Detecting Data Poisoning Attacks using Federated Learning with Deep Neural Networks: An Empirical Study

Hatim Alsuwat
Department of Computer Science, College of Computers and Information Systems
Umm Al-Qura University, Makkah, Saudi Arabia

***Abstract*—The advent of intelligent networks powered by machine learning (ML) methods over the past few years has dramatically facilitated various facets of human lives, including healthcare, transportation, and entertainment. However, the use of ML in intelligent networks raises serious concerns about privacy and security, particularly in the context of data poisoning attacks. In order to address these concerns, this research paper presents a novel technique for detecting data poisoning attacks in intelligent networks, focusing on addressing privacy and security concerns associated with the use of machine learning (ML) methods. The research combines federated learning and deep learning approaches to analyze network data in a distributed and privacy-preserving manner. The technique employs a federated neural network to identify malicious data by analyzing network traffic, leveraging the power of Bayesian convolutional neural networks for efficient and accurate detection. The research follows an empirical approach, conducting experimental analyses to evaluate the proposed technique's effectiveness in terms of network security and data classification. The results demonstrate significant performance, including high throughput, quality of service, transmission rate, and low root mean square error for network security. Furthermore, the technique achieves impressive accuracy, recall, precision and malicious data analysis for data detection. The findings of this research contribute to enhancing the security and integrity of intelligent networks, benefiting various stakeholders, including network administrators, data privacy advocates, and users relying on secure network communication.**

***Keywords*—Poisoning attacks; deep learning; network security; data classification; malicious data**

I. Introduction

In recent years, the widespread use of systems and the data they generate has increased significantly, thanks to rapid advancements in technology [1]. This has led to a surge in the velocity at which data is produced, enabling systems to access and utilize it without requiring detailed programming. As an application of artificial intelligence, machine learning (ML) techniques enable systems to produce meaningful results by learning data on their own [2]. These techniques are extensively applied in cybersecurity, where they are used to identify malware, malicious network traffic, and improper system behavior [3]. Commercial products, such as Exabeam, Fortscale, and E8 Security, leverage these and related ML techniques for cybersecurity.

However, to bypass such detection systems and compromise the security of critical areas by exploiting flaws in ML methodologies, attackers have resorted to deploying adversarial ML techniques. Adversarial ML is a strategy employed in the field of ML that aims to deceive methods using nefarious input in either training or decision-making time.

When building a machine learning algorithm, the first step is to collect data, such as a set of images for developing computer vision applications. Ideally, this data should be collected and labeled in a controlled and secure environment [4]. However, this is a time-consuming and costly operation that not all organizations and individuals can afford. Therefore, they sometimes collect data from the Internet or other untrusted sources. For example, when building security systems, users may download labeled data from external vendors, such as VirusTotal, for malware data annotation.

However, applying ML in Internet of Things (IoT) environments poses unique security challenges, as attackers may tamper with sensors and modify the training data. A poisoning attack, also known as a targeted misclassification or bad behavior assault, allows adversaries to significantly reduce overall performance, introduce backdoors and neural Trojans, and cause targeted misclassification or bad behavior [5].

The study of how adversarial approaches could exploit ML algorithms and the development of effective defenses against their exposure led to the creation of the discipline of adversarial machine learning (AML). AML has been extensively researched in various disciplines, including intrusion detection and picture categorization. However, IoT systems have not been thoroughly studied in this regard.

Although the prevalence of data-driven applications and our growing reliance on networked systems have many advantages, they have also raised serious security concerns [6]. Data poisoning attacks, in which malicious actors inject harmful data into the system to manipulate its behavior and compromise its performance, are a serious security risk. These attacks have the potential to have devastating effects, including incorrect decisions, privacy breaches, and possibly catastrophic outcomes in crucial systems like those that control finance, healthcare, and industry. Effective detection and mitigation of data poisoning attacks may not be possible with current security measures and data classification techniques. Furthermore, since sensitive data is frequently made available

to a single entity or server, centralized approaches to data analysis raise privacy issues. There is a growing interest in investigating decentralized and privacy-preserving techniques, like federated learning, which enables local data analysis while aggregating knowledge globally, to address these problems.

The main goal of this paper is to suggest a novel method for detecting data poisoning attacks with a focus on classifying malicious data using federated and deep learning techniques. The goal of the paper is to tackle the problem of spotting and countering malicious activity in networked systems while protecting data security and privacy. The proposed method enables decentralized data analysis and guarantees that sensitive data is stored and protected locally by using federated learning and a federated adversarial neural network. The analysis of harmful network data is further improved by using BCNN, producing more precise and trustworthy results.

The major contributions of this research study are as follows:

- The study suggests a novel method for identifying data poisoning attacks that focuses on the classification of malicious data. The suggested approach improves the capacity to recognize and counteract malicious activities within the network by utilizing federated and deep learning techniques.
- To analyze network data, spread across various participants, the research introduces the use of a federated adversarial neural network. With this strategy, sensitive information is stored locally, privacy is maintained, and effective analysis of malicious activity is still possible.
- The analyzed data is collected and processed using a cloud module. This federated learning system's participants can communicate with each other easily thanks to the cloud-based approach's efficient data handling.
- In this study, harmful network data is analyzed using a BCNN. The ability of BCNNs to capture model parameter uncertainty allows for more accurate analysis and classification of malicious data.
- The research makes use of real-world datasets, such as the Duchenne Smile Dataset, Product Dataset, and Sentiment Dataset, to show the effectiveness of the suggested attack approach.

The proposed method was chosen based on the distinct advantages of combining federated learning and deep learning approaches for effectively detecting data poisoning attacks in intelligent networks.

Acknowledging the limitations of existing methods in addressing data poisoning attacks in intelligent networks, such as scalability and sensitivity to biased data distributions, emphasizes the need for our proposed approach. By overcoming these constraints, our method offers a compelling alternative to enhance the effectiveness of detecting and mitigating such attacks.

The rest of this paper is organized into four sections. In Section II, we provide a comprehensive review of the existing literature, focusing on data poisoning attacks and their detection. Section III describes the system model used in this research and the architecture of our proposed technique. Section IV presents the results of our experimental analysis, which evaluates the effectiveness of our proposed technique. Finally, in Section V, we summarize our research and its contributions, discuss the implications of our findings, and identify areas for future work.

II. Literature Review

In recent years, the potential threats posed by adversarial machine learning (AML) have been widely studied by researchers. In this section, we provide a comprehensive review of the existing literature on data poisoning attacks and their detection techniques in machine learning.

In the realm of machine learning applications, the data generated for training and testing models is susceptible to manipulation by malicious actors who can gain control over a multitude of devices [7]. Biggio et al. [8] conducted the first systematic poisoning assault against the linear regression method by taking control of many devices, and introduced the TRIM algorithm, which is a more potent method than conventional methods for identifying poisoning spots on training data. Khalid et al. [9] highlighted potential AML attacks and training data poisoning risks, and provided examples of these assaults, including a less damaging training data poisoning attack. The study in [10] proposed a data poisoning attack that modifies labels of labeled data and affects machine learning systems' capacity to categorize data. To guarantee label clearing against this assault, they then put forth a defense method based on the k-nearest neighbors (K-NN) algorithm.

