing complete compression to 0. The mathematical expression of Leaky ReLU is as follows:

$$f(x)_{\text{LeakyReLU}} = \begin{Bmatrix} x, x > 0 \\ mx, x \leq 0 \end{Bmatrix} \tag{3}$$

d: TANH ACTIVATION FUNCTION

The tanh activation function takes values ranging between -1 and 1; thus, its output is within the interval $[-1, 1]$. It is commonly used in neural networks for handling data with symmetric distributions. The mathematical expression of the tanh function is as follows:

$$f(x)_{Tanh} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

2) RNN TECHNOLOGY

RNN is a neural network model widely used in the processing of sequential data [37]. Compared to traditional feedforward neural networks, RNNs introduce recurrent connections that allow information to be transmitted and stored within the network, enabling the handling of data with temporal dependencies. The core idea of RNN is to incorporate a hidden state to store past information and combine it with the current input. This propagation of hidden states allows RNN to model elements within a sequence and possess the ability to retain memory and context when processing sequential data. The basic structure of RNN consists of an input layer, a hidden

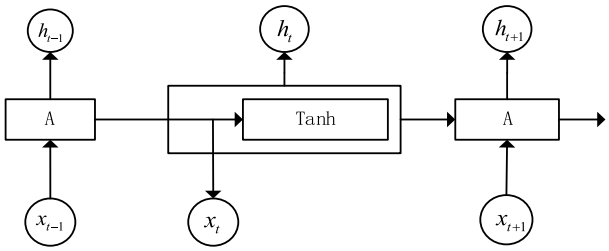

FIGURE 2. RNN architecture.

layer, and an output layer, as shown in Figure 2, where x_t represents the input at time step t, and h_t represents the output of the hidden layer at time step t. RNN can extract temporal features from the data, making it suitable for combining with CNN to jointly extract spatiotemporal features from the input data.

3) SVM TECHNOLOGY

SVM is a machine learning algorithm [38] primarily used for binary and multiclass classification problems. The objective of SVM is to find an optimal hyperplane or decision boundary that separates samples from different classes. During operation, SVM maps the input data to a high-dimensional feature space using a kernel function to better separate data points of different classes. This mapping allows the original data to be transformed from a lower dimension to a higher dimension, introducing nonlinear features for better sample discrimination. Subsequently, SVM utilizes the hinge loss function to optimize model parameters and find an optimal hyperplane in the feature space. This hyperplane effectively separates samples from different classes and maximizes the distance of the closest samples to the hyperplane, enhancing classification robustness and generalization. These closest samples to the hyperplane are referred to as support vectors. Figure 3 illustrates the working principle of SVM.

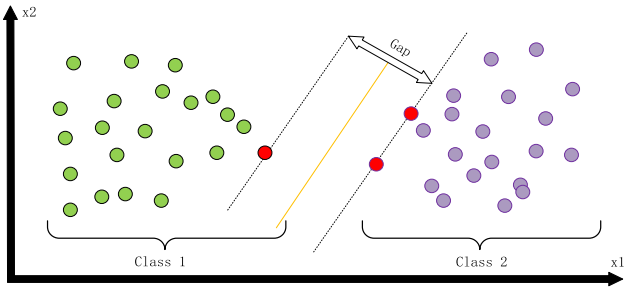

FIGURE 3. Working principle of SVM.

In SVM, the kernel function is a technique used to handle nonlinearly separable problems [39]. The kernel function maps the input data from a low-dimensional feature space to a high-dimensional feature space, making the data linearly separable in the high-dimensional space. This enables SVM to handle nonlinear classification problems. Table 1 presents several commonly used kernel functions:

TABLE 1. SVM kernel function and its formula.

No.	Kernel	Mathematical Functions
1	Linear	$K(x,z) = < x,z >$
2	Polynomial	$K(x,z) = (\gamma < x,z > +r)^d$
3	RBF	$K(x,z) = \exp(-\gamma \parallel x-z \parallel^2)$
4	Sigmoid	$K(x,z) = \tanh(\gamma < x,z > +r)$

III. FRAMEWORK STRUCTURE

In IoT systems, various devices, sensors, and controllers communicate and exchange data through networks. The communication between these devices generates network traffic, including the transmission of data packets, interaction of communication protocols, and communication patterns among devices. By analyzing and monitoring these network traffic data, it is possible to understand the communication behavior between devices, identify anomalous activities, and perform intrusion detection for security and performance-related applications. The network traffic data in industrial IoT environments may encompass various types of communication, such as the upload of sensor data, the issuance of control commands, and collaborative communication among devices. These data are crucial for monitoring and optimizing the operation of industrial IoT systems. Therefore, collecting, analyzing, and processing network traffic data in industrial IoT environments constitutes a crucial step toward achieving intelligent security and management. The proposed CRSF framework is divided into two main parts: the pretraining part and the decision-making part. The pre-training part utilizes CNN2D-RNN to capture the spatiotemporal features of network traffic data in industrial IoT environments and performs classification using softmax. Adequate pretraining is conducted in the pre-training part, resulting in an initial classification outcome. The decision-making part employs SVM to accept the initial classification outcome and undergoes final training, yielding the ultimate decision content. The details of the framework architecture are further explained, and a comprehensive inference process along with the meaning of the parameters used is presented in Table 2. The overall structure of the CRSF is illustrated in Figure 4.

A. INFERENCE OF THE PRE-TRAINING PART IN CRSF

In the industrial Internet of Things, if there is one network traffic event that has m features, the shape of this event after dimension-up (add a channel to the dimension of the feature vector converted from the traffic event) can be represented as $Shape_{event} = (1, 1, m, 1)$. The CNN part of the CRSF framework's pre-training part consists of c convolutional blocks, where the activation function for each convolutional layer is selected as the ReLU function. The input shape of the first convolutional layer is (1, m, 1). The shape of the

TABLE 2. The parameters used in the CRSF reasoning process and their corresponding meanings.

