Toward Generative Adversarial Networks for the Industrial Internet of Things

Cheng Qian*, Wei Yu*, Chao Lu*, David Griffith†, and Nada Golmie†
*Towson University, USA
Emails: cqian1@students.towson.edu, {wyu,clu}@towson.edu
†National Institute of Standards and Technology (NIST), USA
Emails:{david.griffith, nada.golmie}@nist.gov

***Abstract*—Machine learning, as a viable way of conducting data analytics, has been successfully applied to a number of areas. Nonetheless, the lack of sufficient data is one critical issue for applying machine learning in Industrial Internet of Things (IIoT) systems. Insufficient data raises could negatively affect the accuracy of machine learning models. To tackle this issue, we design a framework to systematically investigate the impacts of insufficient data on model training. This framework employs the Generative Adversarial Network (GAN) and continuous learning to generate and engage new data in model training, enabling us to study the security risks of introducing new data in the model training process and develop countermeasures to mitigate these risks. To validate the efficacy of our framework, we consider a representative IIoT scenario, in which a variety of industrial components need to be recognized by Convolutional Neural Networks (CNNs), and design and implement three evaluation scenarios that are based on a real-world IIoT dataset. Our experimental results confirm that insufficient data can have a significant impact on the model accuracy, but that new data generated by GAN and continuous learning can greatly improve the model accuracy. Our experimental results also show that the data poisoning threat posed by the GAN can significantly reduce the model accuracy. However, our proposed defensive mechanism is capable of securing the model learning process. We conclude the paper by discussing some emerging issues that need to be addressed in future work.**

***Keywords*-Industrial Internet of Things, GAN, Machine Learning**

I. Introduction

The Industrial Internet of Things (IIoT), also known as Industry 4.0, has been widely considered for a variety of industry and manufacturing systems [1], [2]. In IIoT systems, massive numbers of smart sensing and actuating devices are deployed and interconnected to enable efficient and intelligent monitoring and control of systems. As smart sensing devices are deployed to monitor things/objects in the system, massive amounts of data will be collected. By analyzing collected data, the states and operational conditions of systems can be accurately estimated so that efficient monitoring and control of IIoT systems can be realized.

Machine learning is a viable technique for conducting data analytics in IIoT systems. Advancements in machine learning techniques have demonstrated great success in a variety of areas, including image recognition, object classification, natural language processing, and security [3], [4], [5]. Generally speaking, machine learning techniques can be categorized into three classes: supervised learning, unsupervised learning, and reinforcement learning.

Nonetheless, applying machine learning techniques to IIoT systems poses some challenges [1], [6]. For example, one challenge is designing machine models that can assist the co-design of larger, more complex IIoT systems. Another challenge is that decisions made by machine learning techniques need to satisfy the strict performance requirements of IIoT systems with respect to accuracy, latency, reliability, dependability, and safety, among others. Accuracy is an essential performance requirement for machine learning techniques. For example, when training a supervised machine model, a sufficient amount of data is required to avoid under-fitting the model. An under-fit model can cause classification failure, high loss, and low classification accuracy, significantly affecting the operations of IIoT systems. In an IIoT environment, the lack of training data can occur when an IIoT system is newly implemented and deployed but has not been operating for a sufficiently long time. Considering the smart manufacturing system as an example, product quality inspection using machine learning techniques requires sufficient data to ensure that the model correctly identifies components. Also, such a system could operate in a complex environment, and the operating environment may change over time. Thus, it is difficult to obtain sufficient data for model training in such a complex and dynamic environment. To address this issue, we use the GAN model to generate additional data to assist the CNN model training.

To address this issue, we design a framework to understand three fundamental problems when dealing with insufficient data in IIoT systems: (i) what is the impact of insufficient data on the performance of models? (ii) How can new data be generated from existing available data such that its inclusion in the system can improve the performance of models? (iii) What are the security risks of providing generated data to the learning process and how can the learning process and model be secured?

Using our framework, we study the effect of data on model training performance. We leverage a GAN to generate new data samples in example IIoT systems. GAN consists of two key components: generator and discriminator, which compete against one another to achieve their opposite objectives. The generator fabricates samples based on the features and statistics of original (real) samples, with the goal of deceiving

the discriminator. In opposition, the discriminator attempts to distinguish synthetic (generated) samples from original (real) samples. We systematically investigate the use of GAN for improving the performance of the model and understanding its security risks.

To summarize, our contributions in this paper are two-fold:

First, we design a framework that addresses the issue of insufficient data in IIoT systems. We use a typical CNN for industrial component recognition, which has broader applications in IIoT systems, as a case study to demonstrate the use of our framework. We define three scenarios. The first scenario considers deploying the CNN to recognize industrial components automatically, but where insufficient data may be available for model training. In the second scenario, we study how to apply the GAN to generate a sufficient volume of samples based upon real data that are available. We also leverage continuous learning to engage newly data collected from sensors, such that the model is continuously updated. In the third scenario, we investigate the security threats of using new and generated data for model training, which enables an adversary to launch data poisoning attacks by injecting adversarial examples of their own to negatively affect model accuracy. To defend against such a threat, we design a defensive mechanism to filter out adversarial examples from training data input to secure the training process of the model.

Second, we carry out extensive experiments to validate the efficacy of our framework based on a real-world IIoT dataset. Our experimental results show that our model without GAN data generation and continuous learning can suffer low accuracy when there is insufficient data available. After using the data generated by the GAN and new data collected by sensors, the accuracy of model can be significantly improved. Our experimental results also show that the data poisoning attack can significantly affect the learning accuracy, and that our designed filter-based countermeasure can effectively deal with such an attack.

The remainder of this paper is organized as follows: In Section II, we review the concept of machine learning techniques and GAN. In Section III, we propose the framework to address insufficient data in IIoT systems and describe the three scenarios, along with designed strategies. In Section IV, we present performance evaluation results. In Section V, we discuss some open issues and our future research directions. In Section VI, we review the existing research efforts that are closely relevant to our study. Finally, we conclude the paper in Section VII.

II. Preliminary

In this section, we briefly review machine learning techniques and GAN architecture.

Machine learning has shown great potential in conducting data analytics and providing intelligence and automation to a number of areas, including image and video recognition, natural language process, and others [3], [7], [8], [9], [10]. It has also been applied to a variety of IoT systems, including smart health, smart grid, smart transportation, and smart manufacturing, among others [11], [6], [12]. Generally speaking, applying machine learning has two essential steps: perception and cognition. As shown in Fig. 1, perception involves data collection while cognition involves building the model. Supervised learning usually involves classification and regression. It uses labeled training data to build a model through a training process so that the actual label of testing data can be provided subjected to a level of accuracy. This type of machine learning can be used in IIoT such as industrial component recognition systems.

GAN is a specific machine learning technique tailored for generating new data based on the existing data [13]. GAN consists of two key components: generator and discriminator. The generator learns the knowledge from the existing real data samples and generates new data samples based on the knowledge while the discriminator distinguishes the generated data samples from the real data samples. The generator updates its model based on the feedback from the discriminator and the discriminator updates its model based on the updated generated data samples and real data samples.