Adversarial attacks are a critical threat to the integrity of machine learning models, and adversaries can manipulate the data generated for these applications by commandeering multiple devices [11]. Within this context, the TRIM algorithm proposed in [12] has been shown to be a more effective and powerful technique for identifying poisoning points in training data, and it was the first systematic attack against the linear regression method. In [13], the author identified potential adversarial machine learning (AML) attacks and risks associated with training data poisoning, including a less harmful attack on training data. Additionally, [14] described a data poisoning attack that modifies the labels of labeled data and impairs the ability of machine learning systems to classify data. To combat this attack, the authors proposed a defense method based on the k-nearest neighbors (K-NN) algorithm. It is highly improbable for training data to represent all possible scenarios, and "adversarial areas" near the decision boundary are particularly vulnerable locations for machine learning models. As a result, adversaries may use trial and error or reverse engineering to uncover "adversarial samples" that are not covered by the training data. Indeed, adversaries can use trial and error or reverse engineering to uncover "adversarial samples" and deceive the model, endangering its integrity. These evasion attacks are experimental and frequently used [15]. For instance, creator [16] utilized a generative network

called Malware-GAN to make ill-disposed malware samples for a black-box classifier, causing the classifier to fail after the assault.

An attack is viewed as causal when an attacker approaches training data and is allowed to harm it. The author in [17] showed that a peculiarity identification strategy on network traffic that had been polluted by refuse traffic infusion enhanced the bogus negative rate to 28 percent for single preparation period harming and to more than 70% for multi-preparing period harming. They also presented a cure strategy that can reject harmful preparation information and is less vulnerable to exceptions for extensive inconsistency detection. The study in [18] proposed the RONI safeguard strategy, which was effective, but had limitations in that it must be tested and trained on spam email data. Furthermore, it could potentially dispose of important information from training data, requiring further examination.

The authors in [19] endeavored to address a limitation that had been encountered by several previous studies. In addition, the authors in [20] proposed a detrimental attack that has the potential to bypass current safeguards with ease. The attack was subjected to testing against a range of hypothetical adversaries, providing valuable insights into its efficacy. The study in[21] also suggested a detrimental attack that utilizes a generative approach to expedite the generation of manipulated data by leveraging the gradient of the model. These advances in adversarial machine learning highlight the urgent need for developing more robust and effective techniques to detect and mitigate these attacks.

In the literature, several studies have proposed techniques to detect data poisoning attacks and other adversarial attacks in machine learning. Data poisoning attack detection tools currently available have some drawbacks and restrictions. Some defense strategies, such as the K-NN algorithm, aren't robust enough to handle complex data poisoning attacks, leaving room for attackers to get around these strategies and successfully poison the training data. When used on contaminated training data, some detection techniques may produce a high percentage of false negatives, failing to accurately detect some cases of data poisoning. Additionally, some current solutions might be effective against the kinds of attacks but fall short when faced with fresh or unexpected attack patterns, which restricts their applicability in real-world situations. Additionally, some proposed techniques are less applicable to a variety of real-world datasets because they rely on data for testing and training, such as spam email data. Additionally, some techniques are impractical for use in real-world applications because of their inability to scale effectively to large datasets or distributed environments. Additionally, although BCNNs demonstrate promise in capturing uncertainty and identifying adversarial samples, their ability to estimate uncertainty may not be accurate enough to defend against all possible data poisoning attacks. Finally, the limited collaborative defense mechanisms used in current methods fail to fully capitalize on the benefits of federated learning and distributed learning for improved detection. Together, these flaws highlight the need for a more thorough and potent method to deal with the constraints imposed on current solutions. The proposed method integrates federated learning, adversarial neural networks, and BCNNs to close this research gap. It hopes to accomplish this by developing a more reliable and scalable method of identifying data poisoning attacks. The proposed method seeks to improve the defense against data poisoning attacks through careful experimental analysis, ultimately advancing the field of adversarial machine learning research.

III. System Model

This section presents a novel technique for detecting data poisoning attacks based on federated and deep learning techniques. The overview of the proposed approach is shown in Fig. 1. By randomly flipping labels in a section of the training dataset, the data poisoning process creates poisoned datasets with varying poisoning rates that include both legitimate and adversarial samples. The method uses an adversarial neural network integrated with a federated learning approach to counter these poisoning attacks. Participants (clients) in this collaborative setting use local datasets to jointly train a global model. As a result of the inclusion of adversarial elements in the learning process, the model is better equipped to fend off poisoning attacks during the federated learning procedure. To further improve model robustness, the proposed technique makes use of BCNNs. The ability of BCNNs to capture prediction uncertainty allows for more accurate detection of potential adversarial samples. Each of the poisoned datasets is used to train a separate BCNN during the phase of model training and evaluation. The effectiveness of both the global model and the BCNNs is then evaluated using results from a shared test dataset. Analyzing the BCNN predictions' levels of uncertainty on the test dataset is a step in the process of detecting data poisoning attacks. The method effectively identifies potential data poisoning attacks by establishing an uncertainty threshold. The detailed description of each step of the proposed approach is presented in the subsequent subsections.

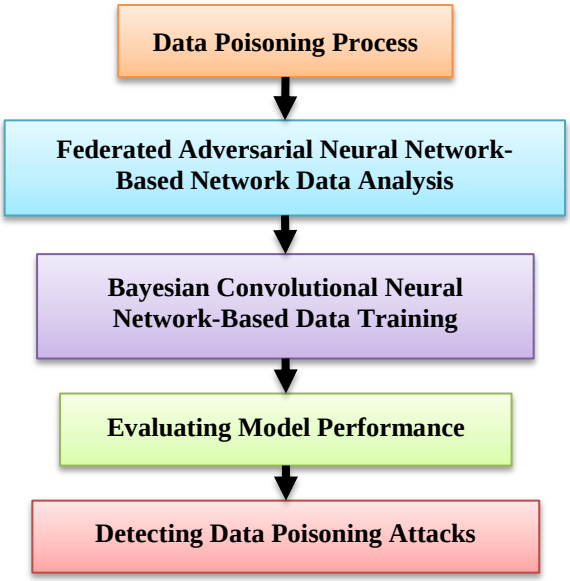


Fig. 1. Overview of the proposed approach.

A. Data Poisoning Process

To simulate the information poisoning attack on data classes, training datasets were created. To prevent a bias towards poisoning a significant amount of either normal data or attack data, normal observations were randomly selected, and most of the normal traffic observations were pooled. After randomizing the data, a Python script was used to determine the number of labels to flip based on a specified rate. To demonstrate the impact of data poisoning on classifiers, the data was poisoned at four rates (rpoison), as outlined in Algorithm 1.

Algorithm 1: Data Poisoning Process

Input: Sanitized Training Dataset (D^5)
Output: Poisoned Training Datasets $\left(D_5^P, D_{10}^P, D_{20}^{P_2}, D_{30}^P\right)$
Steps:
Randomise D^5 observations should not be biased by poisoning either normal or attack observations.
For every rate rpoison of data poisoning [0.05,0.1,0.2,0.3]
Evaluate a number of labels to flip Lpoison=D5*rpoison
Flip Lpoison labels within Ds
End for
Return poisoned training datasets $D_5^P, D_{10}^P, D_{15}^P, D_{30}^P$

B. Federated Adversarial Neural Network-based Network Data Analysis

Participants in federated learning may not always have the same learning objectives or method structures. The central server sends the most recent global model parameters to the chosen participants (mt) at the beginning of each communication round. Then, using the relevant local data, these participants go on to update and train their local models. Each participant uploads their updated model to the central server following the local training process. The central server then averages the models that were uploaded and incorporates the resulting information into the central model. This update procedure is implemented as shown in Eq. (1), ensuring that the central model gains from the group learning of all participants while maintaining the security and privacy of the data. The federated learning approach is a flexible and strong framework that can be applied to various scenarios because it allows participants to maintain their individuality when defining their learning objectives and selecting the best method structures.

$$M_{t+1} = M_t + \frac{1}{m_t}\sum_{k=1}^{m_t} u_t^k \tag{1}$$

In Eq. (1), u_t^k represents the method updates submitted by the k^{th} participant, and Mt represents the current global method at the t^{th} iteration. A federated learning system can achieve high accuracy when users download the same method with the same initialization, which is averaged by the central method with all valid uploads. We now introduce a new method, presented in Eq. (2), for training supervised federated learning models.

$$f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x_i), \psi(x) := \frac{1}{2n}\sum_{i=1}^{n} \|x_i - \bar{x}\|^2 \tag{2}$$

where, $\lambda \geq 0$ is a penalty specification, $x := (x_1, x_2, \dots, x_n) \in \mathbb{R}^{nd}$ are local methods, and $x := \frac{1}{n}\sum_{i=1}^{n} \bar{x}_i$ is average of local methods. Since Eq. (2) has a unique solution, which we designate by Eq. (3), F is strongly convex due to assumptions on fi that we will make.