No.	Parameters	Meaning
1	$Shape_{event}$	The shape of a traffic event after the dimension-up operation
2	$Shape_{filter}$	The shape of the convolution kernel in the convolution layer
3	$Shape_{pooling}$	The shape of the pooling layer
4	$cstep_s$	The moving step of the s-th convolution kernel
5	$pstep_r$	The moving step size of the r-th pooling kernel
6	x_i	The i-th feature in the feature vector represented by a piece of network traffic
7	X_{New}	The eigenvector corresponding to the network traffic after the convolution operation
8	X_{New_i}	The process value obtained by convolution operation of the initial feature vector of the i-th step
9	X_{final}	Feature vectors after CNN operations in the pre-training part
10	X_{final_i}	The process value obtained by pooling the eigenvector of the i-th step
11	$(X_{final})_t$	The input vector at time t of the RNN in the pre-training stage
12	Num_s	The feature dimension of the feature vector after the convolution operation (including the padding operation) is equal to the value before the convolution operation.
13	Num_r	The feature dimension of the feature vector after the pooling operation
14	q	The number of fully connected layers in the pre-training part
15	p	The number of dropout layers in the pre-training part
16	N_s	The length of the convolution kernel, where s represents the convolution kernel in the s-th convolution layer
17	N_r	The length of the pooling kernel, where r represents the pooling kernel in the rth pooling layer
18	b	vector of bias values for the convolution operation
19	c	The number of convolutional blocks in the pre-training part of the CRSF framework
20	h_t	The output of the RNN in the pre-training part at time t
21	H	Feature vectors after the RNN operation in the pre-training part
22	A	Weight parameters of the RNN model
23	L	The output of the fully connected layer in the pre-training part
24	B	The weight parameters of the fully connected layer
25	b_q	The bias vector of the fully connected layer
26	w_l	The weight parameters of the decision-making part of the SVM
27	b_l	The bias vector for the decision-making part of the SVM

convolutional kernels in the subsequent convolutional layers is defined as $Shape_{filter}= (1,N_s)$ *where* $s = 1,2,3,\cdots,c$, and the convolutional kernels move with a step size of $cstep_s cstep_s$. The latter half of the pre-training part consists of an RNN layer, q fully connected layers, and p dropout layers.

The output of the RNN layer serves as the input for the fully connected layers. In the fully connected layers,except for the last fully connected layer that uses the softmax function, all other layers use the ReLU activation function.

By taking the feature vectors $X = (x_1,x_2,x_3,x_4,\ldots\ldots,x_m)$ of samples as the input to the convolutional layer, the CRSF framework first pads the input data to a certain dimension and then maps it to a new vector X_{New} using convolutional kernels The process of changing the eigenvector from X to X_{New} is shown in formulas (5)∼(7).

$$X_{New\_i} = ReLU(w_s x_{i:i+N_s}^T + b) \tag{5}$$

Here, b represents the bias value vector, and X_{New_i} corresponds to the intermediate values obtained by convolving the initial feature vector at each step.

$$Num_s = \left\lceil \frac{m - N_s + 1}{cstep_s} \right\rceil \tag{6}$$

$$X_{New} = [X_{New\_1}, X_{New\_2}, X_{New\_3}, \ldots, X_{New\_Num_s}] \tag{7}$$

Since the padding operation is performed in advance, the overall dimensionality of the feature vector X_{New} remains unchanged, and it is applied as the output of the convolutional layer to the max pooling layer. Assuming the shape of the pooling layer is given by $Shape_{pooling} = (1, N_r)$ *where* $r = 1, 2, 3, \cdots, c$ and the stride for each movement is $pstep_r$, the MaxPooling2D operation maps X_{New} to X_{final}. The process of changing the eigenvector from X_{New} to X_{final} is shown in formulas (8)∼(10).

$$X_{final\_i} = Max(X_{New\_i}, X_{New\_i+1}, \ldots, X_{New\_i+N_r-1}, \ldots, X_{New\_i+N_r}) \tag{8}$$

$$Num_r = \left\lceil \frac{m - N_r + 1}{pstep_r} \right\rceil \tag{9}$$

$$X_{final} = [X_{final\_1}, X_{final\_2}, \ldots, X_{final\_Num_r}] \tag{10}$$

A mapping function f_{CNN} is established to represent the transformation of the input vector through a convolutional block. formula (11) summarizes the process of formulas (8)∼(10) as f_{CNN} mapping and expresses it.

$$X_{final} = f_{CNN}(X) \tag{11}$$

Formula (12) expresses the output of the t-th time step in the RNN of the CRSF pre-training part. The output X_{final} of the CNN layer is used as the input of the RNN layer. h_t represents the output of one time step in the RNN layer, and its value is given by:

$$h_t = tanh(A \times concat(h_{t-1}, (X_{final})_t)) \tag{12}$$

where h_{t-1} is the output vector obtained from the previous time step; $(X_{final})_t$ is the input vector at the current time step; *concat* represents the concatenation of the two vectors; and A denotes the weight parameters, which are the trainable parameters in the recurrent neural network. In formula (13), we define H as the output result of the RNN, which is given by:

$$H = RNN(X_{final}, A) \tag{13}$$

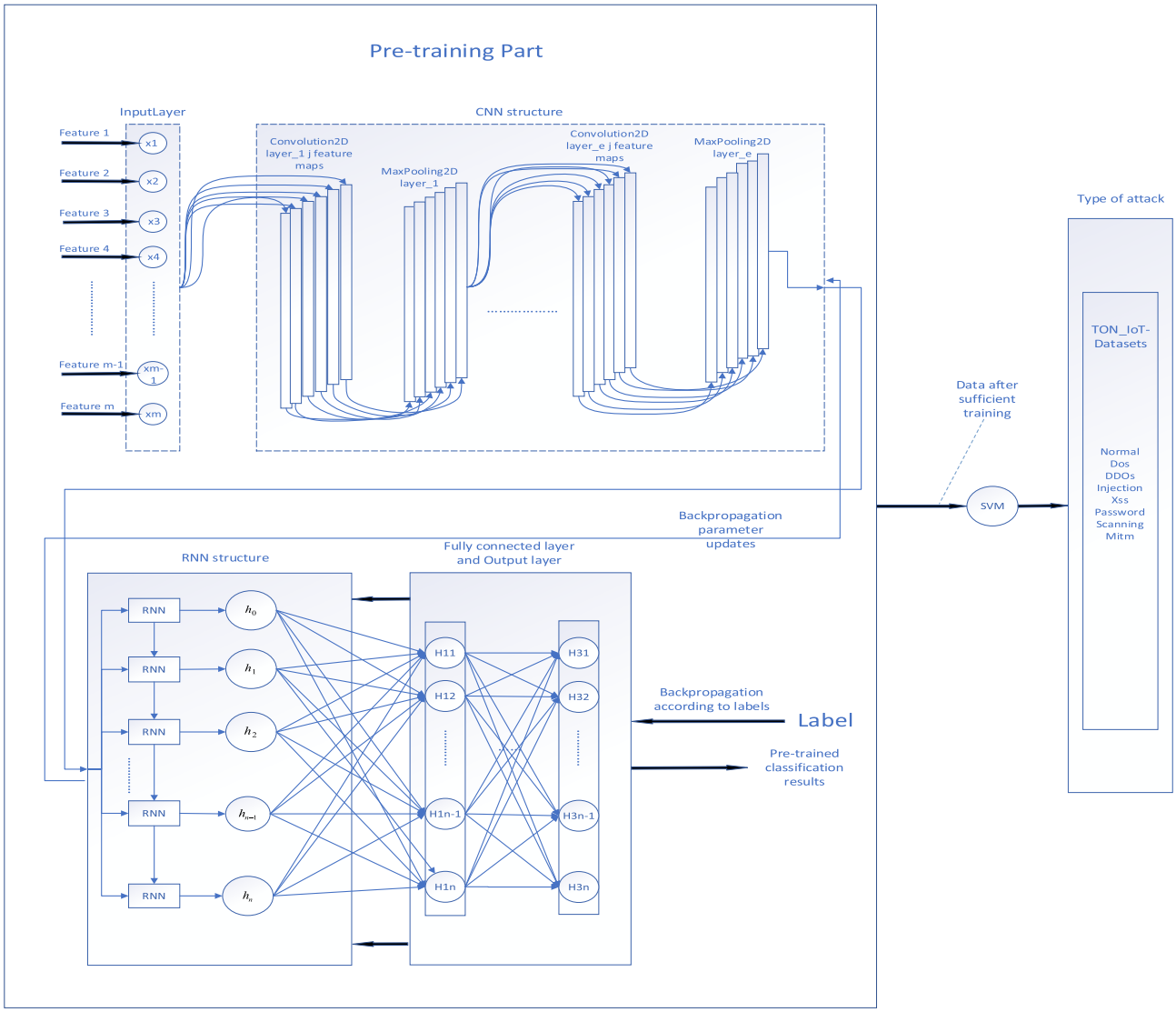

FIGURE 4. Structure of the CRSF.

Formula (14) represents the process of the H passing through the fully connected layer in the CRSF pre-training part. Taking H as the input of the fully connected layer, we obtain the output result L of the fully connected layer, which is given by:

$$L = R\text{elu}\left(BH + b_q\right) \tag{14}$$

where B represents the weight parameters that need to be trained in the fully connected layer, and b_q represents the bias value vector of the fully connected layer.

In the last layer of the fully connected layer, the pre-training part uses a softmax classifier to perform initial classification on the obtained data. The cross-entropy loss function is used to calculate the loss value of the pre-training part. Based on the loss value, backpropagation is performed to update the weight parameters of each layer.

B. INFERENCE OF THE DECISION-MAKING PART IN CRSF

After sufficient training of the pre-training part, the output of the layer before the last layer (which uses the softmax function) is used as input for the SVM. The SVM is trained to perform classification and output the results. After training, the decision function $F(P)$ can generate a score vector for each class (where the output of the well-trained pre-training part is defined as P), denoted as $F(P) = sign(w_l P + b_l)$. Formula (15) shows the process of using the argmax function to get the final category. To obtain the index of the highest score from the score vector, the argmax function is introduced. The predicted result can be represented as follows:

$$pred\_Class = argmax(F(P)) \tag{15}$$

IV. EXPERIMENT AND ANALYSIS

In this paper, we compare and analyze the proposed CRSF framework with typical deep learning models, including

CNN, RNN, CNN-RNN, and RNN-SVM. We also compare it with a set of baseline models [15] and current state-of-the-art models. The CNN mentioned above is divided into CNN1D and CNN2D to demonstrate the feasibility of selecting the dimension of CNN convolutional kernels and processing the input data dimensions. The selection of these models for comparison with CRSF aims to demonstrate the superior performance of the CRSF framework over single models in intrusion detection classification. We believe that the pre-training part of the CRSF framework effectively extracts spatiotemporal features from the data and leverages the classification advantages of both softmax and SVM, thereby better meeting the high accuracy requirements of the industrial IoT. The experimental results include the ACC, F1-Score, and AUC metrics for each model on the TON_IoT-Datasets. The data preprocessing is introduced in Section IV-A, the specifications and environment of the machines used for model training are described in Section IV-B, and the specific model parameters and training parameters are presented in Section IV-C. The experimental results and discussions are provided in Section IV-D.

A. DATA PREPROCESSING

TON_IoT-Datasets [40], [41], [42], [43], [44], [45], [46] is a new generation of industrial 4.0 datasets primarily used for evaluating the accuracy and fidelity of intrusion detection in industrial IoT using machine learning and deep learning models.

During the preprocessing process, rows with missing values were first removed. The dataset was then split, with 80% of each class's instances assigned to the training set and the remaining 20% to the test set for dataset partitioning. The ''type'' column was extracted from both the training and test sets and used for training and testing, respectively. The labels (The categories of the labels consist of 8 classes, where one class represents the non-attack type ''Normal,'' and the remaining seven classes represent different types of attacks, namely: ''DDos,'' ''Dos,'' ''Injection,'' ''Xss,'' ''Password,'' ''Scanning,'' and ''Mitm.'') in the training and test sets were one-hot encoded, and columns with all-zero data were eliminated to reduce the feature dimensionality, such as ''Processor_pct_C3_Time,'' ''Processor_pct_C2_Time,'' ''Processor_C2_Transitions_sec,'' and others. Additionally, standardization was applied to the training and test sets to eliminate feature differences arising from varying scales.

B. MACHINE SPECIFICATIONS AND ENVIRONMENT

The machine used for the experiments was equipped with an Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz (12 CPUs), ~2.2 GHz processor, 16,256 MB of RAM, and Windows 10 operating system. The hardware configuration also included an NVIDIA GeForce GTX 1050 Ti GPU. The experiments were conducted using the keras framework version 2.9.0, built upon TensorFlow version 2.9.1.

C. MODEL SPECIFIC PARAMETERS AND TRAINING PARAMETERS

The section introduces the model structure settings for CRSF (pre-training part), CNN2D-RNN, CNN1D-RNN, RNN-SVM (pre-training part), CNN2D, CNN1D, and RNN used in the experiments. In theory, we believe that CRSF has the ability to extract temporal features from network traffic while also capturing additional spatial features through dimensionality expansion. Moreover, CRSF combines the classification results of softmax and SVM, and thus, its optimal state is expected to outperform the aforementioned models.

