Deep Learning-Based Intrusion Detection for Distributed Denial of Service Attack in Agriculture 4.0

Mohamed Amine Ferrag 1, Lei Shu 2,3,*, Hamouda Djallel 1 and Kim-Kwang Raymond Choo 4

1 Department of Computer Science, Guelma University, Guelma 24000, Algeria; ferrag.mohamedamine@univ-guelma.dz (M.A.F.); hamouda.djallel@univ-guelma.dz (H.D.)
2 College of Artificial Intelligence, Nanjing Agricultural University, Nanjing 210031, China
3 School of Engineering, University of Lincoln, Lincoln LN6 7TS, UK
4 Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA; raymond.choo@fulbrightmail.org

* Correspondence: lei.shu@ieee.org

Abstract: Smart Agriculture or Agricultural Internet of things, consists of integrating advanced technologies (e.g., NFV, SDN, 5G/6G, Blockchain, IoT, Fog, Edge, and AI) into existing farm operations to improve the quality and productivity of agricultural products. The convergence of Industry 4.0 and Intelligent Agriculture provides new opportunities for migration from factory agriculture to the future generation, known as Agriculture 4.0. However, since the deployment of thousands of IoT based devices is in an open field, there are many new threats in Agriculture 4.0. Security researchers are involved in this topic to ensure the safety of the system since an adversary can initiate many cyber attacks, such as DDoS attacks to making a service unavailable and then injecting false data to tell us that the agricultural equipment is safe but in reality, it has been theft. In this paper, we propose a deep learning-based intrusion detection system for DDoS attacks based on three models, namely, convolutional neural networks, deep neural networks, and recurrent neural networks. Each model’s performance is studied within two classification types (binary and multiclass) using two new real traffic datasets, namely, CIC-DDoS2019 dataset and TON_IoT dataset, which contain different types of DDoS attacks.

Keywords: deep learning approaches; intrusion detection system; Agriculture 4.0; DDoS attack; smart agriculture

1. Introduction

The 4th revolution of the industrial era (or Industry 4.0) is the new industry trend that defines the Smart Factory concept [1]. This concept is based on emerging technologies such as Fog computing, Cloud computing, Artificial Intelligence, Deep learning... etc. To provide an optimization of operations and reduction of costs, these technologies are employed to establish a connection between machines and the Internet, through the Internet-of-Things, to collect information in the Cloud and Edge and then process them using artificial intelligence algorithms. Industry 4.0 is expected to transform the agricultural industry and advance the 4th agricultural revolution, known as Agriculture 4.0. The first three industrial revolutions deeply reshaped the agricultural industries from indigenous agriculture (Agriculture 1.0) towards mechanized agriculture (Agriculture 2.0) and recent precision farming (Agriculture 3.0), as presented in Figure 1 [2,3].

In the most recent years, the IoT application has been deployed for Agriculture 4.0 using wireless sensor networks such as, Supply chain management, Smart monitoring, Smart water, Agrochemicals applications, Disease management, and Smart harvesting. Figure 2 presents the IoT, IIoT, Industry 4.0, Agriculture 4.0 and the common concepts. Industry 4.0 focuses primarily on the manufacturing sector, Agriculture 4.0 focuses on the Agriculture sector, whereas IIoT covers all sectors where industrial/professional equipment
is used. However, with thousands of IoT-based devices deployed in the open field, there are many new cyber security threats in Agriculture 4.0. When an adversary attempting to penetrate Agriculture 4.0 network, it use several different approaches such as DDoS attacks, scanning attacks, false data injection attacks, to disrupt the functioning of the IoT-based devices. For example, in the soil pH parameters, if the pH rises excessively, it means that the farmer will increase ammonium input, and if the pH falls, it indicates that the farmer will reduce ammonium input. With this information, an adversary can launch DDoS attacks to disrupt the pH parameters. Hence, this private information (i.e., the pH parameters) must be protected from cyber attacks [4]. To protect Agriculture 4.0 from destruction, change, unauthorized access, or attack, Security researchers propose the use of an intrusion detection system (IDS) beside the authentication, access control, and integrity techniques [5–7].

![Figure 1. Agricultural revolutions with industrial revolutions and related cyber security threats.](image)

The IDS system is a mechanism monitoring the network traffic, which is used to detect suspicious or abnormal activities and then enables preventive measure on the intrusion risks. Therefore, intrusion detection systems can be divided into two major types, namely, (1) Network Intrusion Detection Systems (NIDS) and (2) Host Intrusion Detection Systems (HIDS). The NIDS is typically deployed or located at critical network points to ensure that it covers the locations where the traffic is more susceptible to attacked, while the HIDS systems works on any device on the network that has Internet access. To detecting intrusion, there are two main techniques, namely, IDS (1) based on anomalies and (2) IDS based on signatures [8]. The signature-based IDS (i.e., Misuse Detection or Knowledge-based Detection) concentrates on identifying a “signature”, patterns of intrusion event, and it is as efficient as updating the database at a specific moment of time. Based on monitoring regular activities, the anomaly-based IDS (i.e., Behavior-based Detection) uses machine learning techniques to compare trustworthy behavioral patterns with new behaviors. When an administrator receives an alert via the IDS system, it uses Intrusion Prevention Systems (IPS) to block the threat such as Trojan horse, DDoS attacks, etc. [9].
This paper focuses on developing and employing deep-learning approaches for detecting cyber threats (i.e., anomaly-based IDS). There are some recently proposed IDS systems that employ deep learning strategies for IoT applications, such as wireless networks [10], big data environments [11], industrial cyber–physical systems [12], SCADA systems [13], smart grids [14], internet of vehicles [15], and cloud computing [16]. Deep learning approaches are also used in Agriculture 4.0, for crop hail damage, soil and vegetation/crop mapping, crop monitoring, irrigation, greenhouse monitoring, etc. [17]. However, there are eight big challenges in the field of intrusion detection systems for Agriculture 4.0: (1) Data collection that contains IIoT traffics with cyber attacks, (2) Less amount of training data, (3) Non-representative training data, (4) Poor quality of data, (5) Irrelevant/unwanted features, (6) Overfitting the training data, (7) Underfitting the training data, (8) Offline learning & deployment of the model [18]. Our proposed model overcomes these challenges. The datasets used in our paper are very popular, recent, and used by the scientific community for developing intrusion detection systems for IIoT networks.

Our contributions in this work are:

- We propose three deep learning-based IDS models, including a convolutional neural network-based IDS model, a deep neural network-based IDS model, and a recurrent neural network-based IDS model.
- We provide a performance evaluation and comparative analysis of machine learning and deep learning approaches for cyber security in agriculture 4.0.
- We review three models of deep learning; namely, convolutional neural networks, deep neural networks, and recurrent neural networks. Each model’s performance is studied within two classification types (binary and multiclass) using two new real traffic datasets, namely, CIC-DDoS2019 dataset and TON_IoT dataset.
- We focus on the following important performance indicators: false alarm rate (FAR), precision, F-score, detection rate (DR), recall, True Negative Rate (TNR), False Accept Rate (FAR), ROC Curve, and accuracy.

The rest of this work is structured as follows. Section 2 review the related works. Section 3 presents the implementation of IDSs. Section 4 provides a comparative study on Deep Learning based-IDS for Agriculture 4.0. Lastly, Section 5 presents conclusions.

2. Related Work

The popularity of deep learning in different fields of Big Data has created a lot of attention in the area of cyber security. According to its architectural conception, deep learning can be categorized in various types, such as generative and discriminative [19]. The intrusion detection systems have been investigated with the use of shallow and deep
networks to identify anomalous patterns in both network and host-based systems. The summary of deep learning approaches for network intrusion detection for the IoT networks is presented in Table 1.