Here in Eq. (2), $\lambda \geq 0$ is a penalty specification, $x := (x_1, x_2, \dots, x_n) \in \mathbb{R}^{nd}$ represents the local methods, and $x := \frac{1}{n}\sum_{i=1}^{n} \bar{x}_i$ is the average of the local methods. Since Eq. (2) has a unique solution, which we designate as Eq. (3), F is strongly convex due to the assumptions we make on fi.

$$x(\lambda) := (x_1(\lambda), \dots, x_n(\lambda)) \in \mathbb{R}^{\mathrm{nd}} \tag{3}$$

We further let $\bar{x}(\lambda) := \frac{1}{n}\sum_{i=1}^{n} x_i(\lambda)$. We now provide a statement regarding the new formulation's justification. Let's now examine the limit case $\lambda \to \infty$. The ideal local models should be forced to be mutually identical by this limit case while minimising the loss f, according to intuition. This limit situation will specifically be solved using Eq. (4).

We also define $\bar{x}(\lambda) := \frac{1}{n}\sum_{i=1}^{n} x_i(\lambda)$. We now provide a statement regarding the justification for the new formulation. Let us consider the limit case $\lambda \to \infty$. In this limit, the ideal local models should be forced to be identical to each other while minimizing the loss function f, according to intuition. This limit situation is specifically solved using Eq. (4).

$$\min\{f(x): x_1, \dots, x_n \in \mathbb{R}^d, x_1 = x_2 = \cdots = x_n\} \tag{4}$$

Eq. (4) is the equivalent global formulation. Therefore, we define xi(∞) as the optimal solution to Eq. (4) for each i, and let x(∞) := (x1(∞), . . . , xn(∞)).

For vectors $x = (x_1, \dots, x_n) \in \mathbb{R}^{nd}$ and $y = (y_1, \dots, y_n) \in \mathbb{R}^{nd}$, we define the standard inner product and norm as follows: $\langle x, y\rangle := \sum_{i=1}^{n} \langle x_i, y_i\rangle, \| x \|^2 := \sum_{i=1}^{n} \|x_i\|^2$. Note that the separable structure of f implies that $((\nabla f(x))_i = \frac{1}{n}\nabla f_i(x_i))$, i.e., $\nabla f(x) = \frac{1}{n}\left(\nabla f_1(x_1), \nabla f_2(x_2), \dots, \nabla f_n(x_n)\right)$.

Furthermore, note that f is L_f-smooth with with $L_f := \frac{L}{n}$ and μ_f-strongly convex with $\mu_f := \frac{n}{n}$. Clearly, ψ is convex by construction, and it is given that ψ is L_ψ-smooth with $L_\psi = \frac{1}{n}$. We can observe that $(\nabla\psi(x))_i = \frac{1}{n}(x_i - \bar{x})$, which, in turn, implies by Eq. (5), (6), and (7).

$$\psi(x) = \frac{n}{2}\sum_{i=1}^{n} \|(\nabla\psi(x))_i\|^2 = \frac{n}{2} \| \nabla\psi(x) \|^2 \tag{5}$$

$$\psi(x(\lambda)) \leq \frac{f(x(\infty)) - f(x(0))}{\lambda} \tag{6}$$

$$f(x(\lambda)) \leq f(x(\infty)) \tag{7}$$

For every $\lambda > 0$ and $1 \leq I \leq n$, we have by (8):

$$x_i(\lambda) = \bar{x}(\lambda) - \frac{1}{\lambda}\nabla f_i(x_i(\lambda)) \tag{8}$$

We have $\sum_{i=1}^{n} \nabla f_i(x_i(\lambda)) = 0$. By subtracting a multiple of the local gradient from the average model, the best local models Eq. (5) can be obtained. Note that at optimality, the local gradients always add up to zero. This is clearly true for $\lambda = 0$, but it is less clear that this is true for $\lambda = \infty$, or for any $\lambda > 0$.

Let P(z):=$\frac{1}{n}\sum_{i=1}^{n} f_i(z)$. Then, according to Eq. (9), x(∞) is the unique minimizer of P.

$$\| \nabla P(\bar{x}(\lambda)) \|^2 \leq \frac{2L^2}{\lambda}(f(x(\infty)) - f(x(0))) \quad (9)$$

If α ⩽ 2L , we then by (10) have the following.

$$\mathbb{E}\left[\|x^k - x(\lambda)\|^2\right] \leq \left(1 - \frac{\alpha\mu}{n}\right)^k \|x^0 - x(\lambda)\|^2 + \frac{2na\sigma^2}{\mu} \quad (10)$$

where, $\mathcal{L} := \frac{1}{n}\max\left\{\frac{L}{1-p}, \frac{\lambda}{p}\right\}$ and by (11), we have the following.

$$\sigma^2 := \frac{1}{n^2}\sum_{i=1}^{n}\begin{pmatrix} \frac{1}{1-p}\|\nabla f_i(x_i(\lambda))\|^2 \\ +\frac{\lambda^2}{p}\|x_i(\lambda) - \bar{x}(\lambda)\|^2 \end{pmatrix} \quad (11)$$

Let us determine the values of p and α that lead to the fastest rate for pushing the error within a $\left(\mathcal{O}(\varepsilon) + \frac{2na\sigma^2}{\mu}\right)$ - neighborhood of the optimum. In other words, we aim to achieve Eq. (12).

$$\mathbb{E}\left[\|x^k - x(\lambda)\|^2\right] \leq \varepsilon\|x^0 - x(\lambda)\|^2 + \frac{2\mathrm{nn}\,\sigma^2}{\mu} \quad (12)$$

The parameter $p^* = \frac{\lambda}{L+\lambda}$ reduces the predicted number of communications for attaining as well as the number of repetitions. The optimal expected number of communications is $2\frac{L+\lambda}{\mu}\log\frac{1}{\varepsilon}$, while best number of iterations is $2\frac{L+\lambda}{\mu}\log\frac{1}{\varepsilon}$. We employ relativistic average discriminator D_{Ra} to render the output image virtually identical to the original. According to Eq. (13), the objective functions are as follows:

$$\begin{aligned} \mathcal{L}_{Ra,D} &= -\mathbb{E}_x[\log(D_{Ra}(x))] \\ &\quad - \mathbb{E}_{x,v,}[\log(1 - D_{Ra}(G(x,v,c)))] \\ \mathcal{L}_{Ra\_G} &= -\mathbb{E}_{x,0,\mathcal{C}}[\log(D_{Ra}(G(x,v,c)))] \\ &\quad - \mathbb{E}_x[\log(1 - D_{Ra}(x))] \\ D_{Ra}(x) &= \mathrm{sigmoid}\big(H(x) - \mathbb{E}_{x,v,c}[H(G(x,v,c))]\big) \\ D_{Ra}(G(x,v,c)) &= \mathrm{sigmoid}\begin{pmatrix} H\big(G(x,v,c)\big) \\ -\mathbb{E}_x[H(x)] \end{pmatrix} \end{aligned} \quad (13)$$

The parameter $p^* = \frac{\lambda}{L+\lambda}$ reduces the predicted number of communications required to achieve the desired accuracy, as well as the number of repetitions needed. The optimal expected number of communications is $2\frac{L+\lambda}{\mu}\log\frac{1}{\varepsilon}$, while the optimal number of iterations is $2\frac{L+\lambda}{\mu}\log\frac{1}{\varepsilon}$.