1) SPECIFIC PARAMETERS OF THE PREPROCESSING PART OF THE CRSF FRAMEWORK

The first convolutional block's Convolution2D layer consists of 32 convolutional filters, each with a shape of (1, 5) and a stride value of 1. The ''padding'' parameter is set to ''same'' to ensure that the feature vectors do not undergo dimension reduction after passing through the convolutional layer. The activation function for the layer is chosen as the ''ReLU'' function. The MaxPooling2D layer in all convolutional blocks has a pool shape of (1, 2) and a stride value of 2. The second convolutional block's Convolution2D layer consists of 64 convolutional filters, each with a shape of (1, 4). The third convolutional block's Convolution2D layer consists of 128 convolutional filters, each with a shape of (1, 3). The fourth convolutional block's Convolution2D layer consists of 256 convolutional filters, each with a shape of (1, 2). All other parameters remain the same as those in the Convolution2D layer of the first convolutional block. For the RNN part, the ''units'' parameter value is set to the number of features in a single sample of the current dataset.

2) SPECIFIC PARAMETERS OF THE CNN2D-RNN MODEL

The CNN2D-RNN model is the same as the pre-training part of the CRSF framework.

3) SPECIFIC PARAMETERS OF THE CNN1D-RNN MODEL

The model's CNN part consists of 4 convolutional blocks, with each block containing a Convolution1D layer and a MaxPooling1D layer. All MaxPooling1D layers share the same parameters: a pool shape of 2 and a stride of 2. The first Convolution1D layer has 32 filters, a filter shape of 5, a stride of 1, and the ''same'' padding parameter and uses the ReLU activation function. In the remaining three convolutional blocks, the number of filters is doubled sequentially, while the filter shape decreases by one each time. All other parameters remain the same as in the first convolutional block. The RNN part and the fully connected layer of the model are the same as in the CNN2D-RNN model and are not described in detail. Due to the one-dimensional nature of the convolutional layers and max pooling layers in the CNN part of this model, the output data from the CNN part are two-dimensional. Therefore, there is no need for a

TABLE 3. The parameter configuration of all experimental models.

No.	Parameters corresponding to different layers	CRSF	CNN2D-RNN	CNN1D-RNN	RNN-SVM	CNN2D	CNN1D	RNN
1	C_n of Conv2D (No.1,No.2,No.3,No.4)	32,64, 128,256	32,64, 128,256	-	-	32,64, 128,256	-	-
2	C_m of Conv2D (No.1,No.2,No.3,No.4)	(1,5),(1,4), (1,3),(1,2)	(1,5),(1,4), (1,3),(1,2)	-	-	(1,5),(1,4), (1,3),(1,2)	-	-
3	C_s of Conv2D (No.1,No.2,No.3,No.4)	1	1	-	-	1	-	-
4	C_p of Conv2D (No.1,No.2,No.3,No.4)	"same"	"same"	-	-	"same"	-	-
5	C_n of Conv1D (No.1,No.2,No.3,No.4)	-	-	32,64, 128,256	-	-	32,64, 128,256	-
6	C_m of Conv1D (No.1,No.2,No.3,No.4)	-	-	(1,5),(1,4), (1,3),(1,2)	-	-	(1,5),(1,4), (1,3),(1,2)	-
7	C_s of Conv1D (No.1,No.2,No.3,No.4)	-	-	1	-	-	1	-
8	C_p of Conv1D (No.1,No.2,No.3,No.4)	-	-	"same"	-	-	"same"	-
9	C_m of MaxPooling2D (No.1,No.2,No.3,No.4)	(1,2)	(1,2)		-	(1,2)		-
10	C_s of MaxPooling2D (No.1,No.2,No.3,No.4)	2	2		-	2		-
11	C_m of MaxPooling1D (No.1,No.2,No.3,No.4)	-	-	(1,2)	-	-	(1,2)	-
12	C_s of MaxPooling1D (No.1,No.2,No.3,No.4)	-	-	2	-	-	2	-
13	Units of RNN	104	104	104	104	-	-	104
14	Whether it has reshape operation	Yes	Yes	No	No	Yes	No	No
15	The dropout ratio of the dropout layer between the fully connected layers	0.5	0.5	0.5	0.5	0.5	0.5	0.5
16	Activation functions for fully connected layers (except the last layer) and convolutional layers	"relu"	"relu"	"relu"	"relu"	"relu"	"relu"	"relu"
17	Units of Fully connected layer(No.1,No.2,No.3,No.4)	500,200, 50,8	500,200, 50,8	500,200, 50,8	500,200, 50,8	500,200, 50,8	500,200, 50,8	500,200, 50,8

reshape operation between the CNN and RNN parts to handle dimensionality reduction.

4) SPECIFIC PARAMETERS OF THE RNN-SVM FRAMEWORK

The RNN part of the model utilizes an RNN layer along with a dropout layer, where the dropout rate is set to 0.5. A fully connected layer is employed to receive the input from the RNN layer. The fully connected layer consists of three layers, and the parameters are set the same as the fully connected layers in the CRSF preprocessing part. Similarly, after ample pre-training, the data from RNN part t were passed to an SVM model for training.

5) SPECIFIC PARAMETERS OF THE CNN2D MODEL

The model consists of a CNN2D part and a fully connected layer part. The CNN2D part and the fully connected layer part are the same as the CNN2D and fully connected layer parts of the CNN2D-RNN model. The difference is that there is no RNN part in this model. However, since a two-dimensional convolutional kernel is used, the output data dimension of the CNN part is three-dimensional, while the fully connected layer requires one-dimensional data. Therefore, a flattened layer is added between the CNN and the fully connected layer to reduce the dimensionality of the output data from the CNN.

6) SPECIFIC PARAMETERS OF THE CNN1D MODEL

The model consists of a CNN1D part and a fully connected layer. The parameters of both parts are the same as the corresponding parts in the CNN1D-RNN model.

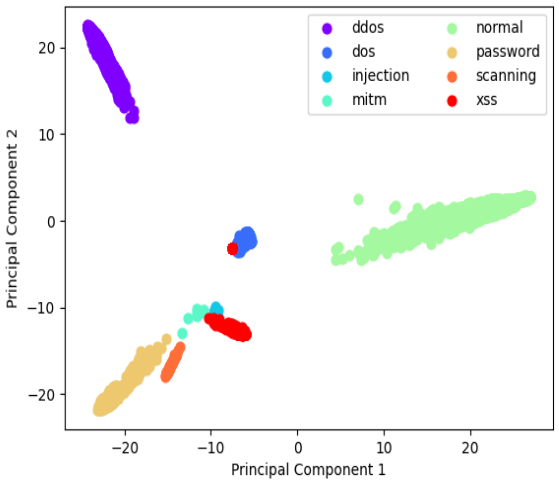

FIGURE 5. The distribution of the data output from the preprocessing part of the CRSF after pre-training.