Table 1. Summary of deep learning approaches for network intrusion detection for the IoT networks.

| System                        | Year   | Network Model | Deep Learning Techniques | The Basic Idea                                                                 | Dataset Used                        | Performance Metrics                  |
|-------------------------------|--------|---------------|--------------------------|--------------------------------------------------------------------------------|------------------------------------|--------------------------------------|
| Diro and Chilamkurti [20]     | 2018   | Social internet of things | Deep learning approach with softmax as activation function | Deploy the distributed attack detection system at the fog computing layer         | NSL-KDD, ISCX, and KDDCUP99       | Accuracy, detection rate, and false alarm rate |
| Muna et al. [21]              | 2018   | Industrial internet of things | Unsupervised deep auto-encoder algorithm | The unsupervised deep auto-encoder algorithm is used to learn normal network behaviors, while a standard supervised deep neural network model is used to classify network behaviors | NSL-KDD and UNSW-NB15              | Accuracy, detection rate, and false positive rate |
| HaddadPajouh et al. [22]      | 2018   | Internet of things | Deep Recurrent Neural Network | Detecting IoT malware based on three stages, namely, collection data, feature extraction, and deep threat classifier | IoT malware dataset                  | Accuracy, detection rate             |
| Vinayakumar et al. [23]       | 2020   | The Internet of Things networks of smart cities | Cost-sensitive model-based deep learning, | Uses a two-tier environment for monitoring DNS logs                                | AmritaDGA                          | F1-score, true positive rate, False positive rate, precision, accuracy, recall |
| Parra et al. [24]             | 2020   | Internet of things | CNN and LSTM | The CNN is used in an IoT micro-security add-on, while the LSTM is used by the back-end server | N-BaIoT dataset                     | F1-score, True Positive Rate, True Negative Rate, precision, Accuracy, recall |
| Latif et al. [25]             | 2020   | Industrial internet of things | Lightweight random neural network, | Uses a model with 1 input layer, 8 hidden layers, and 1 output layer | DS20S dataset                       | Accuracy, precision, recall, and F1 score |
| Manimurugan et al. [26]       | 2020   | Internet of Medical Things | Deep belief network technique | Uses the greedy layer-wise scheme to optimize the deep learning structure | CICIDS 2017 dataset                  | Accuracy, detection rate, precision, recall, F-measure |
| Konnionts et al. [27]         | 2020   | Internet of things | Deep Neural Network | Detecting IoT attacks based on three stages, namely, extracting data, adapt parameters of deep learning, and identify the anomalous incidents | Bot-iot and UNSW-NB15 datasets      | Recall, F-measure, accuracy, precision |
| Zhou et al. [28]              | 2020   | Industry 4.0 | Variational long short-term memory (VLSTM) learning model | Detecting IoT attacks based on an encoder–decoder neural network | UNSW-NB15 dataset                   | Accuracy, False alarm rate, F1, Area under curve |
| NG and Selvakumar [29]        | 2020   | Fog computing-enable Internet of things | Convolutional deep learning technique | The computations are performed in the fog nodes | UNSW’s dataset Bot-iot dataset | Accuracy, precision, recall, F-measure |
| Khoa et al. [30]              | 2020   | IoT industry 4.0 | Deep neural networks | Uses smart “filters” deployed at the IoT gateways for detecting network attacks | - KDD, NSL-KDD, and UNSW - N-BaIoT dataset | Accuracy, Recall, Precision and F-measure |
| Ferrag and Lean-dros [31]     | 2020   | Smart Grids | Recurrent neural networks | Employs recurrent neural networks with blockchain for detecting network attacks | - Bot-IoT dataset - CIDCID2017 dataset - Power system dataset | - False alarm rate, detection rate, accuracy |
| Popoola et al. [31]           | 2020   | Internet of Things | Deep bidirectional long short-term memory | User deep bidirectional long short-term memory to identify the traffic of botnet attacks from benign traffic in IoT networks | Bot-iot dataset | Matthews Correlation Coefficient |
| Al-Hawawreh et al. [32]       | 2020   | Internet of Things | Deep learning techniques | Uses a deep pattern extractor to identify the attack types of malicious patterns | - TON-IoT dataset - N-BaIoT dataset | Accuracy, DR, FPR, FNR, MCC |
| Ge et al. [33]                | 2021   | Internet of Things | Customised deep learning technique | Uses the concepts of deep learning and transfer learning for cyber security in IoT networks | Bot-IoT dataset | Accuracy, Recall, Precision, and F1 score |
| Our Work                      | /      | Agriculture 4.0 | Convolutional neural network, Deep neural network, and Recurrent neural network | Study the performance of three deep learning models to identify the traffic of DDoS attacks from benign traffic in Agriculture 4.0 | - CIC-DDoS2019 dataset - CIC-DXO2019 dataset [34] - TON_IoT dataset [35] | Detection rate (DR), false alarm rate (FAR), precision, F-score, recall, True Accept Rate (TAR), ROC Curve, and accuracy |

Diro and Chilamkurti [20] designed a distributed attack detection system based on deep learning for the IoT networks. The authors proposed to deploy this system at the fog computing layer for hosting attack detection systems and training models. Three cyber security datasets are used in the performance evaluation, including, NSL-KDD, ISCX, and KDDCUP99, in which the results show a precision of 71%, 98.56%, and 97%, for R2L, U2R attacks, Probe attacks, and DoS attacks, respectively. Muna et al. [21] an anomaly detection system, named ADS, for detecting cyber attacks in the industrial internet of things. The ADS system uses deep learning techniques, in which the unsupervised deep auto-encoder algorithm is used to train network normal patterns of behavior and generate the correct settings. Both NSL-KDD and UNSW-NB15 are used in the evaluation of performance.
which the results of the experiments demonstrate a detection ratio of 99% and a false positive ratio of 1.8%.

Based on scanning the DNS services in smart city applications of IoT, Vinayakumar et al. [23] presented an instruction detection mechanism against botnet attacks. Specifically, the proposed mechanism uses a two-tier environment for monitoring DNS logs and searching the domain name generated by the domain generation algorithm through deep learning algorithms to minimize false alarm rates. Latif et al. [25] designed an intrusion detection mechanism that uses a lightweight random neural network for detecting cyber threats in the industrial internet of things. Compared to traditional machine learning techniques such as SVM, ANN, and decision tree, the proposed system shows good performance on an open-source dataset called DS2OS. The DS2OS dataset contains seven types of attacks, including DoS attacks, Scan, Data type probing, Malicious control, Wrong setup, Spying, and Malicious operation. These attacks are not sufficient to prove the performance of an intrusion detection mechanism for identifying cyber threats in the industrial IoTs.

To identify an adversary who tries to insert useless data and detecting phishing and Botnet attack, Parra et al. [24] proposed a distributed architecture using two deep learning approaches, namely, a Distributed CNN scheme and LSTM network. The DCNN is used in an IoT micro-security add-on, while the LSTM is used by the back-end server. The N-BaIoT dataset is employed in the evaluation of performance, where outcomes show an accuracy of 98% and 94.30% during the training phase and the testing phase, respectively. To detecting IoT malware, Haddad Pajouh et al. [22] designed an intrusion detection mechanism using a recurrent neural network. The proposed mechanism employs three stages; namely, collection data, feature extrication, and deep threat classifier. The results of the experiments demonstrate the highest accuracy of 98.18% under the 10-fold cross-validation analysis and efficient compared to conventional machine learning classifiers such as Naive Bayes, K-Nearest Neighbor, Random Forest, and Decision Tree. Koroniotis et al. [27] proposed a network forensics scheme, called PDF, for detecting and monitoring the attack patterns in IoT based networks. The PDF scheme is based on three stages, namely, (1) extracting data, (2) adapt parameters of deep learning, and (3) identify anomalous incidents. The particle swarm optimization algorithm is used in the second stage, whereas the deep neural model is used in the third stage. The experimental results reveal an accuracy of 99.9% compared to 93.2% with decision tree and 72.7% with naïve bayes.

NG and Selvakumar [29] proposed an anomaly detection framework based on vector convolutional deep learning technique. The authors proposed also that the computations are processed at the fog nodes. The experiments conducted on the UNSW Bot-IoT dataset show an accuracy of 99.71%, 99.80%, 99.92%, 77.22%, for DDoS attacks, DoS attacks, Reconnaissance attacks, and Theft attacks, respectively. Therefore, to detect cyber attacks in the Internet-of-Medical-Things (IMoT), Manimurugan et al. [26] introduced an intrusion detection mechanism using the deep belief network technique. The proposed mechanism is evaluated using the CICIDS 2017 dataset which shows an accuracy of 97.71% and 96.37% for PortScan attack and infiltration attack, respectively. Popoola et al. [31] developed a hybrid intrusion detection mechanism, called LAE-BLSTM, for the detection of botnets in IoT networks. The LAE-BLSTM mechanism uses deep Bidirectional Long Short-Term Memory (BLSTM) and Long Short-Term Memory Autoencoder (LAE). The LAE is used for the dimensionality reduction of the feature, while the BLSTM is used to identify the traffic of botnet attacks from benign traffic in IoT networks. The Bot-IoT dataset used in the evaluation of performance, which demonstrates that the LAE-BLSTM mechanism reached a data size reduction ratio of 91.89%.