The architecture of GAN is shown in Fig. 2 that consists of two networks: generator G and discriminator D. Here, X is the training dataset that consists of real data examples, which is required to learn by the generator G. This dataset serves as $input(x)$ to the discriminator network D. The random noise $vector(z)$ contains random numbers that the generator uses as an entry point for synthesizing fake data samples. The generator G takes a vector of random number z as input and output fake data samples X^*. Its goal is to make fake samples indistinguishable from real samples in the training dataset. Also, $G(z)$ represents samples generated by G following the distribution of real data samples P_d. The discriminator network D takes the input from either a real data sample X from the training dataset or a fake data sample X^* produced by the generator G. In each sample, the discriminator D determines and outputs the probability of whether the data sample is real or not. If the data sample is real, the output is 1; otherwise, the output is 0.

In the iterative training process, we use the results from the generator and discriminator to iteratively tune the discriminator and generator network through backpropagation. To improve the performance of the discriminator, the weights and biases in the discriminator can be updated to maximize its classification accuracy. Meanwhile, if the performance of the generator needs to be updated, its weights and bias can be updated to maximize the probability that the discriminator misclassifies X^* as real.

Since GAN was introduced, a number of GAN models have been proposed [14], including the convolution-based GAN and condition-based GAN. In convolution-based GAN, there is one network called Deep Convolutional Generative Adversarial Network (DCGAN). Different from the original GAN, DCGAN uses the CNN model instead of Multi-Layer Perceptron (MLP) to improve the performance. Also, the condition-based GAN leverages an additional variable to control the noise vector so that the vanishing gradient issue in the training process can be overcome. Conditional Generative Adversarial Network (CGAN) [15], Information Maximizing Generative Adversarial Network (InfoGAN) [16], and Auxil-

iary Classifier Generative Adversarial Network (ACGAN) [17] are some examples of condition-based GAN. For instance, Chen [16] *et al.* proposed InfoGAN, which leverages a latent code to modify the noise (i.e., generating the same digit with different shapes). Likewise, ACGAN is an improvement to the CGAN proposed by Odena *et al.* [17]. Compared to DCGAN, ACGAN provides the capability of generating high resolution images. By introducing a class label, the generator can generate a specific class of images (e.g., flowers).

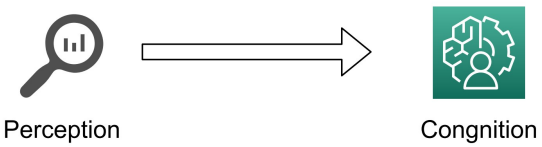

Fig. 1. Perception and Cognition Process

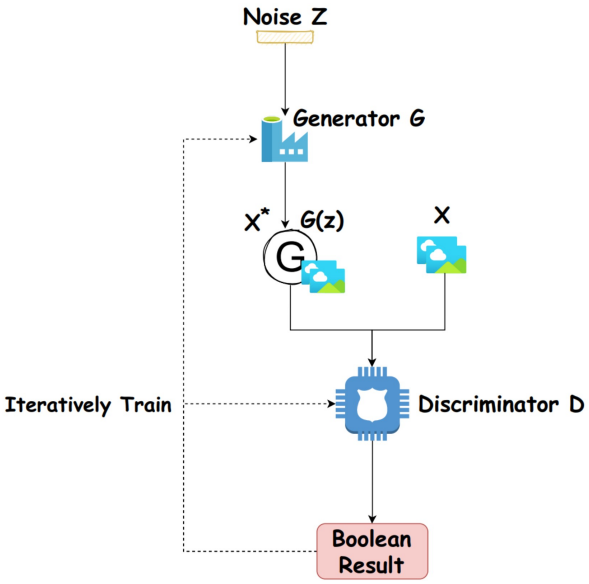

Fig. 2. GAN Architecture

III. Our Approach

In this section, we present our approach in detail.

A. Framework

In this paper, we aim to address the insufficient data issue of machine model training in IIoT systems. In Fig. 3, we design a framework that considers the following three fundamental problems: (i) *Problem A*: How can we understand the impact of insufficient data on model training? (ii) *Problem B*: How can we generate new data and/or engage new data from the system to improve the model accuracy? (iii) *Problem C*: What is the security risk of using new data in the model training and how can we deal with such a risk?

To demonstrate the use of our designed framework, we consider the IIoT scenario that applies the CNN model to carry out the recognition of images of industrial components. This scenario is generic and can be applied to numerous

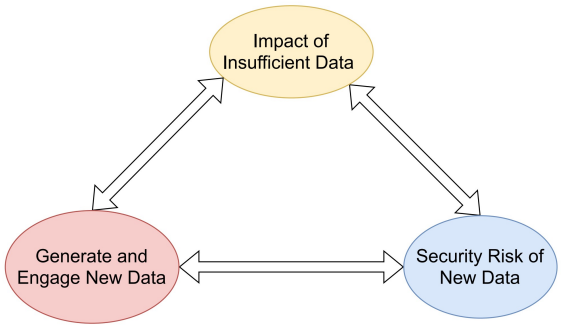

Fig. 3. Framework for addressing insufficient data in CNN training

industrial applications. Recall that, as a common problem in IIoT systems, at the beginning of training process, we may not have sufficient data (e.g., images) to train an accurate CNN model.

For Problem A, we study the accuracy of the CNN model by varying the number of data samples for model training. For Problem B, we leverage GAN to automatically generate more data samples based on the distribution of existing data samples that are available. We also engage continuous learning to incorporate newly collected data samples from sensors. Both data samples generated by GAN and/or collected by sensors can be further used as inputs to train the machine model. For Problem C, we study the security risk of new data generation, which can be used by an adversary to generate adversarial examples and launch data poisoning attacks against the training process of the model. After understanding the negative impact of adversarial examples on the accuracy of model, we design a defensive mechanism that leverages the discriminator of GAN to filter the adversarial examples. By doing this, the data poisoning threat via adversarial examples will not have much impact on the model.

DCGAN is one scheme that has been widely used in image processing [18]. Similar to the the original GAN [13], DCGAN consists of one generator and one discriminator. However, the generator of DCGAN is based on CNN, while the original GAN is based on MLP. Fig. 5 illustrates the architecture of the generator in DCGAN, in which the input is a list of random noise samples represented as a vector (say a size of 100). With the input, four deconvolution layers are employed to generate data, which is approximate to the distribution of real data samples. As the output of generator, a 64 pixel $\times$ 64 pixel three-layer Red-Green-Blue (RGB) color image is generated.

Similar to the original GAN, the generator and discriminator in GCGAN are learning networks that have their own loss functions. We denote the loss functions of the generator and discriminator as J^G and J^D, respectively. According to [13], the discriminator D is defined as a binary classifier. The loss function is represented by the cross-entropy as $J^{(D)} = -\frac{1}{2}[\mathbb{E}_{x\sim p_d}\log(D(x)) - \mathbb{E}_{z\sim Pz(z)}(\log(1-D(G(z))))]$, where x is sampled from the training data with the distribution $P_d(x)$, z is a noise vector sampled from the prior distribution $P_z(z)$ based on uniform or Gaussian distribution. Also, denote $E(\cdot)$ as the expectation operator, $G(z)$ as the data generated by the generator G, $D(x)$ as the probability that the discriminator D

discriminates x as real data, and $D(G(z))$ as the probability that the discriminator successfully determines whether the data is generated or real. The goal of the discriminator D is to distinguish the data successfully, so that the target value of $D(G(z))$ should approach 0. The goal of the generator G is the opposite; its goal is to bring $D(G(z))$ to 1. Based on this idea, there is a conflict between the generator and the discriminator.