7) SPECIFIC PARAMETERS OF THE RNN MODEL

The model consists of an RNN part and a fully connected layer part, with both parts having the same parameters as their corresponding parts in the CNN2D-RNN model.

The parameter configuration of all experimental models is shown in Table 3. In the table, C_n represents the number of cores, C_m represents the shape and size of the core, C_s represents the moving step of the core, and C_p represents the filling method. The number of CNN2D layers of all models (if there is such a layer) is 4, which is reflected in the form of No. 1, No. 2, No. 3, and No. 4 in the table, and the

TABLE 4. Training parameters of different models.

No.	Model	Batch Size	optimizer	Epochs	Learning Rate	SVM C	SVM degree
1	CRSF	64	Adam	50	0.0001	1	3
2	RNN-SVM	64	Adam	50	0.0001	1	3
3	CNN2D-RNN	64	Adam	50	0.0001	NULL	NULL
4	CNN1D-RNN	64	Adam	50	0.0001	NULL	NULL
5	RNN	64	Adam	50	0.0001	NULL	NULL
6	CNN2D	64	Adam	50	0.0001	NULL	NULL
7	CNN1D	64	Adam	50	0.0001	NULL	NULL

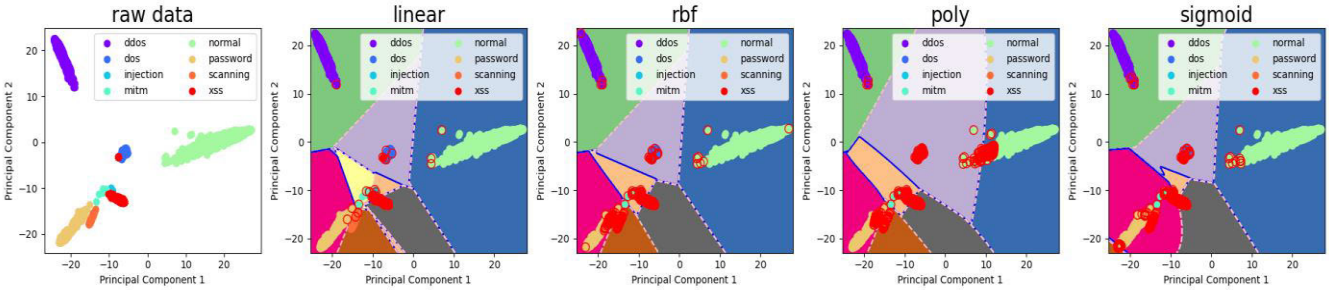

FIGURE 6. The classification effect of using different kernel functions in the SVM part of CRSF(The figure shows the division of "category regions" and the number and distribution of "support vectors" after CRSF uses four kernel functions to find the decision boundary).

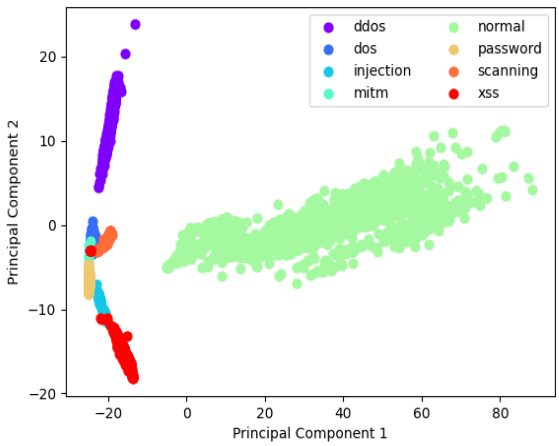

FIGURE 7. The distribution of the data output from the preprocessing part of the RNN-SVM framework after pre-training.

CNN1D layer (if there is such a layer) and fully connected layer are also similar to this set up. In addition, both CRSF and RNN-SVM in Table 3 show the structural parameters of the pre-training part.

The training parameters for each model are shown in Table 4.

D. EXPERIMENTAL RESULTS AND DISCUSSION

The experimental results of each model using the TON_IoT-Datasets dataset are shown in Table 5. The output obtained from the pre-training part of CRSF is shown in Figure 5.

TABLE 5. Performance of different models.

No.	Model	kernel	ACC	F1-score	AUC
1	CRSF	linear	0.9959	0.9959	0.9977
		rbf	0.9943	0.9944	0.9967
		poly	0.9885	0.9885	0.9934
		sigmoid	0.9938	0.9938	0.9964
2	RNN-SVM	linear	0.9909	0.991	0.9948
		rbf	0.9902	0.9902	0.9944
		poly	0.9462	0.9462	0.9692
		sigmoid	0.9490	0.9502	0.9709
3	CNN2D-RNN	NULL	0.9914	0.9914	0.9882
4	CNN1D-RNN	NULL	0.9581	0.9581	0.9698
5	CNN2D	NULL	0.9575	0.9575	0.9587
6	CNN1D	NULL	0.7328	0.7328	0.5429
7	RNN	NULL	0.9407	0.9409	0.8861

The classification results of SVM using different kernel functions are illustrated in Figure 6.

When utilizing the linear kernel function, the SVM component of CRSF achieves the best classification performance with an accuracy of 0.9959, outperforming other kernel functions. This can be attributed to the observation from Figure 6

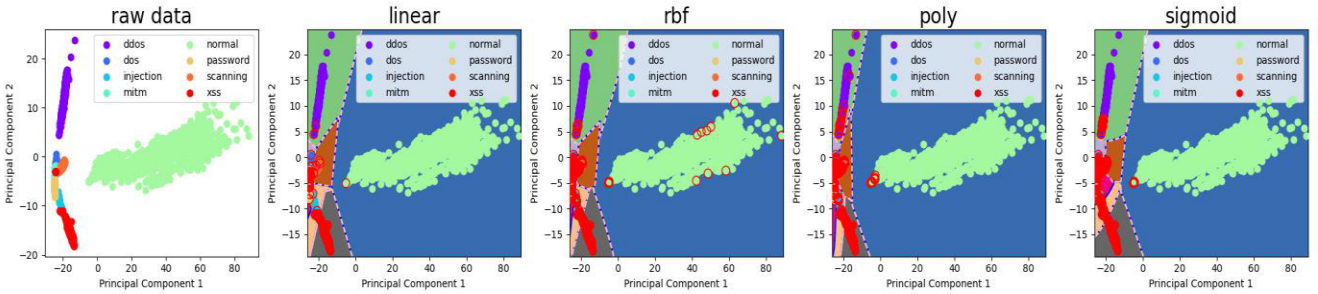

FIGURE 8. The classification effect of using different kernel functions in the SVM part of RNN-SVM(The figure shows the division of "category regions" and the number and distribution of "support vectors" after RNN-SVM uses four kernel functions to find the decision boundary).