3. IDS Implementation

In this section, we propose three deep learning-based IDS models for detection of cyber attacks in Agriculture 4.0, including recurrent neural network-based IDS model, convolutional neural network-based IDS model, and deep neural network-based IDS model.
3.1. Network Model

The considered network model of Agriculture 4.0 is presented in Figure 3, which is based on three layers, namely, (1) Agricultural sensors layer, (2) Fog computing layer, and (3) Cloud computing layer. The agricultural sensors layer consists of various IoT devices and drones applied to monitor agricultural environment data. Actuators are activated in the agricultural sensors layer when the data meet specific conditions. New energy technology and smart grid architecture are placed in the agricultural sensors layer for supplying energy for IoT devices. In each fog node, a deep learning based-intrusion detection system is placed. The IoT data are transmitted directly to the fog computing layer from the agricultural sensors layer for analysis and machine learning algorithms, while Cloud computing nodes provide the storage and end-to-end services. The computations of deep learning based-intrusion detection systems are performed in the fog nodes. We consider that there is a group of attackers that launch DDoS attacks in order to affect the functioning of the network, which can affect food safety, agri-food supply chain efficiency, and agricultural productivity.

![Figure 3. The proposed deep learning-based IDS for DDoS attack in Agriculture 4.0.](image)

3.2. Rnn-Based Ids

Recurrent neural networks, or RNN, are a category of neural networks for processing sequential data (i.e., a sequence of values $X^{(1)}, \ldots, X^{(T)}$). It allows the previous predictions to be used as inputs, using hidden states. The RNN is based on the multilayer perceptron which is an acyclic neural network structured in layers, namely, an input layer, one or more intermediate layers called hidden layers, and an output layer. The multilayer perceptron is described by the $n$ layers that compose it and which are successive. The layer $L \in [1, N]$ of multilayer perceptron is defined by: $T \cdot L = (n, L, \sigma _{L}, a_{L})$ where $n_{L} \in \mathbb{N}$ is the number of neurons in the layer $L$, $a_{L} : \mathbb{R}^{n_{L-1}} \rightarrow \mathbb{R}^{n_{L}}$ is the affine transformation defined by the vector $b_{L} \in \mathbb{R}^{n_{L}}$ and the matrix $W_{L} \in \mathbb{R}^{n_{L} \times n_{L}}$. Note that the $b_{L}$ is a bias vector, which is an additive set of weights in a neural network that requires no input.

For an input vector sequence $x(t)$ containing the features of an input dataset of attacks with $t \in [1, t_{f}]$, we have an output vector sequence $o(t)$ with $t \in [1, t_{f}]$ by the initialization of the internal state vectors as follows:
State_0(t) = x(t), \forall t \in [1, t_f] \tag{1}

and

State_L(0) = x(0), \forall t \in [1, N] \tag{2}

Then, for each time step, all the layers of the network are recursively applied to the previous layer’s output vector:

\[ g_L(t) = W_L \times h_L(t) - 1(t) + V_L \times h_L(t - 1) + b_L \] \tag{3}

\[ h_L(t) = \sigma_L(g_L(t)) \] \tag{4}

\[ o(t) = h_N(t) \] \tag{5}

To alleviate the vanishing gradient problem, there are two solutions, namely, Gated recurrent units (GRUs) and Long short-term memory (LSTM). We use in our algorithm the LSTM which is one of the most popular RNN architectures to date. The LSTM is described as follows [36]:

\[ \text{Cell}_t = \text{Forget}_t \odot \text{Cell}_t - 1 + \text{Input}_t \odot \tilde{\text{Cell}}_t \] \tag{6}

\[ \text{Forget}_t = \sigma(W_{fhh}_t - 1 + W_{fxx}_t + b_f) \] \tag{7}

\[ \text{Output}_t = \sigma(W_{ohh}_t - 1 + W_{oxx}_t + b_o) \] \tag{8}

\[ \text{Input}_t = \sigma(W_{ihh}_t - 1 + W_{ixx}_t + b_i) \] \tag{9}

\[ \tilde{\text{Cell}}_t = \tanh(W_{chh}_t - 1 + W_{cxx}_t + b_c) \] \tag{10}

\[ \text{Hidden}_t = \text{Output}_t \odot \tanh(\text{Cell}_t) \] \tag{11}

Where \text{Forget}_t is the forget gate, \text{Cell}_t is the memory cell, \text{Input}_t is the input gate, \text{Output}_t is the output gate, and \text{Hidden}_t is the new hidden state.

The proposed RNN-based IDS architecture for detection of cyber attacks in Agriculture 4.0 is presented in Algorithm 1 which is written in Python language. The description of each function used in Algorithms 1–3 are presented in Table 2.

Table 2. Functions used in Algorithms 1–3.

| Function        | Description                                                                 |
|-----------------|-----------------------------------------------------------------------------|
| \text{model} = \text{Sequential}() | Create a sequential model incrementally via the \text{add}() method. |
| \text{add}() | The \text{add}() method consists of adding layers. |
| \text{Dropout}() | The dropout is a regularization technique for neural networks and deep learning models, where randomly selected neurons are ignored during training. |
| \text{Dense}() | The dense layer is the regular deeply connected neural network layer. |
| \text{LSTM}() | Adding the Long Short-Term Memory layer. |
| \text{return\_sequences} | Determines whether to return the last output in the output sequence or the full sequence. |
| \text{input\_shape} | The shape of our training set. |
| \text{compile}() | Compile the model. |
| \text{model\_fit}() | Train the model, iterating on the data in batches of X samples. |
| \text{training\_loss} | Get training loss histories. |
| \text{test\_loss} | Get test loss histories. |
| \text{plt\_show}() | Visualize the confusion matrix. |
Table 2. Cont.

| Function                        | Description                                                                                                                                 |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Conv1D()                        | The Conv1D consists of creating a convolution kernel that is convolved with the layer input.                                                 |
| GlobalAveragePooling1D()        | Convert each feature map into one value.                                                                                                 |
| np.argmax                       | Create the confusion matrix.                                                                                                             |
| SGD()                           | Implements the stochastic gradient descent optimizer with a learning rate and momentum.                                                    |
| Accuracy                        | The number of correct predictions made as a ratio of all predictions made.                                                                |
| AreaUnderROCCurve               | A plot of the true positive rate and the false positive rate for a given set of probability predictions.                                    |
| ConfusionMatrix                 | The confusion matrix is a handy presentation of the accuracy of a model with two or more classes.                                        |
| classification_report()         | function displays the precision, recall, f1-score and support for each class.                                                             |
| ReLU                             | The rectified linear activation function.                                                                                                 |
| Sigmoid                          | The sigmoid activation function that takes any real value as input and outputs values in the range 0 to 1.                                   |
| Tanh                             | The hyperbolic tangent activation function that takes any real value as input and outputs values in the range −1 to 1.                     |

Algorithm 1 Build the model using RNN

Input: $x_{train.shape[2]}$, batch_size = 10,000

Initialization: Define Sequential model: $model = Sequential()$

1. model.add(LSTM(67,input_dim=x_train.shape[2], return_sequences=True))
2. model.add(LSTM(300, return_sequences=True))
3. model.add(Dropout(0.2))
4. model.add(LSTM(600, return_sequences=True))
5. model.add(Dropout(0.5))
6. model.add(LSTM(300, return_sequences=True))
7. model.add(Dropout(0.2))
8. model.add(LSTM(67, return_sequences=False))
9. model.add(Dropout(0.1))
10. model.add(Dense(y_train.shape[1], activation='softmax'))
11. model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy',Precision(),Recall()])
12. history=model.fit(x_train1, y_train1, batch_size=batch_size, epochs=15, validation_data=(x_test, y_test), class_weight=class_weights)
13. training_loss = history.history['loss']
14. test_loss = history.history['val_loss']
15. snn_pred = model.predict(d2_x_test,batch_size=10,000, verbose=1)
16. snn_predicted = np.argmax(snn_pred,axis=1)
17. y_eval = np.argmax(y_test,axis=1)
18. cm = confusion_matrix(np.argmax(y_test, axis=1), snn_predicted)
19. cm = (cm.astype('float') / cm.sum(axis=1)[np.newaxis])
20. snn_cm = (cm.astype('float') / cm.sum(axis=1)[np.newaxis])
21. snn_df_cm = pd.DataFrame(snn_cm, target_strings, target_strings)
22. plt.show()
Algorithm 2 Build the model using CNN