The loss of the generator can be derived by defining it to be the additive inverse of the loss function of the discriminator: $J^G = -J^D$. Thus, the optimization problem of GAN can be transformed into the min-max game as, $\min_G \max_D \{\mathbb{E}_{x\sim p\,\mathrm{d}(x)}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]\}$. In the training process, the parameters in G and D are updated. When $D(G(z))$ is 0.5, the discriminator cannot distinguish the real and generated data. When this occurs, the model achieves the optimal solution.

Note that there are several differences between the original GAN and DCGAN. First, in DCGAN, the generator and discriminator networks do not employ all the polling layers. Second, the full connected layer is not used in DCGAN. Third, in DCGAN, ReLU activation function is used in the generator network and then the hyperbolic tangent (tanh) activation function is used in the last layer of the generator network. Lastly, Batch Normalization (BN) is used in both generator and discriminator networks in DCGAN, which can improve the performance when dealing with the initialization of the noise vector. Note that the BN layer is not recommended for the output layer of the generator network and the input layer of the discriminator network since including it could cause oscillation and model instability.

Related to aforementioned three problems, following three scenarios are designed: (i) *Traditional model training (Scenario A):* In this scenario, we build a traditional CNN model to evaluate its ability to recognize industrial components based upon the given image samples. This scenario can be used to assess the impact of sufficient data samples on model accuracy. (ii) *GAN assist model training (Scenario B):* In this scenario, we design the DCGAN-based scheme to generate new image samples based on image samples that are available. Recall that in many IIoT applications, the lack of sufficient image samples of industrial components limits the training ability of machine learning models. We employ GAN and use the generated image samples by GAN as additional training input of model in order to improve the accuracy of model. (iii) *GAN-based data poisoning attack and defense (Scenario C):* In this scenario, we investigate the GAN-based data poisoning threat, which enables an adversary to use DCGAN to generate adversarial image examples and launch the data poisoning attack against the model training. To defend against such an attack, we leverage the discriminator of DCGAN to generate adversarial image examples and record them. Thus, we can remove any adversarial image examples from the input of model training.

Our designed framework is generic and can be used as a foundation to overcome the insufficient data issue in a variety of IIoT systems. In the following, to demonstrate the use of our framework, we consider an IIoT system that leverages CNN to automatically recognize the image of critical industrial components, leading to the automation of IIoT systems. The system architecture is shown in Fig. 4. As shown in the figure, IoT sensors deployed in industrial systems have the ability of capturing the images of objects (e.g., manufacture components in an assembly line). The CNN model is used to automatically recognize different types of IIoT components. When the IIoT system starts to operate, we assume there are limited image samples of IIoT components to use. To overcome the issue of lacking sufficient image samples for accurate model, we leverage DCGAN, which uses the existing image samples to generate synthetic image samples based on the distribution of original images samples. The newly generated image samples DCGAN produces will be fed into the CNN model to perform the training process. In addition, new image samples collected by IoT sensors can be fed into the CNN model so that the model can be further updated.

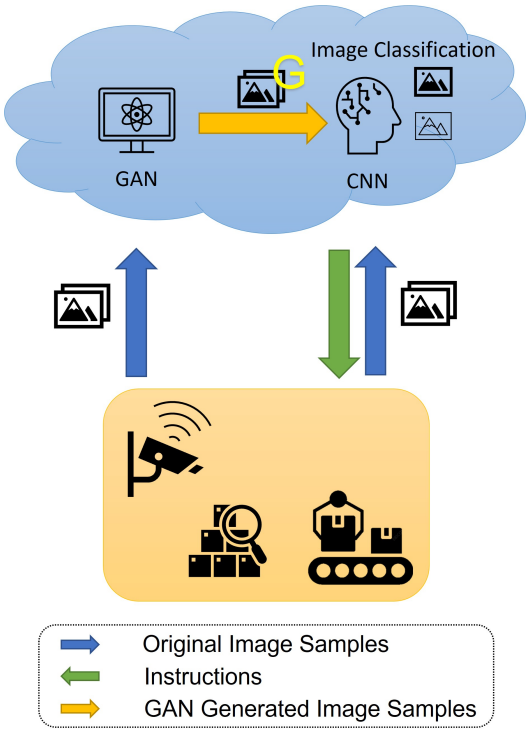

Fig. 4. Component Recognition of IIoT Systems

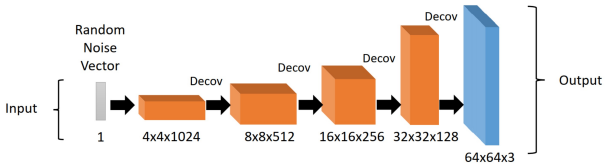

Fig. 5. DCGAN Generator Architecture

B. Scenario A: Traditional Model Training

In this scenario, we use the traditional CNN model to perform image classification so that the IIoT system can recognize industrial components to automate the manufacturing process. The CNN model consists of two convolutional

layers, two max-pooling layers, one flatten layer, two fully connected layers, and one dropout layer. The structure of the CNN model is shown in Fig. 6. The convolutional layer is used to extract features in the examined images. The max-pooling layer is used to compress the information of images extracted by the convolutional layer so that the size of image and computational complexity can be reduced. Also, max-pooling layer with a flatten layer is used to convert the image feature information to a 1-dimensional array and provides the classification information. FC (fully-connected) layer is used to classify the input data into various classes. The dropout layer is used to prevent a model from over-fitting.

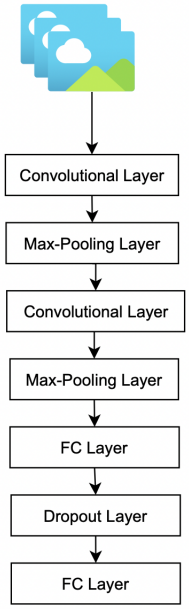

Fig. 6. CNN Architecture

C. Scenario B: GAN Assist Model Training

In this scenario, we propose the DCGAN-based scheme to automatically generate image samples so that the accuracy of CNN model can be improved. Again, in many IIoT applications, there are not enough data samples to build accurate models. To address this issue, we leverage DCGAN to generate additional image samples based on available image samples.

Our scheme in this scenario is shown in Algorithm 1. We first initialize the epochs of the whole training process that represents the training time of DCGAN model. Also, denote m as the number of the image samples for training both the generator and discriminator, n as the number of the generated data, and *iteration* as training stage (e.g., epoch), respectively. Further, *balance* refers to the relationship between the learning rate of generator and discriminator, which is set to 0.5 to make the generator and discriminator have the same learning rate. Also, $g\_threshold$ is configured to determine the time when the generated images are ready to use. Denote z as the noise vector of the generator and $G(z)$ as the generator output, $D(t)$ as the discriminator output for a given input t, which is the real data sample from the dataset or the data sample from the generator $G(z)$. Also, denote $P_z(z)$ as the distribution of the generated samples from the generator.