To make the output image virtually identical to the original, we employ the relativistic average discriminator D_{Ra} . According to Eq. (13), the objective functions are as follows:

The output of the non-changed layer is denoted as H(•). The probability that certifies the real image as genuine is higher than the probability that certifies the generated image as genuine. This can be improved by minimizing the loss function $\mathcal{L}_{Ra,D}$.

To further reduce the loss, we subject the generator to a cycle consistency loss, which is described by Eq. (14) as follows:

$$\mathcal{L}_{cyc} = \mathbb{E}_{x,v,c}[\| x - G(G(x,v,c),v,1-c) \|_1] \quad (14)$$

To identify the source of the image, we add a helper classifier called D_{ind} on top of the discriminator network. According to Eq. (15), the loss function for the image attribution model is as follows:

$$\begin{aligned} \mathcal{L}_{\text{ind}} = \\ -\mathbb{E}_{x,t}\big[\log\big(D_{\text{ind}}(t = 01 \mid x)\big)\big] - \quad \mathbb{E}_{x,v,c,t}\big[\log\big(D_{\text{ind}}(t = 01 \mid \\ G(x,v,c))\big)\big] \end{aligned} \quad (15)$$

The picture producing model fundamentally affects the unraveling organization (c) since it is a common organization, and picture interpretation strategy utilizes essentially less examples than the picture age model does. We integrate the accompanying matched antagonistic misfortune condition (16) to more likely guarantee the fitting of the picture interpretation model:

Since the image generation model is a shared network, it significantly affects the decoding network (c). Moreover, the image attribution method uses significantly fewer samples than the image generation model. To better ensure the fitting of the image attribution model, we integrate the following paired adversarial loss condition as shown in Eq. (16).

$$\begin{aligned} \mathcal{L}_{pis} = -\mathbb{E}_{x_0,x_v}\left[\log\left(D_{pis}(x_0,x_v)\right)\right] - \mathbb{E}_{x,v\mathcal{C}}\big[\log\big(1 - \\ D_{pis}(x,G(x,v,c))\big)\big] \end{aligned} \quad (16)$$

In this scenario, D_{pis} is used to determine if two images belong to the same class. Our objective is to translate x_0 into an output image y that contains variation v, for input image x_0 and action (v, c = 01). Moreover, our goal is to remove variation v from input image xv using the action (v, c = 10).

To achieve this, we add an additional classifier called D_{var} on top of the discriminator network to identify different types of image variations. The classification loss during training of the discriminator network is given by Eq. (17).

$$\mathcal{L}^r_{\text{var}} = -\mathbb{E}_{x,v}[\log(D_{var}(v \mid x))] \quad (17)$$

Discriminator network may categorize real image x into variant type v by minimizing formula. Classification loss during training the generator network is as shown in Eq. (18)

The discriminator network can categorize the real image x into variant type v by minimizing the formula mentioned above. The classification loss during training of the generator network is given by Eq. (18).

$$\mathcal{L}^f_{\text{var}} = -\mathbb{E}_{x,v,c}[\log(D_{var}(v \mid x))] \quad (18)$$

The first condition in Eq. (18) states that the image produced by adding variation v to the input image x_0 should be accurately classified into class v. The second condition states that the image produced by removing variation v from the paired image xv should be classified into class v.

The values of x_m range from 0 to 1. The following Eq. (19) can be used to obtain the final output image.

$$x_{\text{out}} = x + (x_t - x) \odot x_m \tag{19}$$

$\odot$ element-wise product is located. By using Eq. (20), we add the next restriction for the mask x_m:

$$\mathcal{L}_{\text{mask}} = \left(\frac{1}{W}\sum_k |x_m[k]|\right)^2 \tag{20}$$

Here, W represents the number of pixels, and $x_m[k]$ refers to the k^{th} pixel of x_m. The formula shown above encourages minimizing alterations to the source image. Based on the foregoing discussion, the overall loss of the image translation model is given by Eq. (21):

$$\mathcal{L}_D = \mathcal{L}_{\text{Ra\_}D} + \lambda_{\text{pis}}\mathcal{L}_{\text{pis}} + \lambda_{\text{var}}\mathcal{L}_{\text{var}}^r + \lambda_{\text{ind}}\mathcal{L}_{\text{ind}}$$
$$\mathcal{L}_G = \mathcal{L}_{Ra\_G} - \lambda_{pis}\mathcal{L}_{\text{pis}} + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}} + \lambda_{\text{var}}\mathcal{L}_{\text{tar}}^f + \lambda_{\text{ind}}\mathcal{L}_{\text{ind}} + \lambda_{\text{mask}}\mathcal{L}_{\text{mask}} \tag{21}$$

The hyperparameters $\lambda_{\text{pis}}, \lambda_{\text{var}}, \lambda_{\text{ind}}, \lambda_{\text{cyc}}$, and λ_{mask} control the relative significance of each term in Eq. (21) as outlined in Algorithm 2.

Algorithm 2: FANN

Input: K clients are indexed by k, C is client fraction, the T communication rounds are indexed by t, B is local minibatch size, E is number of local epochs, and η is learning rate, PGD Attack $A_{s,\epsilon,\alpha}$: where s, ϵ, α are number of PGD steps, perturbation ball size, step size, r is the adversarial ratio, q is the scale factor.

Output: The global model θ.

On server

Initialize θ_0

For every round $t = 1,2, \ldots \ldots T$ do

$$m \leftarrow \max(1, CK)$$
$$S_t \leftarrow (random\ set\ of\ m\ clients)$$

For every client $k \in S_t$ in parallel do

$$\theta_{t+1}^k \leftarrow client\ update\ (k, \theta_t)$$

End for

$$\theta_{t+1}^k \leftarrow FedWAvg\ (\theta_t, \{\theta_{t+1}^k\} k \in S_t)$$

End for

Return θ_{t+1}

Client update (k,θ)

$$E \leftarrow split\ the\ training\ data\ into\ batches\ of\ size\ B$$

For every local epoch I from 1 to E do

For batch $b \epsilon B$ do

$$n_{adv} \leftarrow r.B$$
$$b_{adv} \leftarrow (random\ set\ \in b\ of\ n_{adv}\ samples)$$
$$b_{nat} \leftarrow (set\ of\ B) - n_{adv} samples$$
$$b \leftarrow b_{bat} U b_{adv}$$

End for

End for

Return θ

C. Bayesian Convolutional Neural Network-based Data Training

BCNNs are a type of neural network that combines the CNN architecture and Bayesian inference principles to model uncertainty in deep learning tasks. In contrast to conventional CNNs, which provide point estimates of the model parameters, BCNNs estimate the model posterior distribution over the parameters, providing a principled method for dealing with model uncertainty. This feature is especially helpful when there is little or noisy data available, enabling more accurate predictions. BCNNs also allow for incorporating prior information and hypotheses, which can improve model performance in challenging real-world datasets. Additionally, BCNNs provide a natural method for model averaging, improving the generalizability of the model. In our work on identifying data poisoning attacks, BCNNs' uncertainty estimation is essential, as it can highlight areas of high ambiguity and possible adversarial inputs, resulting in a more accurate identification of such attacks.

Bayesian neural networks train a model by inferring the model posterior. However, accurate inference of the model posterior is computationally demanding, and even for moderately sized models, it can become intractable. Therefore, the model posterior is usually approximated. One popular and successful method for approximating the model posterior is variational inference. Fig. 2 provides an overview of the BCNN Architecture.