**Input:** \( x_{\text{train}}. \) shape[2], batch_size = 10,000

**Initialization:** Define Sequential model: \( \text{model} = \text{Sequential}() \)

1. model.add(Conv1D(input_shape=(None, 67), filters=64, kernel_size=3, activation='relu', padding='same'))
2. model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same'))
3. model.add(Conv1D(filters=16, kernel_size=2, activation='relu', padding='same'))
4. model.add(GlobalAveragePooling1D())
5. model.add(Dense(52, activation='relu'))
6. model.add(Dense(26, activation='relu'))
7. model.add(Dense(13, activation='softmax'))
8. model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy', Precision(), Recall()])
9. history = model.fit(x_train, y_train, batch_size=batch_size, epochs=35, validation_data=(x_test, y_test))
10. pyplot.plot(history.history['acc'])
11. pyplot.show()
12. training_loss = history.history['loss']
13. test_loss = history.history['val_loss']
14. snn_pred = model.predict(d2_x_test, batch_size=10,000, verbose=1)
15. snn_predicted = np.argmax(snn_pred, axis=1)
16. y_eval = np.argmax(y_test, axis=1)
17. cm = confusion_matrix(np.argmax(y_test, axis=1), snn_predicted)
18. snn_cm = (cm.astype('float') / cm.sum(axis=1)[:, np.newaxis])
19. snn_df_cm = pd.DataFrame(snn_cm, target_strings, target_strings)
20. pyplot.show()

Algorithm 3 Build the model using DNN

**Input:** \( x_{\text{train}}. \) shape[2], batch_size = 10,000

**Initialization:** Define Sequential model: \( \text{model} = \text{Sequential}() \)

1. model.add(Dense(134, input_dim=x.shape[1], activation='relu'))
2. model.add(Dropout(0.2))
3. model.add(Dense(60, activation='relu'))
4. model.add(Dropout(0.2))
5. model.add(Dense(26, activation='relu'))
6. model.add(Dropout(0.2))
7. model.add(Dense(13, activation='softmax'))
8. sgd = SGD(lr=0.05, momentum=0.8)
9. model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy', Precision(), Recall()])
10. history = model.fit(d2_x_train, y_train, validation_data=(d2_x_test, y_test), batch_size=10,000, verbose=1, epochs=30)
11. pyplot.plot(history.history['acc'])
12. pyplot.show()
13. training_loss = history.history['loss']
14. test_loss = history.history['val_loss']
15. snn_pred = model.predict(d2_x_test, batch_size=10,000, verbose=1)
16. snn_predicted = np.argmax(snn_pred, axis=1)
17. y_eval = np.argmax(y_test, axis=1)
18. cm = confusion_matrix(np.argmax(y_test, axis=1), snn_predicted)
19. snn_cm = (cm.astype('float') / cm.sum(axis=1)[:, np.newaxis])
20. snn_df_cm = pd.DataFrame(snn_cm, target_strings, target_strings)
21. pyplot.show()

3.3. Cnn-Based Ids

Convolutional networks, also known as convolutional neural networks, or CNNs, represent a dedicated class of neural network for data processing with a familiar network structure. The name “convolutional neural network” means that the network uses a...
mathematical operation called convolution [37]. Therefore, a convolutional neural network structure is composed of a set of processing layers, as follows:

- The convolution layer (CONV) that manages the data from a receiver cell. There are three hyperparameters to dimension the volume of the convolution layer: the depth, stride, and zero-padding. The formula for calculating the number of neurons in the output volume $W_o$ is described as follows:

$$W_o = \frac{W_i - S + 2M}{P} + 1$$  \hspace{1cm} (12)

where $W_i$ is the size of the input volume, $S$ is the kernel field size of the convolutional layer neurons, $M$ is the amount of zero padding, and $P$ is the stride.

- The pooling layer (POOL), which enables to reduce the size of the intermediate image by compressing the information and operates on each feature map independently.

- The correction layer (Rectified Linear Unit, ReLU), which is often referred to as the “ReLU” in reference to the activation function. The ReLU applies the non-saturating activation function, which is described as follows:

$$f(x) = \text{max}(0, x)$$  \hspace{1cm} (13)

Note that there are other functions that can be used to increase nonlinearity, such as the sigmoid function, which is described as follows:

$$S(x) = \frac{e^x}{e^x + 1}$$  \hspace{1cm} (14)

- The “fully connected” (FC) layer is a perceptron-type layer.

- The loss layer as the final layer of a neural network. Different loss functions can be used such as Euclidean loss, Softmax loss, and Sigmoid cross-entropy loss.

The proposed CNN-based IDS architecture for detection of cyber attacks in Agriculture 4.0 is presented in Algorithm 2 which is written in Python language.

3.4. Dnn-Based Ids

A deep neural network (DNN) is an artificial neural network (ANN) with more layers intermediate between input and output layers. The DNN consist of five-part: neurons, weights, synapses, biases, and functions. The inputs $(X = x_1, \ldots, x_n)$ are linked with weights $(W = w_1 \ldots w_n)$. The full-fledged deep learning system is based on multilayer perception (defined in RNN-based IDS) with the application of various activation functions that produce real values, rather than Boolean values as in the classical perceptron. To help to adjust the input weights and minimize the “loss”, the backpropagation algorithm is applied to perform iterative backward passes. The proposed DNN-based IDS architecture for detection of cyber attacks in Agriculture 4.0 is presented in Algorithm 3 which is written in Python language based on the following packages Pandas (https://pandas.pydata.org/pandas-docs/stable/ (accessed on 1 April 2021)), NumPy (https://numpy.org (accessed on 1 April 2021)), SciPy (https://scipy.org (accessed on 1 April 2021)), TensorFlow (https://tensorflow.org/ (accessed on 1 April 2021)), and Keras (https://keras.io/ (accessed on 1 April 2021)).

4. Performance Evaluation

Agriculture 4.0 consists of integrating advanced technologies into existing farm operations to improve the quality and productivity of agricultural products. These advanced technologies include IoT devices, 5G communications, Drones, Fog/Edge computing, Cloud Computing, Artificial Intelligence, Network Function Virtualization, and Software-Defined Networking. Based on these technologies, we used and selected the most recent data sets that contain DDoS attack scenarios against these technologies used by Agriculture 4.0.
Specifically, we used the following two new real traffic datasets, namely, CIC-DDoS2019 dataset [34] and TON_IoT dataset [35]. They are chosen for three reasons: (1) because they were built for a TCP/IP communication stack, (2) contain DDoS attacks, and (3) represent the nature of Agriculture 4.0 (in particular its IoT and IIoT sensors and cloud-edge traffic). The TON_IoT dataset was designed based on interacting network elements and IoT/IIoT systems with the three layers of Edge, Fog, and Cloud to simulate a real-world execution of current production IoT/IIoT networks. The NSX-VMware platform was utilized as Software-Defined Network (SDN) and Network Function Virtualisation (NFV) technologies to facilitate the management of the interaction between these three layers. The experiment is conducted on Google Colaboratory (https://colab.research.google.com (accessed on 1 April 2021)) using python 3 with the Graphics Processing Unit (GPU) and TensorFlow.

The details of the IDS experiment methodology are shown in Figure 4 and an overview of pre-processing of CIC-DDoS2019 dataset [34] and TON_IoT dataset [35] with the Deep learning-based IDS deployment is presented in Figure 5. More precisely, the approach is composed of four steps: (1) datasets step, (2) pre-processing step, (3) training step, and (4) testing step. The hyperparameters employed in deep learning strategies are shown in Table 3.
CICDDoS2019 dataset

- 50063112 rows, including 50006249 rows for DDoS attacks and 56863 rows for benign traffic. Each row having 86 features.

- Training dataset contains 12 DDoS attacks including NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN and TFTP.

- Test dataset contains 7 attacks including PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag and SYN in testing day.

**Label encoding phase**
- Duplicate rows, empty columns, and unnecessary label columns are dropped.
- Each row having 67 features.

**Reshaping phase**
- x_train = np.reshape(x_train, (x_train.shape[0], 1, x_train.shape[1]))
- x_test = np.reshape(x_test, (x_test.shape[0], 1, x_test.shape[1]))

**Normalization**
- scaler = StandardScaler()
- x = scaler.fit_transform(x)

**Dataset Pre-processing**
- Build the model using RNN
- Build the model using CNN
- Build the model using DNN
- Visualize the performance indicators, including detection rate, false alarm rate, precision, F-score, recall, TNR, FAR, ROC Curve, and accuracy.

**Deep learning-based IDS deployment**
- Uses the same features presented in the original dataset.

**TON_IoT dataset**

- Normal (35000 rows), DDoS (5000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), Ransomware (2865 rows), XSS (866 rows), and Scanning (529 rows). Each row having 7 features.

- Normal (35000 rows), DDoS (5000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), Ransomware (2833 rows), XSS (577 rows), and Scanning (550 rows). Each row having 7 features.

- Normal (70000 rows), DDoS (10000 rows), Injection (10000), Password (10000 rows), Backdoor (10000 rows), Ransomware (5804 rows), XSS (2312 rows), and Scanning (1058 rows). Each row having 6 features.

- Normal (35000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), XSS (577 rows), and Scanning (529 rows). Each row having 7 features.

- Normal (70000 rows), DDoS (10000 rows), Injection (10000), Password (10000 rows), Backdoor (10000 rows), Ransomware (4528 rows), XSS (898 rows), and Scanning (3550 rows). Each row having 6 features.