Algorithm 1: Algorithm for Scenario B

```
Initialization: epochs, m, n iteration, balance=0.5, g_threshold z;
  while iteration < epochs do
      iteration++
      Extract n samples from G(z) with the distribution P_z(z)
      Extract m samples from real dataset with the distribution P_d(x)
      Samples from both G(z) and real image are sent to D(t)
      D(t) sends the results (fake/real)(d_loss) to G(z)
      G(z) adjusts the distribution P_z(z) based on the result of D(t)
        and output g_loss
      if g_loss < g_threshold then
          Save images G(z)
          Send images G(z) to the training set of the CNN model
      end
end
```

During each step of training, we first extract the same number of image samples from both the generator and the real dataset whose distribution is $P_d(z)$. Then, all the image samples from both the generator and the dataset are sent to the discriminator. The discriminator evaluates the generated and real image samples based on the different distribution of the image samples and outputs the binary result (0 means generated image and 1 means real image). Based on the results from the discriminator, the generator adjusts the distribution of generated image samples $P_z(z)$ and outputs the loss of the generator g_loss. If the loss of the generator reaches a threshold $g\_threshold$, the DCGAN training is marked as completion. In this case, we store the image samples, and use them later for training the CNN model. The complexity of this algorithm is $O(n)$, where n represents the number of epochs of the training process.

The new image samples that are collected by IIoT sensors over time can also be used to update the CNN model. To this end, to improve the learning accuracy, we leverage the concept of continuous learning and design Algorithm 2, which incorporates new data samples collected by sensors to retrain the model. The parameters for Algorithm 2 are listed in Table I. In our algorithm, several parameters need to be initialized: epochs as the training time of CNN model with generated data, m as the number of generated data samples, $real\_num$ as the number of images collected from sensors, $real\_threshold$ as the minimum quantity of the image samples collected from sensors that will trigger a retraining of the CNN model, and $epochsr$ as the number of epochs to train the CNN model with the samples collected from sensors. After the initialization, we first train the CNN model with the generated images by DCGAN and save the trained model. After the training process, we record the updated CNN model for future use. Then, we enter the continuous learning phase, where the system collects sensor images. Once the number of new images exceeds the threshold, the CNN model retraining triggers, after which the updated CNN model is saved and the algorithm returns to collecting new sensor images. The complexity of this algorithm is $O(n)$, where n represents the number of epochs of the training process.

TABLE I
LIST OF KEY NOTATIONS FOR SCENARIO B

m	Amount of generated data
$real\_num$	Number of images collected from sensors m
$real\_threshold$	Minimal image samples to trigger the model retraining
$epochsr$	Number of epochs to train the CNN model

Algorithm 2: CNN model update using continuous learning

```
Initialization: epochs, m, real_num, iteration, real_threshold, epochsr
while iteration < epochs do
    iteration++
    Train CNN model with m generated images from Algorithm 1
end
Save the trained model
while true do
    Get new sensor images
    Increase real_num by new image count
    if real_num >= real_threshold then
        load real images and train the CNN model
        while iteration < epochsr do
            iteration++
            Update the CNN model using real images
        end
        Save updated model
        real_num = 0
    end
end
```

D. Scenario C: GAN based Data Poisoning Attack and Defense

In this scenario, we focus on the security issue of GAN-based IoT component recognition from both attack and defense perspectives. First, the GAN can be used by the adversary to generate adversarial image examples based on available image samples. Such adversarial image examples can be used to launch the data poisoning attack to affect the training process, which can further disrupt the operation of IIoT systems. Second, to defend against the data poisoning attack, we design the data filtering mechanism to remove the adversarial examples from the model input during training. We use the discriminator of DCGAN to collect adversarial image examples. Based on the collected adversarial examples, we can filter any adversarial image examples in the training input so that the training process is secured.

For generating adversarial image examples, Algorithm 3 shows the detailed procedure. Similar to other algorithms, we initialize a number of parameters: the *epochs* as the total training time of the model, m as the number of data samples for the discriminator, n as the number of data samples generated by the generator, and the iteration as a counter for training steps. Again, *balance* refers to the learning rate of the generator. The learning rate of the discriminator is set to $1 - balance$. Note that the learning rate represents the update frequency of the discriminator and generator of the GAN model. The learning rate ranges in [0, 1]. When the learning rate of the generator is higher than that of the discriminator, the GAN-generated images can then be used as adversarial examples for the CNN model and affect the classification accuracy. At the same time, if the learning rate of the discriminator is higher than the learning rate of the generator of the GAN model, the discriminator can then distinguish adversarial samples from real samples. Only when the learning rates of the generator and the discriminator are the same, the samples from the generator can more closely follow the distribution of real data samples. By using GAN generated samples, we can improve the classification accuracy of CNN model when the dataset is insufficient. Also, $g\_threshold$ is used to determine whether the adversarial samples are ready to use and z refers to the noise vector for the generator.

To generate adversarial image examples, we first reduce the balance. In this case, the generator has the smaller learning rate than the discriminator, meaning that the discriminator has higher updating rate than the generator. In such a case, the generated image samples will not be likely to follow the distribution of the original images. Thus, when the adversary examples are used in the training process, the accuracy of the model will be reduced. When the adversarial examples are generated, the adversary can launch the data poisoning attack by injecting the generated adversarial image examples as inputs to model training. In real-world practice, the data poisoning attacks based on the generated adversarial examples are feasible when the adversary finds ways to compromise either the image sensors of the IIoT system that generates training samples, network components, or computing servers that collect and process image samples [19], [20]. When the sensors, networking components, or computing servers are compromised, the adversary can launch data poison attacks or false data injection attacks to inject adversarial samples into training samples or even online recognition samples. By doing this, the component recognition accuracy of IIoT systems will be affected. In either case, the adversary can inject the generated adversarial examples into the model training process. On one hand, the adversary can directly launch the attack against the DCGAN model (e.g., modifying the learning rate to enable the generator generates adversarial samples). On the other hand, we leverage continuous learning discussed in Scenario B and use the newly collected data samples from sensors to retrain the CNN model. Thus, the adversary can find ways of compromising sensors, networking components, or computing servers, and launching a data poisoning attack to inject the adversarial data examples into the retraining process of model. The complexity of Algorithm 3 is $O(n)$, where n represents the epochs of the training process.

Algorithm 3: Algorithm for Scenario C (Attack)

```
Initialization: epochs, m, n, iteration, balance=0.1, g_threshold z;
while iteration < epochs do
    iteration++
    Extract n samples from G(z) with the distribution P_z(z)
    Extract m samples from real dataset with the distribution
      P_data(x)
    Samples from both G(z) and real image are sent to D(z)
    D(z) sends the result (fake/real)(d_loss) to G(z)
    G(z) adjusts the distribution P_z(z) based on the result of
      D(z) and output g_loss
    if g_loss < g_threshold then
        Save images G(z)
    end
end
```

To defend against the data poisoning attack imposed by DCGAN, we now describe our countermeasure as shown in Algorithm 4. Here, denote A as the set that consists of the collected adversarial image examples and denote I as a set that consists of images as training input, which include image samples generated by GAN and image samples collected by sensors. We assume some adversarial image examples may be injected into training input I by the data poisoning attack that we discussed before. To defend against the data poisoning attack, we consider the filter mechanism to remove the adversarial image examples before entering into the model training process. When adversarial samples enter the training process, the accuracy of the model may be affected. To deal with such an issue, we then design Algorithm 4 to filter adversarial examples to mitigate the adversarial impact on the performance of the CNN model. The iteration in this algorithm is used to examine all images in I and remove adversarial image samples in I that match the adversarial image samples in A. The computational complexity is determined by the size of I. Thus, the computational complexity of Algorithm 4 is $O(r)$ where r is the size of set I. To generate the adversarial samples, we adjust the balance to 0.1, which makes the discriminator have a higher learning rate than the generator. In this case, the generator is less likely to follow the distribution of the original images so that the adversarial image example can be generated. To deal with the adversarial samples, the defender can utilize the GAN as a defensive tool and increase the learning rate of the discriminator, enabling it to identify adversarial images from real images. In this way, the adversarial impact on the classification accuracy of CNN model can be mitigated. If any adversarial image example is identified, it will be removed from the training input. Note that in Algorithm 4, we leverage the discriminator to distinguish the adversarial images. The matches in Algorithm 4 represent that the adversarial images are identified by the discriminator.