The BCNN architecture process is shown in Fig. 3. The "Start" symbol marks the beginning of the process at the top. Taking input data, which stand for the input set and the corresponding output set, respectively, is the first step. The next step in the flowchart is the "Feature Extraction" module. Utilizing techniques like convolution, non-linear transformations (relu), max-pooling, and local normalization, features are in this case extracted from the input data. The "Feature Selection" module is the next step in the flowchart after feature extraction. To further hone the extracted features, additional feature selection is carried out in this step using non-linear transformations (relu). The "Prediction" module is the next step in the process, where the final output probabilities are computed.

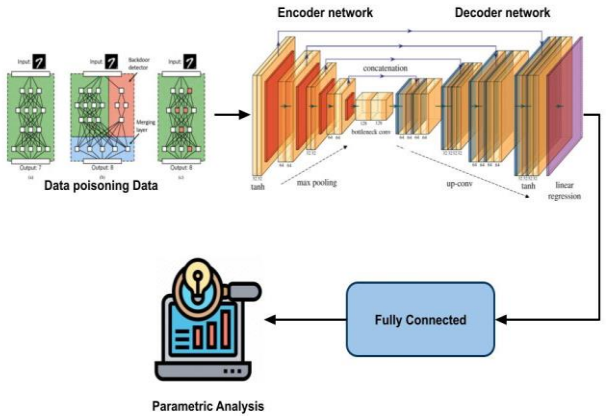


Fig. 2. Overview of the BCNN architecture.

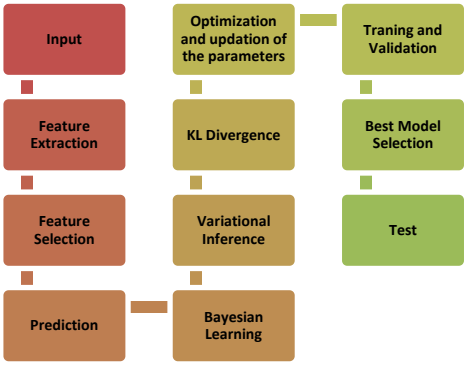


Fig. 3. Flowchart of BCNN architecture process.

The softmax operation provides a probability distribution for each output class C. The flowchart then moves to the "Bayesian Learning" phase, where variational inference is used to obtain the model posterior by approximating it with the variational distribution. The "KL Divergence" step computes the KL divergence, which is reduced through optimization of the model parameters W and b to increase the log evidence lower bound. The "Training and Validation" phase trains the model on the training set and assesses its performance on the validation set after each epoch. The "Best Model Selection" step chooses the model with the best validation performance. The "Test" step tests the chosen model on the test set to evaluate its final performance metrics. The "End" symbol marks the end of the process. The detailed process is presented in this section.

Given the input set $X = x_1, x_2, \ldots, x_N$ and the corresponding output set $y = y_1, y_2, \ldots, y_N$, the function $f(X) = y$ estimates the output y from the inputs X. Bayesian learning provides a principled approach to obtain the model posterior $p(f|X, y)$. To calculate the posterior, two components are required. First, a prior distribution $p(f)$ that captures a prior belief about the estimator functions. Second, a likelihood function $p(y|f, X)$ that indicates how likely it is for the model f to predict the output y given the observations X. More specifically, given an unseen data point $(x*, y*)$, the posterior is obtained by integrating over all possible estimator functions f that are parametric models with a parameter set θ, as shown in Eq. (22):

$$p(y^* \mid \mathrm{x}^*, \mathrm{X}, \mathrm{y}) = \int p(y^* \mid f)p(f \mid \mathrm{x}^*, \mathrm{X}, \mathrm{y})df = \int p(y^* \mid f)p(f \mid \mathrm{x}^*, \theta)p(\theta \mid \mathrm{X}, \mathrm{y})df d\theta \quad (22)$$

The integral in Eq. (22) is intractable because the distribution $p(\theta|X, y)$ is intractable. Therefore, the variational approach is to approximate $p(\theta|X, y)$ with a variational distribution $q(\theta)$. The candidate $q(\theta)$ should be as similar as possible to the original intractable distribution. The similarity between $p(\theta|X, y)$ and q(\theta) can be measured by the Kullback-Leibler (KL) divergence. Reducing the KL divergence is equivalent to increasing the log evidence lower bound based on the parameter set θ, as shown in Eq. (23):

$$\mathrm{KL}_{\mathrm{V1}} = q(\theta)p(F \mid \mathrm{X}, \theta)\log_p \int (\mathrm{y} \mid F)\, dF d\theta - \mathrm{KL}(q(\theta) \parallel p(\theta)) \quad (23)$$

Maximizing the KL divergence results in a variational distribution that approximates the posterior. The approximation $q(\theta)$ simplifies Eq. (23) to Eq. (24).

$$q(y^* \mid \mathrm{x}^*) = \int p(y^* \mid f)p(f \mid \mathrm{x}^*, \theta)q(\theta)df d\theta \quad (24)$$

During inference, the network parameters θ are sampled from $q(\theta)$. The feature extraction module at stage l, denoted as $g^{(l)}$, extracts the features $H^{(n)}$ as specified by Eq. (25).

$$H^{(n)} = g^{(l)}\left(H^{(l-1)}; W^{(l)}, b^{(l)}\right) = \text{normalize}\left(\text{pool}\left(\text{relu}\left(W^{(l)} * H^{(l-1)} + b^{(l)}\right)\right)\right) \quad (25)$$

The $*$ operator denotes convolution, which is one of the specific processes that go into feature extraction, along with non-linear transformations, max-pooling, and local normalization. After the convolution operation, a dot product is computed, which is followed by a non-linear transformation specified in Eq. (26) within the feature selection module f (l).

$$H^{(n)} = f^{(l)}\left(H^{(l-1)}; W^{(l)}, b^{(l)}\right) = \left(\text{relu}\left(W^{(l)} \cdot H^{(l-1)} + b^{(n)}\right)\right) \quad (26)$$

In Eq. (26), $H^{(l-1)}$ denotes the activation of the (l - 1) th hidden layer, and (.) denotes the dot product. To provide a probability distribution over every output class C, as represented in Eq. (27), the softmax operation is used as the final step in the prediction module.

$$p(C \mid X; W, b) = \text{softmax}\left(W^{(n)} \cdot H^{(l-1)} + b^{(n)}\right) \quad (27)$$

The DCNN model architecture is constructed by stacking the feature extraction, selection, and prediction modules, as shown in Eq. (28).

$$p(C \mid X; W, b) = \text{softmax}\left(f^{(5)}\left(f^{(4)}\left(g^{(3)}\left(g^{(2)}\left(g^{(1)}(X)\right)\right)\right)\right)\right) \quad (28)$$

During this optimization, local connections and weight sharing are implemented, resulting in a reduction in the number of parameters. Eq. (29) and Eq. (30) can be used to define 1-D and 2-D convolutional operations in a CNN, respectively:

$$\mathrm{o}_{i,k} = (\mathrm{x} * \mathrm{v})_{i,k} = \sum_{l,m} \mathrm{x}_{(i-1)s+m,l} \mathrm{V}_{m,l,k,} \quad (29)$$

$$\mathrm{O}_{i,j,k} = (\mathrm{X} * \mathrm{V})_{i,j,k} = \sum_{l,m,n} \mathrm{X}_{(i-1)s+m,(j-1)s+n,l} \mathrm{V}_{m,n,l,k} \quad (30)$$

In Eq. (29), x is the 1-D input, v is the convolutional kernel, and o is the output. Similarly, V and O are the corresponding kernel and output in Eq. (30), where X is the input of the 2-D convolutional operation. The number of data points skipped between two convolutional operations is referred to as the stride, denoted by s.