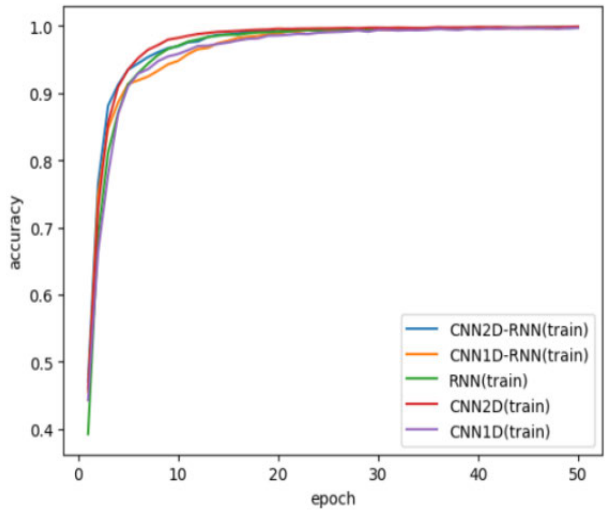

FIGURE 9. The training accuracy of CNN2D-RNN, CNN1D-RNN, RNN, CNN2D, and CNN1D changes with the epoch.

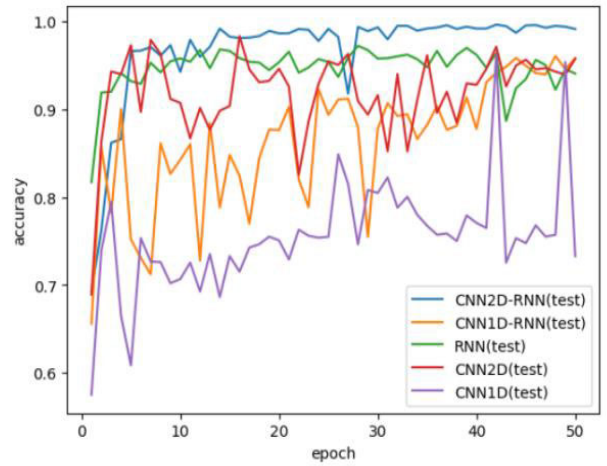

FIGURE 10. The training accuracy of CNN2D-RNN, CNN1D-RNN, RNN, CNN2D, and CNN1D changes with the epoch.

that the data after the pre-training part are already in a linearly separable state, and the linear kernel function is most effective in handling such data. The distribution of output vectors from the pre-training part in the RNN-SVM model is displayed in Figure 7. The classification results of the SVM component with different kernel functions in this model are shown in Figure 8.

TABLE 6. Performance comparison of some baseline models and the method proposed in this paper.

Model	ACC	F1-score	AUC
LR	0.61	0.46	-
LDA	0.68	0.62	-
KNN	0.84	0.84	-
RF	0.85	0.85	-
CART	0.88	0.88	-
NB	0.66	0.51	-
SVM	0.61	0.46	-
LSTM	0.81	0.80	-
Our framework	**0.99**	**0.99**	**0.99**

Experimental results demonstrate that similar to CRSF, the RNN-SVM model with sufficient pre-training provides promising initial classification results. Regardless of the kernel function chosen, the RNN-SVM framework achieves higher accuracy compared to the RNN model without SVM, consistent with the findings in reference [18].

Furthermore, Figure 9 presents the training accuracy variations with epochs for the CNN2D-RNN, CNN1D-RNN, RNN, CNN2D, and CNN1D models. It is evident that the training curves have converged after 40 epochs. Figure 10 depicts the test accuracy variations with epochs for these models, where the CNN2D-RNN model consistently outperforms the other two models within the first 50 epochs.

The experimental results indicate that the CNN2D-RNN model outperforms the CNN1D-RNN, CNN2D, CNN1D, and RNN models in terms of performance. This not only highlights the superiority of the CNN2D-RNN model as the preprocessing component of the CRSF framework but also demonstrates that the CNN2D model can extract more spatial features and patterns from the input data after dimensionality augmentation, thereby improving the model's accuracy. Additionally, the performance of the CRSF framework is superior to that of the CNN2D-RNN model. The comparison between CRSF and CNN2D-RNN, as well as between RNN-SVM and RNN, suggests that combining softmax and

TABLE 7. Performance comparison of state-of-the-art methods in the literature and the method proposed in this paper.

Model	ACC	F1-score	AUC
Adaptive Voting [47]	0.9890	0.9890	-
SATIDS (with MRMR-FS) [48]	0.9750	0.9810	-
Weighted KNN Model (with RELIEF) [49]	0.9822	0.9870	0.9900
Fine Gaussian SVM Model (with RELIEF) [49]	0.9797	0.9850	0.9900
DFF (with NetFlow) [50,51]	0.9474	0.9600	0.9843
Extra trees (with NetFlow) [52]	0.9805	0.9800	-
DFF with CHI (with NetFlow) [53]	0.8561	0.9100	0.8519
RF with COR (with NetFlow) [53]	0.9938	1	0.9946
Our framework	**0.9959**	**0.9959**	**0.9977**

SVM (using ''linear'' or ''rbf'' kernel) leads to significantly better classification performance than using the softmax classifier alone. Finally, by comparing with selected benchmark models [15] and state-of-the-art models, the effectiveness and superiority of the CRSF framework on the TON-IoT dataset are verified. The comparison results are presented in Table 6 and Table 7.