- Normal (35000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), XSS (577 rows), and Scanning (529 rows). Each row having 7 features.

**Figure 5.** Overview of pre-processing of CIC-DDoS2019 dataset [34] and TON_IoT dataset [35] with Deep learning-based IDS deployment.
### Table 3. The hyperparameters employed in deep learning approaches.

| Hyperparameter                | Value          |
|-------------------------------|----------------|
| Activation function           | Sigmoid        |
| Classification function       | SoftMax        |
| Batch size                    | 10,000         |
| Hidden nodes (HN)             | 15–100         |
| Number of epoch               | 100            |
| Learning rate (LR)            | 0.01–0.5       |

#### 4.1. Pre-Processing of the CIC-DDoS2019 Dataset

The CIC-DDoS2019 dataset [34] includes 50,063,112 records, including, 50,006,249 rows for DDoS attacks and 56,863 rows for benign traffic. Each row having 86 features. The statistics for attacks in training and testing for the dataset are summarized in Table 4. The training dataset contains 12 DDoS attacks including, Network Time Protocol (NTP), Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Microsoft SQL Server (MSSQL), Network Basic Input Output System (NetBIOS), Simple Network Management Protocol (SNMP), Simple Service Discovery Protocol (SSDP), User Datagram Protocol (UDP), UDP-Lag, WebDDoS, SYN and TFTP, while the test dataset contains seven attacks, namely, MSSQL, NetBIOS, PortScan, LDAP, UDP, UDP-Lag and SYN in testing day.

- **NTP-based attack**: is a DDoS attack based on a reflection where an attacker uses Network Time Protocol (NTP) server functionality to flood a specific client-server or other networks with an increased quantity of UDP data traffic. This attack can make the destination and its network infrastructure unavailable to normal traffic.
- **DNS-based attack**: is a DDoS attack based on a reflection where an attacker uses a Botnet to create a large number of resolution requests to a targeted IP address. This attack can generate large responses (amplified), reflected to a target server.
- **LDAP-based attack**: is a DDoS attack based on a reflection where an attacker sends requests to a publicly available vulnerable LDAP server to generate large responses (amplified), reflected to a target server.
- **MSSQL-based attack**: is a DDoS attack based on a reflection where an attacker exploits the Microsoft SQL Server Resolution Protocol (MC-SQLR) by executing scripted requests using a forged IP address in order to appear as coming from the target server.
- **NetBIOS-based attack**: is a DDoS attack based on a reflection where an attacker sends spoofed “Name Release” or “Name Conflict” messages to a victim machine in order to refuse all NetBIOS network traffic.
- **SNMP-based attack**: This attack is a volumetric DDoS threat that uses the Simple Network Management Protocol (SNMP) to generate attack volumes of hundreds of gigabits per second in order to clog the target’s network pipes.
- **SSDP-based attack**: is a DDoS attack based on a reflection where an attacker sends an amplified amount of traffic to a targeted victim using Universal Plug and Play (UPnP) networking protocols.
- **UDP-Lag-based attack**: This attack aims to slow down/interrupt the targeted host with IP packets containing UDP datagrams.
- **WebDDoS-based attack**: This threat takes advantage of legitimate HTTP GET or POST queries to compromise a Web server or application.
- **SYN-based attack**: This attack exploits the normal TCP three-way handshake (i.e., sending SYN (synchronize), sending SYN-ACK (synchronize-acknowledge), and responds with an ACK (acknowledge)) to use resources on the targeted network server and make it unresponsive.
- **TFTP-based attack**: This attack exploits the Trivial File Transfer Protocol (TFTP) by employing TFTP servers connected to the internet. Specifically, an attacker performs
a request by default for a file, and the victim TFTP server sends the data back to the requesting target host.

- PortScan-based attack: This attack performs a network security audit by conducting port scanning on a specific machine or on an entire network. The scanning is done using queries to determine which services are running on a remote host.

Table 4. Attack types in CICDDoS2019 dataset.

| Attack Type          | Flow Count |
|----------------------|------------|
| Benign               | 56,863     |
| DDoS_DNS             | 5,071,011  |
| DDoS_LDAP            | 2,179,930  |
| DDoS_MSSQL           | 4,522,492  |
| DDoS_NetBIOS         | 4,093,279  |
| DDoS_NTP             | 1,202,642  |
| DDoS_SNMP            | 5,159,870  |
| DDoS_SSDP            | 2,610,611  |
| DDoS_SYN             | 1,582,289  |
| DDoS_TFTP            | 20,082,580 |
| DDoS_UDP             | 3,134,645  |
| DDoS_UDP-Lag         | 366,461    |
| DDoS_WebDDoS         | 439        |

To analyze the efficiency of machine learning and deep learning strategies in terms of binary classification (i.e., Classification tasks with two classes) and multi-class classification (i.e., Classification tasks with more than two classes), we create three different datasets, named Dataset_2_class, Dataset_7_class, and Dataset_13_class. The statistics for attacks in training and testing for each dataset are summarized in Tables 5–7, respectively.

Table 5. Attack types in Dataset_2_class.

| Category   | Training | Test |
|------------|----------|------|
| Benign     | 56,101   | 17,146 |
| Attack     | 997,054  | 314,716 |

Table 6. Attack types in Dataset_7_class.

| Category              | Type of Attack   | Training | Test  |
|-----------------------|------------------|----------|-------|
| Reflection-based attacks | DrDoS_NetBIOS    | 619,700  | 136,729 |
|                       | DrDoS_MSSQL      | 619,446  | 157,076 |
|                       | DrDoS_LDAP       | 619,251  | 150,701 |
| Exploitation-based attacks | DrDoS_UDP        | 618,696  | 150,706 |
|                       | UDP-lag          | 183,662  | 1873   |
|                       | Syn              | 790,662  | 150,416 |
| Exploitation/Reflection-based attacks | Others DoS attacks | 938,733  | 28,127 |
| Benign                | Benign           | 56,101   | 17,146 |
Table 7. Attack types in Dataset_13_class.

| Category          | Type of Attack | Flow Count | Training/Test |
|-------------------|---------------|------------|---------------|
| BENIGN            | BENIGN        | 56,101     |               |
|                   | DrDoS_LDAP    | 99,943     |               |
|                   | DrDoS_SSDP    | 98,576     |               |
|                   | DrDoS_DNS     | 96,567     |               |
|                   | DrDoS_MSSQL   | 95,700     |               |
|                   | DrDoS_NetBIOS | 93,560     |               |
|                   | DrDoS_SNMP    | 91,578     |               |
|                   | TFTP          | 72,116     |               |
| Reflection-based  |               |            |               |
| attacks           |               |            |               |
| Exploitation      |               |            |               |
| based attacks     |               |            |               |

4.2. Pre-Processing of the Ton_IoT Dataset

The TON_IoT dataset [35] is a new testbed for an IIoT network that contains three types of data, namely, network data, operating systems data, and telemetry data [38]. The telemetry datasets of IoT and IIoT sensors are presented in 7 files as presented in Table 8. The contents of these files are described as following:

- File 1 “Train_Test_IoT_Weather”: It contains Normal (35,000 rows), DDoS (5000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), Ransomware (2865 rows), XSS (866 rows), and Scanning (529 rows). The file presents the IoT data of temperature measurements, pressure readings, and humidity readings of a weather sensor linked to the network.
- File 2 “Train_Test_IoT_Fridge”: It contains Normal (35,000 rows), DDoS (5000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), Ransomware (2902 rows), and XSS (2942 rows). The file presents the IoT data of temperature measurements and temperature conditions of a fridge sensor linked to the network.
- File 3 “Train_Test_IoT_Garage_Door”: It contains Normal (70,000 rows), DDoS (10,000 rows), Injection (10,000), Password (10,000 rows), Backdoor (10,000 rows), Ransomware (5804 rows), XSS (2312 rows), and Scanning (1058 rows). The file presents the IoT data of a door sensor linked to the network where the door is closed or open.
- File 4 “Train_Test_IoT_GPS_Tracker”: It contains Normal (35,000 rows), DDoS (5000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), Ransomware (2833 rows), XSS (577 rows), and Scanning (550 rows). The file presents the IoT data of latitude value and longitude value of GPS tracker sensor linked to the network.
- File 5 “Train_Test_IoT_Modbus”: It contains Normal (35,000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), XSS (577 rows), and Scanning (529 rows). The file presents the IoT data of Modbus function code that is responsible for reading an input register.
- File 6 “Train_Test_IoT_Motion_Light”: It contains Normal (35,000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), XSS (898 rows), and Scanning (3550 rows). The file presents the IoT data of a light sensor that is either on or off.
- File 7 “Train_Test_IoT_Thermostat”: It contains Normal (35,000 rows), Injection (5000), Password (5000 rows), Backdoor (5000 rows), Ransomware (2264 rows), XSS

Splitting the data between train / test

\[ \text{x\_train, x\_test, y\_train, y\_test} = \text{train\_test\_split(x, y, test\_size = 0.25, stratify = y)} \]
(449 rows), and Scanning (61 rows). The file presents the IoT data of the current temperature reading of a thermostat sensor connected with the network.