The data is split into training, validation, and test sets. The method is then trained on the training set, and after every epoch, the method is validated. After training, the model with the best validation kappa score is selected and evaluated on the test set.

Deep neural networks are capable of extracting features from raw input data. However, the quality and quantity of the training data are important requirements for achieving good performance. When the available data is limited, the network may not converge. In such cases, a pre-processing step can be applied to eliminate redundancy and reduce the feature dimensionality, which can help the network converge.

Consider a deep learning method with a model specification W. The training dataset consists of M samples, denoted as $S = (x1, y1), (x2, y2), (xM, yM)$, and so on. The model parameters are calculated using the Bayes formula, as shown in Eq. (31):

$$p(\mathrm{W} \mid \mathrm{S}) = \frac{p(\mathrm{S}|\mathrm{W})p(\mathrm{W})}{p(\mathrm{S})} = \frac{p(\mathrm{S}|\mathrm{W})p(\mathrm{W})}{\int_{\mathrm{W}} p(\mathrm{S}|\mathrm{W})p(\mathrm{W})\mathrm{dW}} \quad (31)$$

The prior distribution, denoted by $p(W)$, is based on an assumption, knowledge from the past, or experience. The likelihood function is $p(S|W)$, where $p(S)$ denotes the distribution of the training samples, and the predicted distribution of the model specification W is $p(W|S)$. However, the Bayes formula cannot be used directly to obtain the specification evaluation because it is challenging to calculate p(S). To address this issue, a new distribution, $q(W)$, is developed to approximate $p(\theta|S)$. The idea of Kullback-Leibler (KL) divergence can be used to calculate the difference between $q(W)$, and $p(W|S)$. Eq. (32) is used to express the KL divergence.

$$\begin{aligned} \mathrm{D_{KL}}\big(q(\theta) \parallel p(\theta \mid \mathrm{S})\big) &= \int_{\mathrm{W}} q(\mathrm{W}) \log \frac{q(\mathrm{W})}{p(\mathrm{WS})} \mathrm{dW} \\ &= \int_{\Theta} q(\mathrm{W}) \log \frac{q(\theta)\int_{\mathrm{W}} p(\mathrm{SW})p(\mathrm{W})\mathrm{dW}}{p(\mathrm{SW})p(\mathrm{W})} \mathrm{dW} \\ &= \int_{\Theta} q(\mathrm{W})p(\mathrm{S})\mathrm{dW} - \int_{\Theta} q(\mathrm{W}) \log \frac{p(\mathrm{SW})p(\theta)}{q(\mathrm{W})} \mathrm{dW} \quad (32) \end{aligned}$$

The goal of variational inference is to maximize the second term on the left-hand side of (33), which corresponds to $q(\theta)$, while minimizing the KL divergence.

$$\begin{aligned} q^*(\mathrm{W}) &= \underset{q(\mathrm{W})}{\operatorname{argmin}} \, \mathrm{DKL}\big(q(\mathrm{W}) \parallel p(\mathrm{W} \mid \mathrm{S})\big) \\ &= \underset{q(\mathrm{W})}{\operatorname{argmax}} \int_{\mathrm{W}} q(\mathrm{W}) \log \frac{p(\mathrm{S}|\mathrm{W})p(\mathrm{W})}{q(\mathrm{W})} \mathrm{dW} \quad (33) \end{aligned}$$

Assuming that q(W) is a joint Gaussian distribution and that each specification W_i in the specification matrix W follows an independent Gaussian distribution allows us to transform the variational problem into an optimization problem, as shown in Eq. (34):

$$(\mathrm{W}) = N(\mathrm{W}, \mu, \sigma^2) = \prod_i^{n_{W_i}} N(W_i, \mu_i, \sigma_i^2) \quad (34)$$

In Eq. (34), the mean value matrix and standard deviation are denoted by μ and σ, respectively. Determining the optimal values of the mean and standard deviation matrices, as shown in Eq. (35), will yield the ideal distribution $q(W)$:

$$\begin{aligned} \mu^*, \sigma^* = \underset{\mu_{1,\sigma},\sigma}{\operatorname{argmax}} & \sum_{k=1}^{n_\rho} \mathbb{E}_{q(\sigma_2 \varepsilon_k + u_k)}\big[\log\big(p(\sigma_k \varepsilon_k + u_k)\big)\big] \\ & + \sum_{k=1}^{n_\theta} \mathbb{E}_{q_k(\sigma_k \varepsilon_k + u_k)}\big[\log\big(q_k(\sigma_k \varepsilon_k + u_k)\big)\big] \\ & + \frac{N_N}{N} \sum_{j=1}^{N/N_B} N_B^N \sum_{i=1}^{N_n} \mathbb{E}_{q(\varepsilon)}\big[\log\big(p(\mathbf{y}_i \mid \mathbf{x}_i, \mu, \sigma, \varepsilon)\big)\big] \quad (35) \end{aligned}$$

IV. Experimental Analysis

In our evaluation, we present the compelling results of our proposed method for detecting data poisoning attacks in intelligent networks. Our approach consistently outperformed the referenced methods, achieving a significantly higher detection rate. The visualizations, including precision-recall curves and confusion matrices, vividly illustrate the superior performance and robustness of our method. These results provide strong evidence of the effectiveness and practical relevance of our approach in bolstering network security against data poisoning attacks.

A. Experimental Setup

The purpose of the experimental setup is to assess how well the suggested attack and defense strategies work. The Duchenne Smile Dataset, Product Dataset, and Sentiment Dataset are three real-world datasets used in the evaluation. Using customary cross-validation methods, these datasets are preprocessed and divided into training, testing, and validation sets. The suggested strategy is put into practice for the attack method using Python's NumPy and sklearn libraries. The datasets are subjected to the attack to evaluate its potential to undermine network security and jeopardize data classification. The suggested method is also put into practice for the defense method using Python's sklearn and NumPy libraries. The defense mechanism is applied to the datasets to test its efficacy in defending the network against threats and enhancing the accuracy and dependability of data classification. A comparison between the proposed methods and current methods, like K-Nearest Neighbors (KNN) and Malware-GAN, is done to ensure thorough evaluation. Throughput, Quality-of-Service (QoS), transmission rate, Root Mean Square Error (RMSE), accuracy, recall, precision, and malicious data analysis are just a few of the performance metrics that are measured and compared.

B. Dataset Description

This section provides a brief overview of the real-world datasets used in our experimental analysis in this section. These datasets are used to assess how well the attack and defense strategies we've suggested improve network security and data classification.

Duchenne Smile Dataset: The aim of this dataset is to determine whether a facial image contains a Duchenne or non-Duchenne smile. The task-creation and label-collection processes were performed using the Amazon Mechanical Turk platform. The dataset consists of 2,134 entries, with 64 regular employees producing 17,729 labels.

Product Dataset: The objective of this dataset is to determine whether two products are the same for each item in the dataset, which comprises pairs of items with descriptions. Participating employees were required to determine whether the two descriptions apply to the same item before providing their labels. This dataset contains 8,315 items, with 176 average workers providing 24,945 labels in total.

Sentiment Dataset: This dataset consists of a tweet about a specific firm for each item. The participating employees were tasked with determining whether the sentiment expressed in the tweet is favorable or unfavorable to the business. We created

1,000 objects using the AMT platform and collected labels from 85 regular workers. This dataset contains a total of 20,000 labels.

C. Performance Matrices

We use a set of performance metrics that cover various facets of the models' performance to assess the efficacy of our suggested attack and defense strategies.