V. CONCLUSION AND FUTURE WORKS

In this paper, we propose a novel IDS framework called CRSF by combining the strengths of the CNN-RNN model and the SVM model and addressing the limitations of manual feature extraction required by a single SVM model. The preprocessing part of this framework converts the input data into two-dimensional image data for better spatial feature and pattern extraction by CNN2D. It then utilizes RNN, which has advantages in processing temporal data for intrusion detection, to capture the temporal characteristics of the input data. After sufficient pre-training, the pretrained data are fed into the SVM for training, and the SVM determines the optimal decision boundary and provides classification results. Through comparative experiments, this paper demonstrates that the CRSF framework outperforms single models and other combined models, achieving superior performance in intrusion detection tasks in the industrial IoT domain. The paper also proves that converting the input data into two-dimensional image data for processing further improves the accuracy of the model. Furthermore, the experimental results are compared with state-of-the-art models, showing that the proposed model exhibits higher accuracy in detecting intrusion attacks. These research findings offer a new approach and method for intrusion detection in the industrial IoT domain, providing valuable exploration and reference for the development and application of this field. They hold practical value in enhancing the accuracy and reliability of intrusion detection. In fact, in terms of computational resources, CRSF requires more than some benchmark models. It achieves high accuracy but is not as lightweight. Such characteristics make CRSF slightly less efficient in resource-constrained devices and environments, limiting its scalability in practical deployments. In the future, we have the following ideas:

1. For the CRSF pre-training stage, using CNN2D-RNN for feature extraction undoubtedly increases its demand for computational resources. An example is the lightweight framework proposed by Jan et al., which does not use deep learning networks such as CNN and RNN for feature extraction. Instead, it utilizes manually obtained feature pools and training data for SVM [54]. We can explore optimization algorithms to partially optimize the feature selection part of CRSF (inspired by the lightweight MFO algorithm by Kalita et al. [55]) and appropriately reduce the number of layers in the CNN component of the CRSF pre-training section while maintaining accuracy, making CRSF sufficiently lightweight.
2. We decided to implement parallel computing in CRSF through parallelization of the pre-training phase and asynchronous parameter updates, enhancing its scalability.

REFERENCES

[1] S. Tharewal, M. W. Ashfaque, S. S. Banu, P. Uma, S. M. Hassen, and M. Shabaz, ''Intrusion detection system for industrial Internet of Things based on deep reinforcement learning,'' *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–8, Mar. 2022.
[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, ''Survey of intrusion detection systems: Techniques, datasets and challenges,'' *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019.
[3] T. S. Kala and A. Christy, ''HFFPNN classifier: A hybrid approach for intrusion detection based OPSO and hybridization of feed forward neural network (FFNN) and probabilistic neural network (PNN),'' *Multimedia Tools Appl.*, vol. 80, no. 4, pp. 6457–6478, Feb. 2021.
[4] M. A. Rahman, A. T. Asyhari, O. W. Wen, H. Ajra, Y. Ahmed, and F. Anwar, ''Effective combining of feature selection techniques for machine learning-enabled IoT intrusion detection,'' *Multimedia Tools Appl.*, vol. 80, pp. 1–19, Mar. 2021.
[5] S. Memon, ''Host-based intrusion detection system using support vector machine,'' *Quaid-e-Awam Univ. Res. J. Eng., Sci. Technol.*, vol. 20, no. 1, pp. 44–54, Jun. 2022.
[6] Y. LeCun, Y. Bengio, and G. Hinton, ''Deep learning,'' *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[7] A. Shrestha and A. Mahmood, ''Review of deep learning algorithms and architectures,'' *IEEE Access*, vol. 7, pp. 53040–53065, 2019.
[8] D.-X. Zhou, ''Theory of deep convolutional neural networks: Downsampling,'' *Neural Netw.*, vol. 124, pp. 319–327, Apr. 2020.
[9] G. Li, M. Zhang, J. Li, F. Lv, and G. Tong, ''Efficient densely connected convolutional neural networks,'' *Pattern Recognit.*, vol. 109, Jan. 2021, Art. no. 107610.
[10] H. Hewamalage, C. Bergmeir, and K. Bandara, ''Recurrent neural networks for time series forecasting: Current status and future directions,'' *Int. J. Forecasting*, vol. 37, no. 1, pp. 388–427, Jan. 2021.
[11] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, ''LEARN codes: Inventing low-latency codes via recurrent neural networks,'' *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 207–216, May 2020.
[12] Z. Wu and Z. Long, ''Research on network traffic classification method based on CNN–RNN,'' in *3D Imaging—Multidimensional Signal Processing and Deep Learning: Multidimensional Signals, Video Processing and Applications*, vol. 2. Singapore: Springer, 2023, pp. 249–257.
[13] H. C. Altunay and Z. Albayrak, ''A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks,'' *Eng. Sci. Technol., Int. J.*, vol. 38, Feb. 2023, Art. no. 101322.
[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[15] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020.

[16] M. A. Almaiah, O. Almomani, A. Alsaaidah, S. Al-Otaibi, N. Bani-Hani, A. K. A. Hwaitat, A. Al-Zahrani, A. Lutfi, A. B. Awad, and T. H. H. Aldhyani, "Performance investigation of principal component analysis for intrusion detection system using different support vector machine kernels," *Electronics*, vol. 11, no. 21, p. 3571, Nov. 2022.

[17] Y. K. Saheed, M. O. Arowolo, and U. Tosho, "An efficient hybridization of K-means and genetic algorithm based on support vector machine for cyber intrusion detection system," *Int. J. Electr. Eng. Informat.*, vol. 14, no. 2, pp. 426–442, Jun. 2022.

[18] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," in *Proc. 10th Int. Conf. Mach. Learn. Comput.*, Feb. 2018, pp. 26–30.

[19] A. R. Khan, M. Kashif, R. H. Jhaveri, R. Raut, T. Saba, and S. A. Bahaj, "Deep learning for intrusion detection and security of Internet of Things (IoT): Current analysis, challenges, and possible solutions," *Secur. Commun. Netw.*, vol. 2022, pp. 1–13, Jul. 2022.

[20] Q. A. Al-Haija and A. Al Badawi, "High-performance intrusion detection system for networked UAVs via deep learning," *Neural Comput. Appl.*, vol. 34, no. 13, pp. 10885–10900, Jul. 2022.

[21] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[22] Z. E. Huma, S. Latif, J. Ahmad, Z. Idrees, A. Ibrar, Z. Zou, F. Alqahtani, and F. Baothman, "A hybrid deep random neural network for cyberattack detection in the industrial Internet of Things," *IEEE Access*, vol. 9, pp. 55595–55605, 2021.

[23] G. Yao, T. Lei, and J. Zhong, "A review of convolutional-neural-network-based action recognition," *Pattern Recognit. Lett.*, vol. 118, pp. 14–22, Feb. 2019.