Table 8. Attack types in TON_IoT dataset.

| TON_IoT Dataset              | Attack Type | Flow Count |
|------------------------------|-------------|------------|
| Train_Test_IoT_Weather       | Normal      | 35,000     |
|                              | DDoS        | 5000       |
|                              | Injection   | 5000       |
|                              | Password    | 5000       |
|                              | Backdoor    | 5000       |
|                              | Ransomware  | 2865       |
|                              | XSS         | 866        |
|                              | Scanning    | 529        |
| Train_Test_IoT_Fridge        | Normal      | 35,000     |
|                              | DDoS        | 5000       |
|                              | Injection   | 5000       |
|                              | Password    | 5000       |
|                              | Backdoor    | 5000       |
|                              | Ransomware  | 5000       |
|                              | XSS         | 2942       |
| Train_Test_IoT_Garage_Door   | Normal      | 70,000     |
|                              | DDoS        | 10,000     |
|                              | Injection   | 10,000     |
|                              | Password    | 10,000     |
|                              | Backdoor    | 100,000    |
|                              | Ransomware  | 5804       |
|                              | XSS         | 2312       |
|                              | Scanning    | 1058       |
| Train_Test_IoT_GPS_Tracker   | Normal      | 35,000     |
|                              | DDoS        | 5000       |
|                              | Injection   | 5000       |
|                              | Password    | 5000       |
|                              | Backdoor    | 5000       |
|                              | Ransomware  | 2833       |
|                              | XSS         | 577        |
|                              | Scanning    | 550        |
### Table 8. Cont.

| TON_IoT Dataset                  | Attack Type | Flow Count |
|----------------------------------|-------------|------------|
| **Train_Test_IoT_Modbus**        | Normal      | 35,000     |
|                                  | Injection   | 5000       |
|                                  | Password    | 5000       |
|                                  | Backdoor    | 5000       |
|                                  | XSS         | 577        |
|                                  | Scanning    | 529        |
| **Train_Test_IoT_Motion_Light**  | Normal      | 70,000     |
|                                  | DDoS        | 10,000     |
|                                  | Injection   | 10,000     |
|                                  | Password    | 10,000     |
|                                  | Backdoor    | 10,000     |
|                                  | Ransomware  | 4528       |
|                                  | XSS         | 898        |
|                                  | Scanning    | 3550       |
| **Train_Test_IoT_Thermostat**    | Normal      | 35,000     |
|                                  | Injection   | 5000       |
|                                  | Password    | 5000       |
|                                  | Backdoor    | 5000       |
|                                  | Ransomware  | 2264       |
|                                  | XSS         | 449        |
|                                  | Scanning    | 61         |

#### 4.3. Performance Metrics

The performance metrics chosen to evaluate machine learning and deep learning strategies is very important. In our study, we focus on the following important performance metrics: detection rate (DR), false alarm rate (FAR), precision, F-score, recall, TNR, FAR, ROC Curve, and accuracy. Table 9 illustrates four possibilities for both correct and erroneous classification.

\[
TNR_{BENIGN} = \frac{TN_{BENIGN}}{TN_{BENIGN} + FP_{BENIGN}} \quad (15)
\]

\[
FAR = \frac{FP_{BENIGN}}{TN_{BENIGN} + FP_{BENIGN}} \quad (16)
\]

\[
Precision = \frac{TP_{Attack}}{TP_{Attack} + FP_{BENIGN}} \quad (17)
\]

\[
Recall = \frac{TP_{Attack}}{TP_{Attack} + FN_{Attack}} \quad (18)
\]

\[
DR_{Attack} = \frac{TP_{Attack}}{TP_{Attack} + FN_{Attack}} \quad (19)
\]

\[
F-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (20)
\]

\[
Accuracy = \frac{TP_{Attack} + TN_{BENIGN}}{TP_{Attack} + FN_{Attack} + TN_{BENIGN} + FP_{BENIGN}} \quad (21)
\]
\[
DR_{\text{Overall}} = \frac{\sum TP_{\text{Each - Attack - Type}}}{\sum TP_{\text{Each - Attack - Type}} + \sum FN_{\text{Each - Attack - Type}}}
\]  

where \(TP\), \(TN\), \(FP\), and \(FN\) denote true positive, true negative, false positive, and false negative, respectively. The False Positive (FP) indicates the benign data that is incorrectly classified as an attack, while the True Negative (TN) indicates the benign data that is correctly classified as benign. The True Positive (TP) indicates the attack data that is correctly classified as an attack. The False Negative (FN) indicates the attack data that is incorrectly classified as benign.

Table 9. Confusion matrix.

| Predicted Class | Negative Class | Positive Class |
|-----------------|----------------|---------------|
| Class           | True negative (TN) | False positive (FP) |
| Negative class  | True negative (TN) | False positive (FP) |
| Positive class  | False negative (FN) | True positive (TP) |

4.4. Results

The performance of deep learning strategies relative to other machine learning strategies (i.e., DT: Decision Tree, RF: Random forests, NB: Naive Bayes, LR: Logistic Regression) in terms of Precision, Recall, and F-score are shown in Figure 6. In term of Precision, the deep learning techniques give good results in comparison to other machine learning strategies, namely, decision tree, random forests, naïve bayes, and logistic regression, and the convolutional neural network model provides the higher ratio with 91%. Both in terms of Recall and F-score, deep learning techniques give good results in comparison to other machine learning strategies, in which the convolutional neural network model provides the higher ratio with 90% and 89%, respectively. The results show that deep learning techniques can provide better performance in cyber security intrusion detection for Agriculture 4.0.

The performance experimental results of deep learning techniques in term of Precision under binary classification and multiclass classification are shown in Figure 7. The results show that deep learning techniques give a higher positive prediction in terms of binary classification compared to multiclass classification. Specifically, the convolutional neural network model achieves a precision of 99% in binary classification compared to 90% in multiclass classification. Therefore, the performance experimental results of deep learning techniques in term of Recall with binary classification compared to multiclass classification are shown in Figure 8. The results show that deep learning techniques give a higher positive prediction in terms of binary classification compared to multiclass classification. Specifically, the deep neural network model achieves a recall of 99% in binary classification compared to 83% and 62% in multiclass classification. In addition, the recurrent neural network model achieves an F-score of 99% in binary classification compared to 88% and 56% in multiclass classification.
Figure 6. The performance experimental results of deep learning approaches relative to other machine learning strategies in terms of Precision, Recall, and F-score.

Figure 7. The performance experimental results of deep learning approaches in term of Precision with binary classification and multiclass classification.
Figure 8. The performance experimental results of deep learning approaches in term of Recall with binary classification and multiclass classification.

Table 10 presents the performance of deep learning approaches relative to benign and various types of attacks in Dataset_7_class (i.e., multi-class classification). We can observe that the convolutional neural network model provides the higher true negative rate with 99% and the highest detection rate for two attacks type, namely, DrDoS_LDAP and Syn. The deep neural network model gives the higher detection ratio for two attack types, namely, DrDoS_MSSQL and Syn. The recurrent neural network model gives the higher detection ratio for four attack types, namely, DrDoS_LDAP, DrDoS_NetBIOS, DrDoS_UDP, and Syn. In addition, we observe a low detection of UDP-lag attack and this is due to the low learning rate of this attack.