Algorithm 4: Algorithm for Scenario C (Defense)

```
Initialization: epochs, m, iteration, balance=0.1, g_threshold, z, A, I
Call Algorithm 3 to generate adversarial examples and store them in A
while p in I do
    if p matches any image in A then
        Remove p from I
    end
end
Train CNN model using I
```

IV. Performance Evaluation

In this section, we show the efficacy of our approach. In the following, we first introduce the methodology and then discuss the evaluation results.

A. Methodology

In our experiment, we use the T-LESS dataset [21] as the IIoT component images. Recall that automatically recognizing images in T-LESS dataset is important in the automation of IIoT systems. The T-LESS dataset has 1296 images for each category (30 categories in total). For insufficient data, we leverage 20 %, 40 % as the training dataset and 30 % of the original set as testing dataset. Since the goal of our study is to investigate the impact of insufficient data on model accuracy, we use the relatively small amount of data, say 20 %, 40 % as training data to emulate such insufficient data situations. Meanwhile, we determine whether the data size is sufficient based on the accuracy of the learning model. For Scenarios B and C, we use 50 % images for each category to train the DCGAN model to generate image samples for model improvement and defend against the data poisoning attack. Note that Scenario A is for assessing the performance of model when different levels of data are available. We expect the learning accurate will increase after DCGAN is involved in Scenario B. Furthermore, in Scenario C, we expect that the adversarial examples generated by the adversary's DCGAN can reduce the model accuracy and the issue can be resolved after the defender uses their own DCGAN to filter out the adversarial image examples.

To measure the accuracy of the model, we define two metrics. One is the classification accuracy in range of $[0, 1]$, which is the ratio of the number of image samples that are successfully categorized to the total number of image samples. A greater value of the ratio indicates better performance of the model. The other metric is the loss of learning objective function, which is defined by the cross entropy $\text{CrossEntropy} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$, where M represents the number of categories (i.e., the T-LESS dataset has 30 categories), y is the binary indicator (0 represents that the classification is not correct while 1 represents that the classification is correct), and p represents the probability of the observation o of class c.

The software that we used in our experiment is MATLAB r2021a, Pytorch 1.8, and Tensorflow 2.4 [1]. The software were run on a PC with Windows 10 version 2004. The hardware in the PC is an Intel i7 8700 CPU and 16 G DDR4 RAM. We evaluate the results for 8 times and show the results with 95 % confidence intervals in all figures.

B. Results

In the following, we show the evaluation results for the three scenarios defined in our framework. All colored regions in all figures represent the designated confidence intervals and the results in the figures represent the validation results. Note that if there exists overfitting, we expect the accuracy will have poor performance during the validation when experiencing unseen data.

1) Scenario A: Traditional Model Training: In Scenario A, the CNN model for IIoT component classification is used to classify different IIoT components from various image samples. As the output image size of DCGAN in Scenarios B and C is 64x64, we resize the input image of CNN model

[1]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

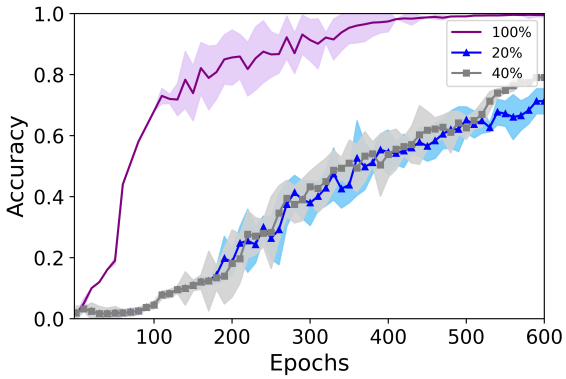

Fig. 7. Accuracy of Scenario A using 20 %, 40 %, and 100 % of the data set for training

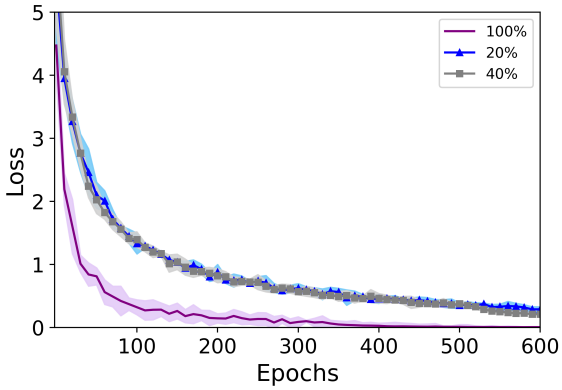

Fig. 8. Loss of of Scenario A using 20 %, 40 %, and 100 % of the data set for training

into 64x64 in Scenario A. As our focus is to recognize the shape of industrial components, we map the input image from RGB to gray-scale. In our CNN model, we leverage 32 convolution kernels (kernel size of 5x5), the stride of 1, and use ReLU activation function in the first convolution layer to perform feature extraction. Then, we implement a max-pooling layer with stride of 2 and padding of 2 to remove the redundant information from the original image. For the second convolutional layer, we leverage 64 convolution kernels with 3x3 kernel size, stride of 1, and ReLU activation function to further extract the information. Also, we leverage the second max-pooling layer with the same setting to remove the redundant information. After that, we leverage fully connected layers with 1024 output space feature maps to classify the features extracted by the convolutional layer. Furthermore, we use a dropout layer with a dropout rate of 0.3 to prevent over-fitting. Finally, a fully connected layer is implemented to classify the features with the output space set of 30.

We set the total training epochs to 600 and conduct the training and validation process. The results are shown in Fig. 7 and Fig. 8. From both figures, we can observe that when the training process is complete, the accuracy ratio approaches 0.998 and the loss value drops to around 0.002. From the results observed, we can see that the CNN model can successfully classify different IIoT components if sufficient amount of data are available.

To evaluate the impact of insufficient data on the accuracy of CNN model, we examine two cases, where we use 20 % and 40 % of original training set to perform training. The results of the CNN training with 20 % of the original dataset are in Fig. 7 and Fig. 8. From the figures, we can see that the accuracy ratio after 600 epochs reaches 0.713 and the loss value drops to around 0.287. The result of the CNN training with 40 % of the original training set is also in Fig. 7 and Fig. 8. The accuracy ratio reaches 0.791 and the loss value drops to around 0.209 when 40 % of original training set is used. Our experimental results confirm that the accuracy of CNN model can be significantly reduced with insufficient dataset and that the marginal improvement associated with doubling the training data fraction from 20 % to 40 % is small, which indicates that we need large amounts of data to properly train the model. We also tested 60 % and 80 % of the original dataset to train the CNN model, the accuracy is in range of 80 % to 91 %. This justifies the reason why we need to automatically generate data samples to address the insufficient data issue and improve the accuracy of CNN model.