Network throughput refers to the amount of data that can be successfully transported over a network in a certain period. It is measured in bits per second (bps) and can also refer to data packets per time slot or packets per second (pps). The aggregate throughput, also known as system throughput, is the total data rates sent to all network endpoints.

Quality-of-service (QoS) is a critical issue in wireless sensor applications, and each application has specific QoS requirements. Accuracy is one specification used to assess classification models, and it refers to the percentage of correct predictions made by a method. Recall and precision are measures of quantity and quality, respectively. A higher recall indicates that the method provides more relevant results, while a higher precision indicates that the method provides more relevant results than irrelevant ones. Precision is evaluated by dividing the total number of true positives (TP) by the total number of TP plus false positives (FP), while recall is calculated as the product of the number of TP divided by the sum of the TP and false negatives (FN).

Root means square error (RMSE) is a commonly used method for assessing the accuracy of forecasts, and it measures the Euclidean distance between the measured true values and forecasts. The standard deviation of residuals is also known as RMSE.

D. Comparative Analysis

Table I presents a comparative analysis between our proposed method and existing methods, based on network security and data classification. The analysis considers various parameters, including throughput, QoS, transmission rate, RMSE, accuracy, recall, precision, and malicious data analysis. The datasets analyzed include the Duchenne Smile Dataset, Product Dataset, and Sentiment Dataset.

Fig. 4 represents a comparative analysis between our proposed method and existing methods for network security. The graph shows that our proposed technique achieved a throughput of 96%, QoS of 83%, transmission rate of 89%, RMSE of 61%, accuracy of 95%, recall of 69%, precision of 79%, and malicious data analysis of 75%. In comparison, the KNN method achieved a throughput of 86%, QoS of 77%, transmission rate of 85%, RMSE of 55%, accuracy of 91%, recall of 65%, precision of 72%, and malicious data analysis of 69%, while the Malware-GAN method obtained a throughput of 94%, QoS of 79%, transmission rate of 88%, RMSE of 59%, accuracy of 93%, recall of 66%, precision of 75%, and malicious data analysis of 73%.

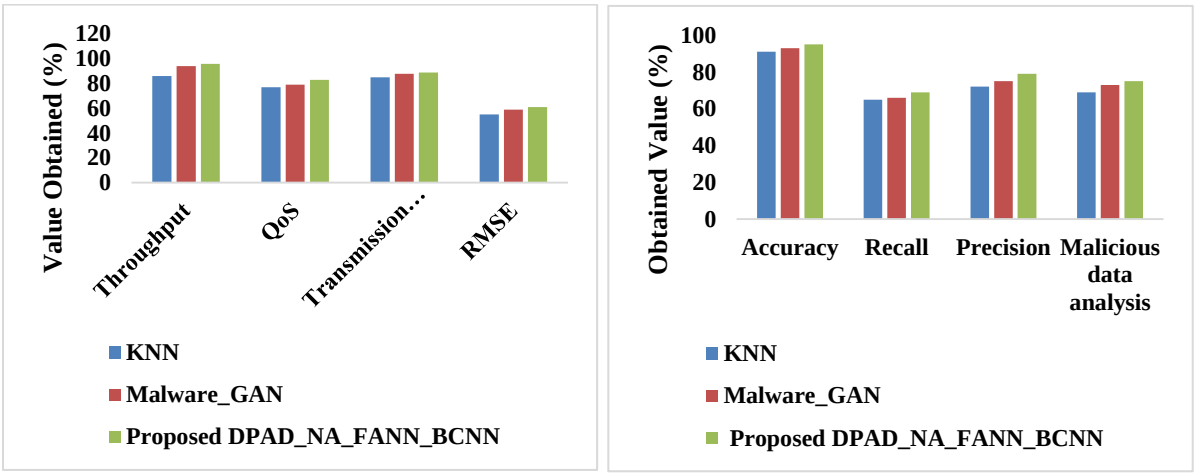
Significant performance differences are found when comparing the proposed method to the current network security methods. Compared to the KNN and Malware-GAN methods, our suggested technique outperformed them in all performance metrics. These findings suggest that when compared to the KNN and Malware-GAN methods, the proposed method is more effective and reliable in the context of network security analysis. The proposed method's efficiency in addressing network security issues is demonstrated by the higher throughput and transmission rate, better accuracy, and MDA.

Fig. 5 provides an analysis based on data classification between our proposed method and existing techniques. The graph shows that our proposed technique achieved a throughput of 95%, QoS of 85%, transmission rate of 93%, RMSE of 69%, accuracy of 96%, recall of 75%, precision of 85%, and malicious data analysis of 86%. In comparison, the KNN method achieved a throughput of 89%, QoS of 81%, transmission rate of 91%, RMSE of 63%, accuracy of 92%, recall of 71%, precision of 81%, and malicious data analysis of 79%, while the Malware-GAN method obtained a throughput of 95%, QoS of 85%, transmission rate of 93%, RMSE of 69%, accuracy of 96%, recall of 75%, precision of 85%, and malicious data analysis of 86%.

The proposed approach performs better than existing techniques for data classification, as shown by the comparative analysis between them. The outcomes show that the suggested method outperforms the KNN and Malware-GAN methods across the board. According to these findings, the proposed method performs data classification more effectively and efficiently than the KNN and Malware-GAN methods. The superiority of the suggested technique in handling data classification tasks is demonstrated by the higher throughput, transmission rate, accuracy, and MDA, along with better QoS and recall.

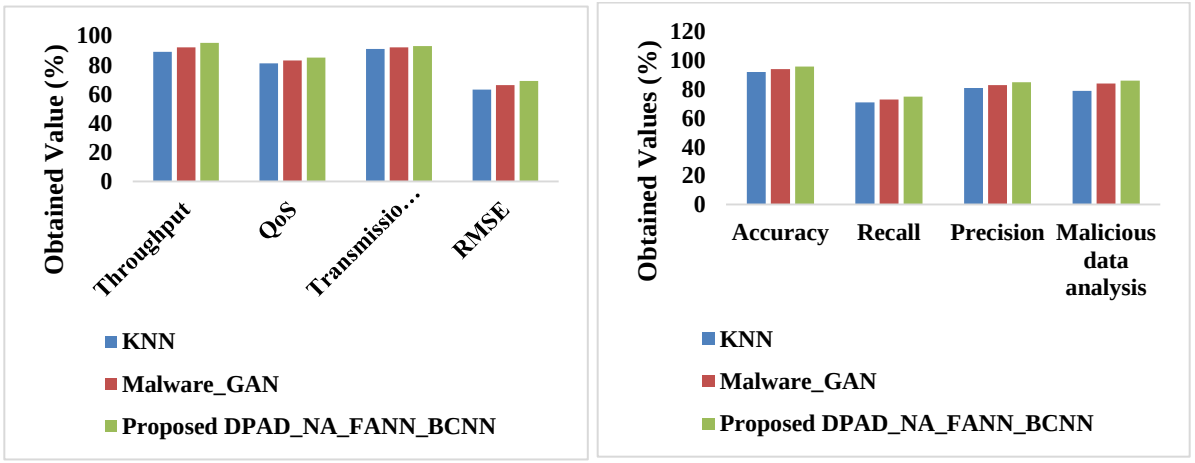
TABLE I. TABLE TYPE COMPARATIVE ANALYSIS OF PROPOSED AND EXISTING METHOD BASED ON NETWORK SECURITY AND DATA CLASSIFICATION

Techniques	Throughput	QoS	Transmission Rate	RMSE	Accuracy	Recall	Precision	Malicious Data Analysis
Case 1: Network security								
KNN	86	77	85	55	91	65	72	69
Malware_GAN	94	79	88	59	93	66	75	73
DPAD_NA_FANN_BCNN	96	83	89	61	95	69	79	75
Case 2: Data classification								
KNN	89	81	91	63	92	71	81	79
Malware_GAN	92	83	92	66	94	73	83	84
DPAD_NA_FANN_BCNN	95	85	93	69	96	75	85	86



(a) (b)

Fig. 4. Comparative analysis between proposed and existing method based on network security analysis: (a) comparison in terms of RMSE, transmission rate, QoS, and throughput (b) comparison in terms of MDA, precision, recall, and accuracy.