Table 10. The performance experimental results of deep learning approaches relative to benign and various types of attacks in Dataset_7_class (Multi-class classification).

|                  | DNN | RNN | CNN |
|------------------|-----|-----|-----|
| TNR (BENIGN)     | 95% | 98% | 99% |
| DrDoS_LDAP       | 96% | 98% | 97% |
| DrDoS_MSSQL      | 96% | 94% | 95% |
| DrDoS_NetBIOS    | 69% | 99% | 94% |
| DrDoS_UDP        | 60% | 71% | 71% |
| Syn              | 100%| 100%| 100%|
| UDP-lag          | 0%  | 0%  | 0%  |

Table 11 presents the performance of deep learning approaches relative to normal and various types of attacks in TON_IoT dataset (i.e., multi-class classification). We can observe that all three deep learning techniques provide a higher true negative rate. The recurrent neural network model gives a higher detection ratio for three attack types, namely, Injection, Password, and Scanning. The convolutional neural network gives the higher detection ratio for five attacks, namely, DDoS, Backdoor, Ransomware, XSS, and Scanning.
Table 11. The performance experimental results of deep learning approaches relative to normal and various types of attacks in TON_IoT dataset (Multi-class classification).

|              | DNN | RNN | CNN |
|--------------|-----|-----|-----|
| Normal       | 93% | 97% | 96% |
| DDoS         | 94% | 95% | 98% |
| Injection    | 92% | 97% | 94% |
| Password     | 91% | 97% | 93% |
| Backdoor     | 93% | 95% | 96% |
| Ransomware   | 94% | 96% | 97% |
| XSS          | 94% | 96% | 97% |
| Scanning     | 94% | 97% | 97% |

Table 12 presents the performance of deep learning approaches relative to benign and various types of attacks in Dataset_13_class (i.e., multi-class classification). We can observe that all three deep learning techniques provide the higher true negative rate. The deep neural network model gives the higher detection ratio for five attack types, namely, DrDoS_DNS, DrDoS_NTP, DrDoS_SSDP, TFTP, and UDP-lag. The recurrent neural network model gives the higher detection ratio for four attack types, namely, DrDoS_MSSQL, DrDoS_NTP, DrDoS_NetBIOS, and DrDoS_UDP. The convolutional neural network gives the higher detection ratio for Syn attack with 65%. Therefore, we observe a low detection of WebDDoS attack and this is due to the low learning rate of this attack. For binary classification, we can be seen that all three deep learning techniques provide the higher true negative rate with 99% and the highest detection rate with 100%, as presented in Table 13.

Table 12. The performance experimental results of deep learning approaches relative to benign and various types of attacks in Dataset_13_class (Multi-class classification).

|              | DNN | RNN | CNN |
|--------------|-----|-----|-----|
| TNR (BENIGN) | 100%| 100%| 100%|
| DrDoS_DNS    | 61% | 56% | 58% |
| DrDoS_LDAP   | 47% | 47% | 47% |
| DrDoS_SNMP   | 67% | 67% | 67% |
| DrDoS_SSDP   | 61% | 58% | 52% |
| DrDoS_UDP    | 47% | 48% | 46% |
| DrDoS_NetBIOS| 93% | 97% | 73% |
| DrDoS_MSSQL  | 55% | 56% | 55% |
| Syn          | 64% | 64% | 65% |
| TFTP         | 100%| 99% | 94% |
| DrDoS_NTP    | 91% | 91% | 90% |
| WebDDoS      | 23% | 24% | 20% |
| UDP-lag      | 99% | 98% | 97% |
The performance experimental results of deep learning approaches relative in Dataset_2_class (Binary classification).

|                  | DNN  | RNN  | CNN  |
|------------------|------|------|------|
| TNR (BENIGN)     | 96%  | 99%  | 99%  |
| Attack           | 100% | 100% | 100% |

The performance experimental results of deep learning approaches in term of F-score with binary classification and multiclass classification are presented in Figure 9. The results show that deep learning techniques can provide better performance in cyber security intrusion detection for Agriculture 4.0. The Receiver Operating Characteristic (ROC) curve for Dataset_13_class is depicted in Figure 10, which is a plot of intrusion detection accuracy against the false positive probability. We can identify the obvious better performance of both deep learning techniques, namely, convolutional neural network and recurrent neural network, since all the values of AUC (Area Under Curve) for three classes, including, class 0: BENIGN, class 1: DrDoS_DNS, and class 2: DrDoS_LDAP, are between 0.94 and 1.00.
Dataset_7_class, Dataset_13_class, and TON_IoT dataset. In binary class classification (i.e., Dataset_2_class) and TON_IoT dataset, the convolutional neural network achieves high accuracy of 99.95% compared to both recurrent neural network and deep neural network, when the number of hidden nodes is 100 and the learning rate is 0.5. In multi-class classification (i.e., Dataset_7_class), the recurrent neural network achieves high accuracy of 93.88% compared to both deep neural network and convolutional neural network, when the learning rate is 0.5 and the number of hidden nodes is 30. In multi-class classification (i.e., Dataset_13_class), the convolutional neural network achieves high accuracy of 95.12% compared to both recurrent neural network and deep neural network, when the number of hidden nodes is 100 and the learning rate is 0.5. These results show that the recurrent neural network is efficient compared to the convolutional neural network with a lower number of hidden nodes.

Table 14. The accuracy, FAR, and training time of deep learning approaches with different hidden nodes and learning rate in four datasets, namely, Dataset_2_class, Dataset_7_class, Dataset_13_class, and TON_IoT dataset.

| Parameters | Performance Metrics | Dataset_2_Class | Dataset_7_Class | Dataset_13_Class | TON_IoT Dataset |
|------------|---------------------|-----------------|-----------------|-----------------|-----------------|
| HN = 30    | ACC                 | 99.92%          | 99.93%          | 99.90%          | 99.90%          |
| LR = 0.01  | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 31              | 60              | 30              | 122             |
| HN = 30    | ACC                 | 99.92%          | 99.93%          | 99.90%          | 99.90%          |
| LR = 0.1   | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 34              | 66              | 35              | 123             |
| HN = 30    | ACC                 | 99.92%          | 99.93%          | 99.90%          | 99.90%          |
| LR = 0.5   | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 38              | 69              | 39              | 138             |
| HN = 60    | ACC                 | 99.93%          | 99.94%          | 99.93%          | 99.90%          |
| LR = 0.01  | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 39              | 63              | 34              | 126             |
| HN = 60    | ACC                 | 99.93%          | 99.94%          | 99.93%          | 99.90%          |
| LR = 0.1   | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 42              | 69              | 40              | 140             |
| HN = 100   | ACC                 | 99.93%          | 99.94%          | 99.93%          | 99.90%          |
| LR = 0.01  | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 42              | 72              | 42              | 123             |
| HN = 100   | ACC                 | 99.93%          | 99.94%          | 99.93%          | 99.90%          |
| LR = 0.1   | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 60              | 102             | 80              | 152             |
| HN = 100   | ACC                 | 99.93%          | 99.94%          | 99.93%          | 99.90%          |
| LR = 0.5   | FAR                 | 1.14%           | 1.13%           | 1.19%           | 1.21%           |
|            | Time                | 102             | 151             | 120             | 180             |

Table 15 compares the performance of our work with other state-of-the-art methods that are tested under the CIC-DDoS2019 dataset and the TON_IoT dataset. The comparison is conducted with respect to network model, dataset, task, machine learning model, and accuracy. We can observe that our IDS model based on CNN incur the best results in terms of accuracy. This is due to the strategy of our pre-processing for both datasets that reduce the computation complexity and due to the use of simple deep learning models with larger batch sizes and fewer layers.
Table 15. Comparison with related work tested on CIC-DDoS2019 dataset and TON_IoT dataset.

| IDS Model          | Year  | Network Model                        | Dataset                  | Task Modality                  | Model                  | Accuracy  |
|--------------------|-------|--------------------------------------|--------------------------|-------------------------------|------------------------|-----------|
| Jia et al. [39]    | 2020  | IoT application                      | CIC-DDoS2019 dataset     | Multiclass (13 class)         | LSTM                   | 98.9%     |
| Li et al. [40]     | 2020  | IoT application                      | CIC-DDoS2019 dataset     | Multiclass (13 class)         | LSTM                   | N/A       |
| de Assis et al. [41]| 2020  | SDN environments in IoT networks     | CIC-DDoS2019 dataset     | Multiclass (13 class)         | CNN                    | 95.4%     |
| Alami et al. [42]  | 2020  | SDN environments in IoT networks     | CIC-DDoS2019 dataset     | Multiclass (13 class)         | Extreme gradient boosting algorithm | 91.26% |
| Zhang et al. [43]  | 2020  | SDN environments in IoT networks     | TON_IoT dataset          | Multiclass                    | Random Forest          | 99.68%    |
| Kumar et al. [44]  | 2021  | IoT application                      | TON_IoT dataset          | Multiclass                    | Extreme gradient boosting algorithm | 97.45% |
| Pouries et al. [45]| 2021  | N/A                                  | CIC-DDoS2019 dataset     | Multiclass (13 class)         | Energy-based flow classifier | 99.1%     |
| de Assis et al. [46]| 2021  | SDN environments                     | CIC-DDoS2019 dataset     | Binary (2 class)              | GRU                     | 99.6%     |
| Kumar et al. [47]  | 2021  | IoT application                      | TON_IoT dataset          | Multiclass                    | Extreme gradient boosting algorithm | 96.3%     |
| Javeed et al. [48] | 2021  | SDN environments in IoT networks     | CIC-DDoS2019 dataset     | Multiclass (13 class)         | LSTM and GRU           | 99.74%    |
| Nie et al. [49]    | 2021  | IoT application                      | CIC-DDoS2019 dataset     | Multiclass (13 class)         | Generative adversarial network | 98.35%   |
| Kumar et al. [50]  | 2021  | Smart agricultural unmanned aerial vehicles | TON_IoT dataset      | Multiclass                    | Stacked Long-Short-Term Memory | 88.82% |
| Our model          |       | Agricultural 4.0 based on the following IoT technologies, including, IoT devices, 5G communications, Drone, Fog/Edge computing, Cloud Computing, NFV, and SDN | CIC-DDoS2019 dataset     | Multiclass (7 class)         | CNN                    | 95.90%    |
|                    |       |                                      |                          | Binary (2 class)              | RNN                    | 94.99%    |
|                    |       |                                      |                          |                              | DNN                    | 94.91%    |
|                    |       |                                      |                          | Multiclass (7 class)         | CNN                    | 95.12%    |
|                    |       |                                      |                          | Binary (2 class)              | RNN                    | 94.88%    |
|                    |       |                                      |                          |                              | DNN                    | 95.88%    |
|                    |       |                                      |                          | Multiclass (13 class)         | CNN                    | 99.92%    |
|                    |       |                                      |                          | Binary (2 class)              | RNN                    | 98.94%    |
|                    |       |                                      |                          |                              | DNN                    | 98.93%    |