2) Scenario B: GAN Assist Model Training: In this scenario, we first leverage the DCGAN to generate IIoT component images based on 50 % of the available images for training an accurate machine model. Then, we train the CNN model with the generated image samples to evaluate the accuracy of our model. In addition, we leverage the continuous model given in Algorithm 2 in Section III to simulate the real-world practice of the IIoT system, in which new images are collected over time and such images should be employed to update the model.

When we implement Algorithm 1, we set the number of training epochs to 60. The image sample m is set to 648, which is 50 % of the total images for each category. The balance is set to 0.5 to balance the learning rate on the generator and discriminator. Also, $g\_threshold$ is set to 3 to control when DCGAN is completed. The length of the noise vector z is set to 100. After the initialization, we load the T-LESS dataset into the model. The generator generates the images based on the distribution of the real image, while the discriminator distinguishes the real/generated data samples and send its feedback to the generator. Based on the feedback, the generator adjusts the image distribution. When the loss of the generator is less than 3, the generated images are ready to use. As an example, we show the initial output of the generator in Fig. 9. From the figure, we can see that there is only noise in the generated images in the initial stage. After 60 epochs of training process, the outputted image is shown in Fig. 10, which is much similar to the original image shown in Fig. 11.

The accuracy of DCGAN is shown in Fig. 12. From the figure, we observe that the accuracy of the discriminator remains the range of between 0.730 and 1 from epochs 0 to 30. This is because the generator is still updating its parameters based on the output of the discriminator and the original images. After epoch 30, the accuracy of both the generator and discriminator are changing more rapidly than previous epochs. Nonetheless, the accuracy of the generator increases while the

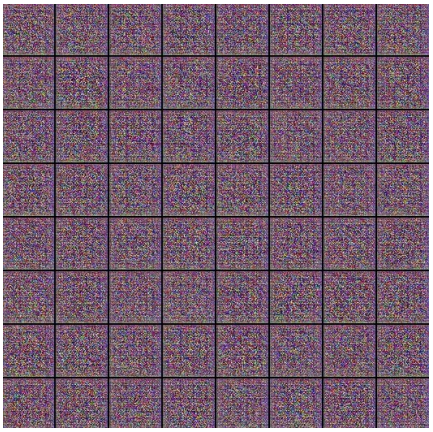

Fig. 9. Initial output of DCGAN for Scenario B

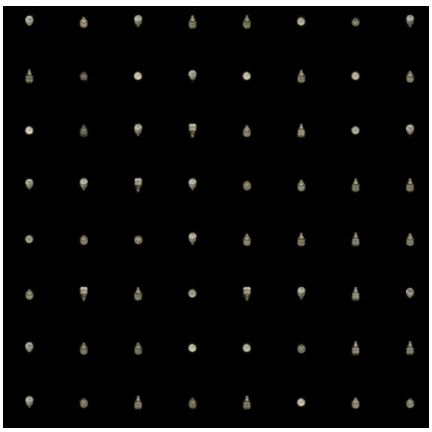

Fig. 10. Example of Generated Images by DCGAN based on T-LESS for Scenario B

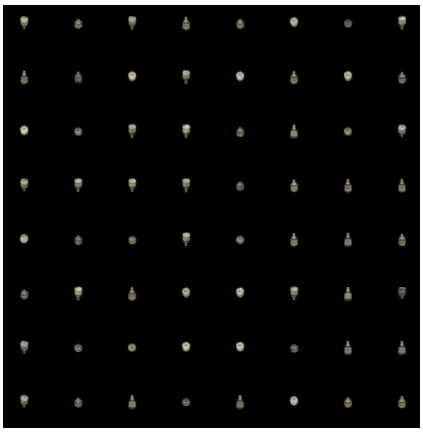

Fig. 11. Example of Original T-LESS Images for Scenario B

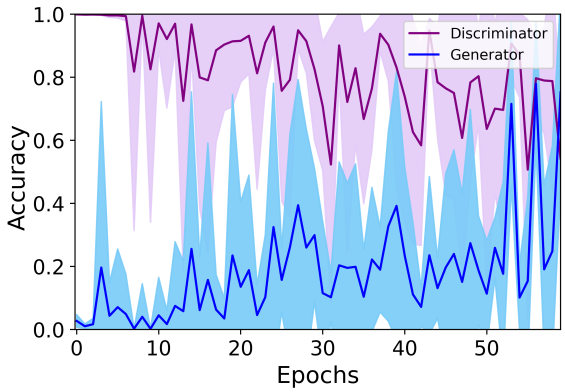

Fig. 12. Accuracy of Scenario B with DCGAN

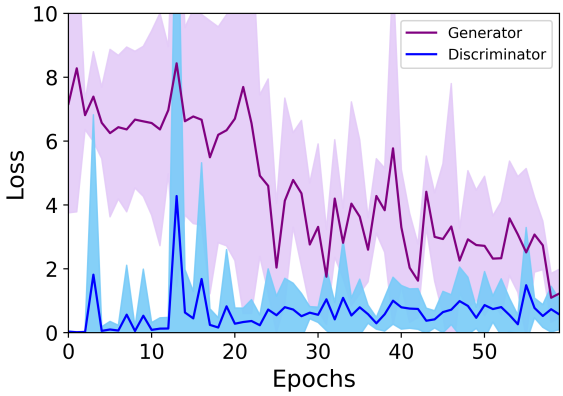

Fig. 13. Loss of Scenario B with DCGAN

accuracy of the generator declines. At the end of the training process, the generator achieves an accuracy score of 0.814, while the discriminator achieves a score of 0.781 in average.

The loss of the DCGAN determined by cross-entropy is shown in Fig. 13. From the figure, we observe that generator incurs a high loss during the training epochs from 0 to 26. Starting from epoch 26, we can see highly variation on both generator loss and discriminator loss. This is because both the generator and discriminator are actively updating their models based on the feedback (e.g., a binary decision of either real or fake image from the discriminator) and the actions for generating the images from the generator.

After we collect the images from the DCGAN model, we implement Algorithm 2 to demonstrate how the generated image samples and continuous learning can jointly improve the learning accuracy of the CNN model. In the algorithm, we set the epochs for CNN training as 600, the quantity of generated images m as 1296. Also, $real\_num$ and iteration are initialized to 0 and $real\_threshold$ is set to 1296 which means the IIoT sensor will send the collected images to the CNN model after collecting 1296 images. Then, the CNN model will be updated based on newly collected image samples from IIoT sensors. The $epochs$ is set to 350, which refers to the training time of the continuous learning process. In the continuous learning process, we leverage the IIoT sensor collected images (over or equal to 1296) to update the CNN model.

We plot the evolution of the accuracy and the loss versus the epoch number of the CNN model in Figs. 14 and 15, respectively. During the initial training period of 600 epochs, we train the CNN model based on the DCGAN generated samples. We observe that the accuracy ratio is 0.889 at epoch 600. From epoch 600 to 950, we add the generated image data samples collected by sensors, which enables the CNN model to update itself via continuous learning process. After updating the CNN model, the accuracy ratio approaches 0.998. Based on our experimental results, we confirm that the DCGAN along with continuous learning is capable of improving the learning accuracy of CNN model when the initial data samples is not sufficient, which is a common problem in numerous IIoT systems. As seen in Fig. 14, the validation accuracy increases with the growth of epochs so that the overfitting problem is not a concern in our proposed model.