(a) (b)

Fig. 5. Comparative analysis between proposed and existing technique based on data classification: (a) comparison in terms of RMSE, transmission rate, QoS, and throughput; (b) comparison in terms of MDA, precision, recall, and accuracy.

We observe that our experimental analysis demonstrates the effectiveness of our proposed method in achieving high levels of network security and data classification performance. Our proposed technique outperforms the existing methods in terms of various parameters, including throughput, QoS, transmission rate, RMSE, accuracy, recall, precision, and malicious data analysis. The results indicate that our proposed method can significantly enhance the security and performance of wireless sensor networks.

Furthermore, the analysis of the Duchenne Smile Dataset, Product Dataset, and Sentiment Dataset reveals that our proposed method is robust and can be applied to various types of datasets. The high levels of accuracy, recall, and precision achieved by our proposed method indicate its potential for use in real-world applications, including in industries such as healthcare, e-commerce, and social media.

The results demonstrate the potential of our proposed method in enhancing the security and performance of wireless sensor networks and its ability to provide accurate and relevant results for data classification tasks. Further research can explore the use of our proposed method for other types of datasets and in different settings to evaluate its robustness and scalability.

The findings of this study highlight the effectiveness of the proposed technique in detecting and mitigating data poisoning attacks in intelligent networks. The achieved high levels of network security and accuracy in data detection demonstrate its practical value for network administrators, ensuring the protection of sensitive data and system integrity. The successful application of the technique contributes to advancements in network security and data analytics, while future research can focus on scalability and addressing potential vulnerabilities to further enhance its robustness. Overall, this study provides valuable insights for the implementation of secure and privacy-preserving intelligent networks.

V. Conclusion

This research proposes a novel technique for detecting data poisoning attacks based on deep learning, which combines

federated learning with adversarial neural networks. The proposed technique utilizes a Bayesian convolutional neural network to train the data analyzed by federated learning for detecting the presence of data poisoning attacks in the network. The experimental analysis is carried out based on network security and data classification, utilizing real-world datasets, such as the Duchenne Smile Dataset, Product Dataset, and Sentiment Dataset. The proposed technique outperforms the existing models in the literature, achieving high levels of performance in various parameters, including throughput, QoS, transmission rate, RMSE, accuracy, recall, precision, and malicious data analysis.

The results of this research indicate that the proposed technique can significantly enhance the security and performance of wireless sensor networks, contributing to a deeper understanding of data poisoning attacks and detection strategies in real-world contexts. Furthermore, this research contributes to the advancement of more effective outlier detection methods across a wider range of applications. Future work must address the challenge of preventing such attacks in a strengthened federated learning environment.

Overall, the proposed technique offers a promising approach to the detection of data poisoning attacks, which have become increasingly prevalent in wireless sensor networks. This research opens new avenues for future research in the field of wireless sensor networks, and the proposed technique holds potential for use in various industries, including healthcare, e-commerce, and social media.

REFERENCES

[1] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," arXiv preprint arXiv:2002.05516, 2020.

[2] Y. Ding, Z. Tang and F. Wang, "Single-sample face recognition based on shared generative adversarial network," Mathematics, vol. 10, no. 5, pp. 752-758, 2022.

[3] T. Wang, H. Li, M. Noori, R. Ghiasi, S. C. Kuok et al., "Probabilistic seismic response prediction of three-dimensional structures based on bayesian convolutional neural network," Sensors, vol. 22, no. 10, pp. 3775-3790, 2022.

[4] G. Zhao, F. Liu, J. A. Oler, M. E. Meyerand, N. H. Kalin et al., "Bayesian convolutional neural network based MRI brain extraction on nonhuman primates," Neuroimage, vol. 175, no. 1, pp. 32-44, 2018.

[5] M. Joshaghani, A. Davari, F. N. Hatamian, A. Maier and C. Riess, "Bayesian convolutional neural networks for limited data hyperspectral remote sensing image classification," arXiv preprint arXiv:2205.09250, 2022.

[6] U. Zafar, M. Ghafoor, T. Zia, G. Ahmed, A. Latif et al., "Face recognition with bayesian convolutional networks for robust surveillance systems," EURASIP Journal on Image and Video Processing, vol. 2019, no. 1, pp. 1-10, 2019.

[7] I. M. Ahmed and M. Y. Kashmoola, "Threats on machine learning technique by data poisoning attack: a survey," in International Conference on Advances in Cyber Security, Penang, Malaysia, pp. 586-600, Springer, Singapore, August 2021.

[8] F. A. Yerlikaya and Ş. Bahtiyar, "Data poisoning attacks against machine learning algorithms," Expert Systems with Applications, vol. 189, no. 1, 118101, 2022.

[9] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild et al., "Dataset security for machine learning: data poisoning, backdoor attacks, and defences," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 2, pp. 1563-1580, 2022.

[10] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu et al., "Data poisoning attacks to deep learning based recommender systems," arXiv preprint arXiv:2101.02644, 2021.

[11] A. E. Cinà, K. Grosse, A. Demontis, B. Biggio, F. Roli et al., "Machine learning security against data poisoning: are we there yet?" arXiv preprint arXiv:2204.05986, 2022.

[12] L. Verde, F. Marulli and S. Marrone, "Exploring the impact of data poisoning attacks on machine learning model reliability," Procedia Computer Science, vol. 192, no. 2, pp. 2624-2632, 2021.

[13] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-pois: An attack-agnostic defense against data poisoning attacks," IEEE Transactions on Information Forensics and Security, vol. 16, no. 1, pp. 3412-3425, 2021.

[14] F. Nuding and R. Mayer, "Data poisoning in sequential and parallel federated learning," in Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics, Baltimore, MD, USA, pp. 24-34, 2022.

[15] H. Liu, D. Li and Y. Li, "Poisonous label attack: Black-box data poisoning attack with enhanced conditional DCGAN," Neural Processing Letters, vol. 53, no. 6, pp. 4117-4142, 2021.

[16] A. Milakovic and R. Mayer, "Combining defences against data-poisoning based backdoor attacks on neural networks," In IFIP Annual Conference on Data and Applications Security and Privacy, Newark, NJ, USA, pp. 28-47, Springer, Cham, 2022.

[17] S. Bhattacharjee, M. J. Islam and S. Abedzadeh, "Robust anomaly based attack detection in smart grids under data poisoning attacks," In Proceedings of the 8th ACM on Cyber-Physical System Security Workshop, Nagasaki, Japan, pp. 3-14, 2022.

[18] F. Razmi and L. Xiong, "Classification auto-encoder based detector against diverse data poisoning attacks," arXiv preprint arXiv:2108.04206, 2021.

[19] S. Farhadkhani, R. Guerraoui and O. Villemaud, "An equivalence between data poisoning and byzantine gradient attacks," In International Conference on Machine Learning, Honolulu, Hawai, pp. 6284-6323, PMLR, June 2022.

[20] Y. Mao, X. Yuan, X. Zhao and S. Zhong, "Romoa: Robust model aggregation for the resistance of federated learning to model poisoning attacks," in European Symposium on Research in Computer Security, pp. 476-496, Springer, Cham, October 2021.

[21] H. Zhang, J. Gao and L. Su, "Data poisoning attacks against outcome interpretations of predictive models," in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Virtual Event Singapore, pp. 2165-2173, 2021.