5. Conclusions

In this paper, we proposed three deep learning-based IDS models, including a convolutional neural network-based IDS model, a deep neural network-based IDS model, and a recurrent neural network-based IDS model. Specifically, we provided a performance evaluation and comparative analysis of machine learning and deep learning approaches for cyber security in agriculture 4.0. Each model’s performance is studied within two classification types (binary and multiclass) using two new real traffic datasets; namely, CIC-DDoS2019 dataset and TON_IoT dataset. The results show that deep learning techniques give good results in comparison to other machine learning strategies (e.g., decision tree, random forests, naive bayes, and logistic regression) in terms of important performance indicators, including detection rate, false alarm rate, precision, F-score, recall, true negative rate, false accept rate, ROC Curve, and accuracy. In addition, the IDS model based on CNN outperforms the state-of-the-art deep learning IDS methods, which were tested under the CIC-DDoS2019 dataset and TON_IoT dataset, by recording an accuracy of 99.95% for binary traffic detection and 99.92% for multiclass traffic detection.

Author Contributions: Conceptualization, M.A.F., L.S., H.D. and K.-K.R.C.; Methodology, M.A.F., L.S., H.D. and K.-K.R.C.; Software, M.A.F., L.S., and H.D.; Validation, M.A.F., L.S., and H.D.; formal analysis, M.A.F., L.S., and H.D.; investigation, M.A.F., L.S., and H.D.; resources, M.A.F., L.S., and H.D.; data curation, M.A.F., L.S., and H.D.; writing—original draft preparation, M.A.F., L.S., and H.D.; writing—review and editing, M.A.F., L.S., and K.-K.R.C.; visualization, M.A.F., L.S.; supervision, M.A.F., L.S. and K.-K.R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Research Start-Up Fund for Talent Researcher of Nanjing Agricultural University under Grant 77H0603 and in part by the National Natural Science...
References

1. Chen, B.; Wan, J.; Shu, L.; Li, P.; Mukherjee, M.; Yin, B. Smart factory of industry 4.0: Key technologies, application case, and challenges. *IEEE Access* **2017**, *6*, 6505–6519. [CrossRef]
2. Friha, O.; Ferrag, M.A.; Shu, L.; Maglaras, L.; Wang, X. Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey of Emerging Technologies. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 718–752. [CrossRef]
3. Liu, Y.; Ma, X.; Shu, L.; Hancke, G.P.; Abu-Mahfouz, A.M. From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges. *IEEE Trans. Ind. Inform.* **2020**, *17*, 4322–4334.
4. Ferrag, M.A.; Shu, L.; Yang, X.; Derhab, A.; Maglaras, L. Security and Privacy for Green IoT-Based Agriculture: Review, Blockchain Solutions, and Challenges. *IEEE Access* **2020**, *8*, 32031–32053. [CrossRef]
5. Yang, X.; Shu, L.; Chen, J.; Ferrag, M.A.; Wu, J.; Nurellari, E.; Huang, K. A Survey on Smart Agriculture: Development Modes, Technologies, and Security and Privacy Challenges. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 273–302. [CrossRef]
6. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [CrossRef]
7. Chen, J.W.; Lin, W.J.; Cheng, H.J.; Hung, C.L.; Lin, C.Y.; Chen, S.P. A smartphone-based application for scale pest detection using multiple-object detection methods. *Electronics* **2021**, *10*, 372. [CrossRef]
8. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
9. Muna, A.H.; Sitnikova, E. Developing a Security Testbed for Industrial Internet of Things. *IEEE Internet Things J.* **2020**, *8*, 5558–5573.
10. Kasongo, S.M.; Sun, Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Comput. Secur.* **2020**, *92*, 101752. [CrossRef]
11. Hassan, M.M.; Gumaei, A.; Alsanaad, A.; Alrubai, M.; Fortino, G. A hybrid deep learning model for efficient intrusion detection in big data environment. *Inf. Sci.* **2020**, *513*, 386–396. [CrossRef]
12. Li, B.; Wu, Y.; Song, J.; Lu, R.; Li, T.; Zhao, L. DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems. *IEEE Trans. Ind. Inform.* **2020**, *17*, 5615–5624. [CrossRef]
13. Gao, J.; Gan, L.; Buschendorf, F.; Zhang, L.; Liu, H.; Li, P.; Dong, X.; Lu, T. Omni SCADA intrusion detection using deep learning algorithms. *IEEE Internet Things J.* **2020**, *8*, 951–961. [CrossRef]
14. Ferrag, M.A.; Maglaras, L. DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids. *IEEE Trans. Eng. Manag.* **2019**, *67*, 1285–1297. [CrossRef]
15. Nie, L.; Ning, Z.; Wang, X.; Hu, X.; Li, Y.; Cheng, J. Data-Driven Intrusion Detection for Intelligent Internet of Vehicles: A Deep Convolutional Neural Network-based Method. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 2219–2230. [CrossRef]
16. Abusitta, A.; Bellaiche, M.; Dagenais, M.; Halabi, T. A deep learning approach for proactive multi-cloud cooperative intrusion detection system. *Future Gener. Comput. Syst.* **2019**, *98*, 308–318. [CrossRef]
17. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
18. Top 8 Challenges for Machine Learning Practitioners. Available online: https://towardsdatascience.com/top-8-challenges-for-machine-learning-practitioners-c4c0130701a1 (accessed on 1 May 2021).
19. Ferrag, M.A.; Maglaras, L.; Janicke, H.; Smith, R. Deep learning techniques for cyber security intrusion detection: A detailed analysis. In *Proceedings of the 6th International Symposium for ICS & SCADA Cyber Security Research 2019*, Athens, Greece, 10–12 September 2019; pp. 126–136. [CrossRef]
20. Diro, A.A.; Chilamkurti, N. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *82*, 761–768. [CrossRef]
21. Muna, A.H.; Moustafa, N.; Sitnikova, E. Identification of malicious activities in industrial internet of things based on deep learning models. *J. Inf. Secur. Appl.* **2018**, *41*, 1–11.
22. HaddadPajouh, H.; Dehghanian, A.; Khayami, R.; Choo, K.K.R. A deep recurrent neural network based approach for internet of things malware threat hunting. *Future Gener. Comput. Syst.* **2018**, *85*, 88–96. [CrossRef]
23. Vinayakumar, R.; Alazab, M.; Srinivasan, S.; Pham, Q.V.; Padannayil, S.K.; Simran, K. A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities. *IEEE Trans. Ind. Appl.* **2020**, *56*, 4436–4456. [CrossRef]
24. Parra, G.D.L.T.; Rad, P.; Choo, K.K.R.; Beebe, N. Detecting Internet of Things attacks using distributed deep learning. *J. Netw. Comput. Appl.* **2020**, *163*, 102662. [CrossRef]

Foundation of China under Grant 62072248. The work of K.-K. R. Choo was supported only by the Cloud Technology Endowed Professorship.

**Data Availability Statement:** We used the CIC-DDoS2019 dataset and TON_IoT dataset, which are publicly accessed datasets (https://www.unb.ca/cic/datasets/ddos-2019.html, https://ieee-dataport.org/documents/toniot-datasets) (accessed on 1 April 2021), for the evaluation of the proposed IDS.

**Conflicts of Interest:** All authors declare no conflict of interest.