3) Scenario C: GAN based Data Poisoning Attack and Defense: We now investigate the security impact of DCGAN on model training in the investigated IIoT system. The adversary can use DCGAN to generate adversarial image examples to bypass the recognition of CNN model, which can significantly affect the component recognition process in the investigated IIoT system. To address this issue, We implemented Algo-

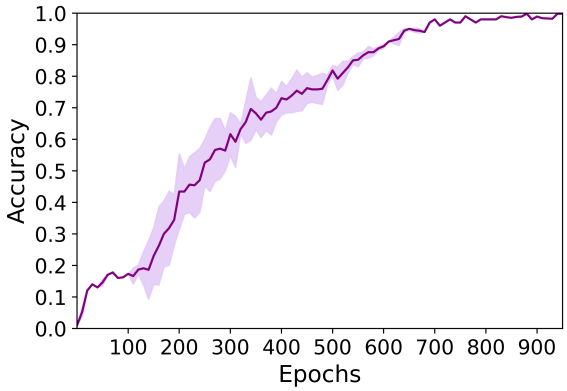

Fig. 14. Accuracy of Scenario B with Continuous Learning

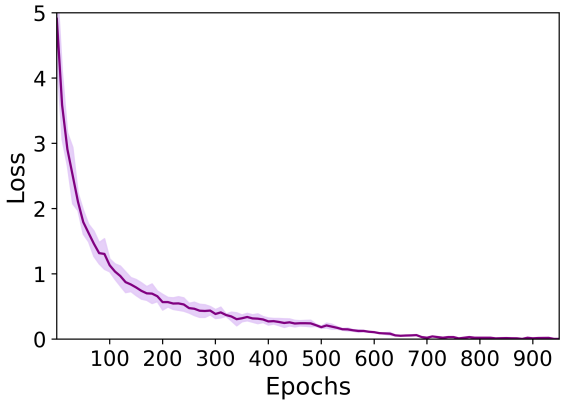

Fig. 15. Loss of Scenario B with Continuous Learning

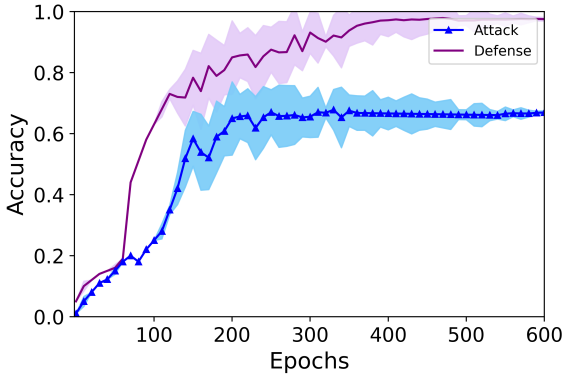

Fig. 16. Accuracy of Scenario C (Attack and Defense)

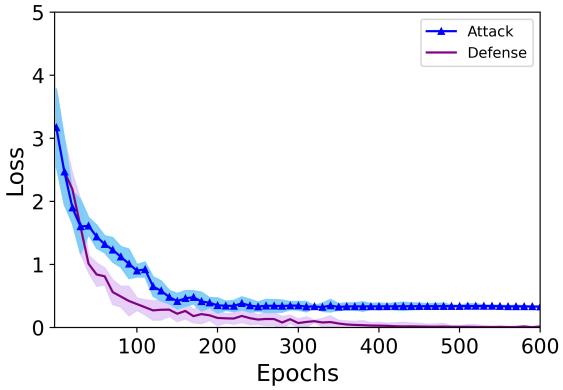

Fig. 17. Loss of Scenario C (Attack and Defense)

rithm 3. In our experiment, we set the number of training epochs to 600, image quantity m to 1296, $iteration$ to 0, and $g\_threshold$ to 4, and z to 100. At this stage, the generated sample is not likely to follow the distribution of the original images that can be regarded as the adversarial examples so that the accuracy of the model will be affected. The balance is set to 0.1, which ensures the generator does not get updated as frequently as the discriminator. After training, the newly generated image samples will be used as inputs to confirm the performance impact of adversarial image examples generated by DCGAN on the trained CNN model.

To demonstrate the attack impact of DCGAN as an attacking strategy, Fig. 16 shows the accuracy of model and Fig. 17 shows the loss of model. From both figures, we can observe that, after the attack is launched, the accuracy ratio is plateaus at 0.635 and the loss can reach to 0.33. Note that the accuracy is significantly lower than that in both Scenarios A and B and the loss is much higher than that Scenarios A and B, respectively. Our experimental results confirm that the adversarial image examples by DCGAN can significantly affect the performance of our model.

The discriminator of DCGAN can be used as a defensive strategy to classify and filter the adversarial image examples so that the security resilience of CNN model can be achieved. Similar to the attack investigated in this scenario, we implement Algorithm 4. We set the number of training epochs to 600, image quantity m to 1296, iteration to 0, $d\_threshold$ to 1, I to 1296, A to 1616, and z to 100. During the training process, since the discriminator has higher learning rate, it has better performance than the generator. In this paper, we leverage the discriminator of the DCGAN to classify the adversarial image examples from the real examples. To simulate the knowledge that the defender knows the adversarial examples, we use the filtered image samples to train the CNN model in Scenario A. The experimental results are shown in Fig. 16 and Fig. 17. As seen in both figures, with the defensive strategy, the accuracy ratio can approach 0.967 while the loss approaches 0.150, showing that our defensive strategy is capable of dealing with data poisoning attacks against the model training process. To deal with the scenario where the adversary uses a different balance value to generate adversarial samples, the defender can use different balance values to generate adversarial examples and train the discriminator with those samples. By doing this, the discriminator will have the ability to filtering the adversarial samples under other balance values.

V. Discussion

There are still several remaining issues that need to be addressed in our future study.

Improving GAN on IIoT Component Recognition: In our implementation, the generator of DCGAN may encounter

mode collapse when dealing with complex data samples. Thus, one issue is how to select different cost functions for further performance gain. Also, additional tuning can be conducted for DCGAN model so that the generator can generate more realistic image samples for the model. With respect to feature extraction, it will be interesting to explore various techniques so that the discriminator can easily recognize adversarial image examples. Another issue is how to select and prioritize the features of images so that we can not only reduce training data size, but also reduce computational overhead. As IIoT systems could collect high resolution images, how to make the discriminator and generator networks in DCGAN adapt images with different resolutions is another issue to study in our future work.

Extending our framework to other IIoT Systems: Recall that in this study, we have proposed the framework to establish the foundations of addressing the problem of insufficient data in IIoT systems, including the study of the impact of insufficient data on the performance of models, the techniques that enable new data generation, collection, and use in model training, as well as the security risk of using new data in model training process. Our framework can be extended to study other GAN models to provide new insights that support the performance improvement of models. Our framework can also be extended to other IIoT applications such as smart cities, smart healthcare, smart transportation, and others. For example, GAN can be used to synthesize new data to build robust models to characterize communication channels and networking environments in complex IIoT systems, in which only limited measurement data could be available. Furthermore, other data security and privacy issues posed by machine learning use in IIoT systems are interesting problems to explore in our future work [22], [20].