[24] A. Dhillon and G. K. Verma, "Convolutional neural network: A review of models, methodologies and applications to object detection," *Prog. Artif. Intell.*, vol. 9, no. 2, pp. 85–112, Jun. 2020.

[25] G. Liu and J. Zhang, "CNID: Research of network intrusion detection based on convolutional neural network," *Discrete Dyn. Nature Soc.*, vol. 2020, pp. 1–11, May 2020.

[26] S. Li, J. Wang, Y. Wang, G. Zhou, and Y. Zhao, "EIFDAA: Evaluation of an IDS with function-discarding adversarial attacks in the IIoT," *Heliyon*, vol. 9, no. 2, Feb. 2023, Art. no. e13520.

[27] J. Y. Song, R. Paul, J. H. Yun, H. C. Kim, and Y. J. Choi, "CNN-based anomaly detection for packet payloads of industrial control system," *Int. J. Sensor Netw.*, vol. 36, no. 1, pp. 36–49, 2021.

[28] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," *Measurement*, vol. 154, Mar. 2020, Art. no. 107450.

[29] I. Ullah and Q. H. Mahmoud, "Design and development of RNN anomaly detection model for IoT networks," *IEEE Access*, vol. 10, pp. 62722–62750, 2022.

[30] C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2017, pp. 261–272.

[31] W. Hao, T. Yang, and Q. Yang, "Hybrid statistical-machine learning for real-time anomaly detection in industrial cyber–physical systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 1, pp. 32–46, Jan. 2023.

[32] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterline, "Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks," *Information*, vol. 11, no. 5, p. 243, May 2020.

[33] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5244–5253, Aug. 2020.

[34] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2018, pp. 149–158.

[35] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.

[36] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Mar. 2021.

[37] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Phys. D, Nonlinear Phenomena*, vol. 404, Mar. 2020, Art. no. 132306.

[38] C. Zhang, D. Jia, L. Wang, W. Wang, F. Liu, and A. Yang, "Comparative research on network intrusion detection methods based on machine learning," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102861.

[39] M. Mohammadi, T. A. Rashid, S. H. T. Karim, A. H. M. Aldalwie, Q. T. Tho, M. Bidaki, A. M. Rahmani, and M. Hosseinzadeh, "A comprehensive survey and taxonomy of the SVM-based intrusion detection systems," *J. Netw. Comput. Appl.*, vol. 178, Mar. 2021, Art. no. 102983.

[40] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustain. Cities Soc.*, vol. 72, Sep. 2021, Art. no. 102994.

[41] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. D. Hartog, "ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 485–496, Jan. 2022.

[42] N. Moustafa, M. Keshky, E. Debiez, and H. Janicke, "Federated TON_IoT windows datasets for evaluating AI-based security applications," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2020, pp. 848–855.

[43] N. Moustafa, M. Ahmed, and S. Ahmed, "Data analytics-enabled intrusion detection: Evaluations of ToN_IoT Linux datasets," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2020, pp. 727–735.

[44] N. Moustafa, "New generations of Internet of Things datasets for cybersecurity applications based machine learning: TON_IoT datasets," in *Proc. eResearch Australasia Conf.*, Brisbane, QLD, Australia, 2019, pp. 21–25.

[45] N. Moustafa, "A systemic IoT-fog-cloud architecture for big-data analytics and cyber security systems: A review of fog computing," *Secure Edge Comput.*, pp. 41–50, 2021.

[46] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, "IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities," *Sustain. Cities Soc.*, vol. 72, Sep. 2021, Art. no. 103041.

[47] Z. Yang, Z. Liu, X. Zong, and G. Wang, "An optimized adaptive ensemble model with feature selection for network intrusion detection," *Concurrency Comput., Pract. Exper.*, vol. 35, no. 4, p. e7529, Feb. 2023.

[48] R. Elsayed, R. Hamada, M. Hammoudeh, M. Abdalla, and S. A. Elsaid, "A hierarchical deep learning-based intrusion detection architecture for clustered Internet of Things," *J. Sensor Actuator Netw.*, vol. 12, no. 1, p. 3, Dec. 2022.

[49] R. H. Mohamed, F. A. Mosa, and R. A. Sadek, "Efficient intrusion detection system for IoT environment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 4, 2022.

[50] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection," *Big Data Res.*, vol. 30, Nov. 2022, Art. no. 100359.

[51] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowl. Inf. Syst.*, vol. 41, no. 3, pp. 647–665, Dec. 2014.

[52] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," 2021, *arXiv:2101.11315*.

[53] M. Sarhan, S. Layeghy, and M. Portmann, "Feature analysis for machine learning-based IoT intrusion detection," 2021, *arXiv:2108.12732*.

[54] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the Internet of Things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019.

[55] D. J. Kalita, V. P. Singh, and V. Kumar, "A novel adaptive optimization framework for SVM hyper-parameters tuning in non-stationary environment: A case study on intrusion detection system," *Expert Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119189.

SHIMING LI received the M.D. degree from the Harbin Institute of Technology. He is currently an Associate Professor with the College of Computer Science and Information Engineering, Harbin Normal University (HRBNU), China Computer Society (CCF 37474M). His main research interests include network and information security, industrial internet, and the Internet of Things.

GUANGZHAO CHAI was born in China, in 2001. He is currently pursuing the master's degree with Harbin Normal University. His research interest includes network and information security.

YUHE WANG received the B.Eng. and Ph.D. degrees from the Harbin University of Science and Technology, Harbin, Heilongjiang, China, in 2012 and 2019, respectively. He was a Lecturer with the Changchun University of Technology. He is currently a Lecturer with Harbin Normal University, Harbin. He has published approximately five articles. His research interests include intelligent computing, industrial network security, and belief rule base.

GUOHUI ZHOU received the Ph.D. degree from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun, China. He has published more than 20 journal articles. His current research interests include artificial intelligence, pattern recognition, and embedded systems. He was a recipient of the Second Prize of the Scientific and Technological Progress of Heilongjiang Province.

ZHENXING LI received the M.S.E. degree from Northeastern University, Shenyang, Liaoning, China. He is currently a Senior Engineer with Harbin Normal University, Harbin. He has published four articles. His research interest includes network and information security.

DAN YU received the M.S. degree from Harbin Normal University, Harbin, Heilongjiang, China. She is currently a Lecturer with Harbin Normal University. She has published four articles. Her research interest includes information security.

RENCAI GAO received the M.D. degree from Jilin University. He is currently an Associate Professor with the College of Computer Science, Baicheng Normal University. His main research interest includes network security.