VI. Related Work

In this section, we review the existing efforts on applying GAN in different problem domains. One important application of GAN is computer vision. GANs are used for image super-resolution, image translation, image texture synthesis, face synthesis, video generation, and text-to-image translation, among others. For instance, Ledgi *et al.* [23] proposed a Super-resolution GAN (SRGAN) to improve the resolution of images. The designed model can take a low-resolution image as input and generates high-resolution image of four times up-scale. Also, You *et al.* [24] leveraged the GAN-Cycle and Super-Resolution Generative Adversarial Network (SRGAN) to improve the quality of Computed Tomography (CT) images. To improve the texture generated by SRGAN, Wang *et al.* [25] proposed an Enhanced SRGAN (ESRGAN) to improve the adversarial loss and perceptual loss. Likewise, Xiong *et al.* [26] improved the loss function of the SRGAN to assist image sensing. Also, Zeng *et al.* [27] integrated Super Resolution with Invert Tone Mapping (SR-ITM) with SRGAN named SR-ITM-GAN to generate 4K images.

Image translation is a way of converting the image content from one domain to another. There are a number of efforts in this aspect. For example, Isola *et al.* [28] designed a Conditional GAN (CGAN), namely as pix2pix (image-to-image translation), to show its effectiveness on both graphics and vision tasks. Wang *et al.* [29] proposed an enhanced strategy called pix2pixHD, which uses a global generator and a local generator to generate images with a specific resolution. Also, Annop *et al.* [30] proposed Sem-GAN to generate higher quality images than other GAN networks. Furthermore, to improve the scalability of the output image, Cycle GAN [31], Disco GAN [32] and Dual GAN [33] were designed to leverage the encoder-decoder framework and use the principle of cyclic consistency to enable the arbitrary size of the input image.

Image texture synthesis can construct large image based on a small digital sample from its structure of the digital sample. For example, Li *et al.* [34] proposed a texture synthesis scheme based on GANs, namely Markovian GANs (MGANs). Their proposed scheme can capture the texture data of Markovian patches and generate the stylized images and videos. Jetchev *et al.* [35] proposed a SpatialGAN (SGAN) that applies full unsupervised learning to texture synthesis. Likewise, Bergmann *et al.* [36] proposed an enhanced SGAN called PeriodicSGAN (PSGAN), which can handle texture information in noise space, synthesis high-resolution textures, and learn periodic textures from a single image or complex dataset. Face synthesis is a method used to generate realistic faces. Related to this aspect, Huang *et al.* [37] proposed a Two-Pathway GAN (TP-GAN) to synthesize high-resolution frontal face images with different directions from a single side photo. Zhang *et al.* [38] designed the self-attention block with GAN (SAGAN) to improve the quality of image synthesis. Also based on SAGAN, Brock *et al.* [39] proposed BigGAN to increase the diversity and fidelity of generated samples. Furthermore, for video generation, Tulyakvo *et al.* [40] proposed unsupervised learning based MoCoGAN to carry out the video generation. For text-to-image translation, Reed *et al.* [41] and Zhang *et al.* [42] designed schemes to generate images based on texture description.

Natural language processing (NLP) is another application that GANs have been applied to. For example, Yu *et al.* [43] proposed SeqGAN using the policy gradient to train the generator, which achieves desirable performance in speech, poetry, and music generation. Lin *et al.* [44] leveraged a margin-based loss function to the discriminator of GAN, named RankGAN, to improve the performance. Likewise, Li *et al.* [45] designed a scheme to generate an open-domain dialogue through adversarial training. Furthermore, GANs can be integrated with reinforcement learning to carry out NLP. For example, Mnih *et al.* [46] leveraged the results of the discriminator as a reward to the generator so that human-like conversations can be generated automatically.

In addition to image processing and NLP, GANs have been applied to other problem domains. For example, anomaly detection (i.e., outlier analysis) tends to recognize data points, events, and/or observations that deviate from normal behaviors, which has broader applications. Related to such areas, Schlegl *et al.* [47] leveraged GAN to carry out the outlier detection in the healthcare system. Liu *et al.* [48] used GAN to generate malware on Android devices. Cai *et al.* [49]

systematically reviewed the privacy and security of GAN. Likewise, Jordon *et al.* [50] leveraged GAN to generate data that complies to defined privacy policies.

VII. Final Remarks

In this paper, we proposed a framework to address the issue of insufficient data for training accurate machine learning models in IIoT systems. In our framework, we designed three scenarios to investigate the impacts of data volume on model accuracy, apply DCGAN to generate additional data samples and leverage continuous learning to improve model accuracy, study the data poisoning attack raised by DCGAN for data generation, as well as design a countermeasure for deal with this attack. To demonstrate the efficacy of our proposed framework, based on a real-world IIoT dataset, we implemented the three scenarios. Our experimental results show that insufficient data can significantly affect model accuracy, and that generated data samples from DCGAN can greatly improve the model accuracy. Our experimental results also show that adversarial examples generated by DCGAN can significantly reduce model accuracy in an attack scenario, and our proposed countermeasure is effective in dealing with such an attack. Finally, we discussed some remaining issues that we plan to address in our future work.

References

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial Internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[2] X. Liu, W. Yu, F. Liang, D. Griffith, and N. Golmie, "Toward deep transfer learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 163–12 175, 2021.

[3] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[4] H. Guo, N. Zhang, S. Wu, and Q. Yang, "Deep learning driven wireless real-time human activity recognition," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[5] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.

[6] H. Xu, X. Liu, W. Yu, D. Griffith, and N. Golmie, "Reinforcement learning-based control and networking co-design for industrial internet of things," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 885–898, 2020.

[7] Y. Sun, Z. Huang, H. Zhang, Z. Cao, and D. Xu, "3drimr: 3d reconstruction and imaging via mmwave radar based on deep learning," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2021, pp. 1–8.

[8] J. Xu, S. Zhu, H. Guo, and S. Wu, "Automated labeling for robotic autonomous navigation through multi-sensory semi-supervised learning on big data," *IEEE Transactions on Big Data*, vol. 7, no. 1, pp. 93–101, 2021.

[9] D. Wu, H. Shi, H. Wang, R. Wang, and H. Fang, "A feature-based learning system for internet of things applications," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1928–1937, 2019.

[10] C. Qian, X. Liu, C. Ripley, M. Qian, F. Liang, and W. Yu, "Digital twinmdash;cyber replica of physical things: Architecture, applications and future research directions," *Future Internet*, vol. 14, no. 2, 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/2/64

[11] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, feb 2022. [Online]. Available: https://doi.org/10.1145/3501296

[12] R. A. Khalil, N. Saeed, M. Masood, Y. M. Fard, M.-S. Alouini, and T. Y. Al-Naffouri, "Deep learning in the industrial internet of things: Potentials, challenges, and emerging applications," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 016–11 040, 2021.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[14] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–43, 2019.

[15] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *arXiv preprint arXiv:1606.03657*, 2016.

[17] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *International conference on machine learning*. PMLR, 2017, pp. 2642–2651.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[19] W. Yu, D. Griffith, L. Ge, S. Bhattarai, and N. Golmie, "An integrated detection system against false data injection attacks in the smart grid," *Sec. and Commun. Netw.*, vol. 8, no. 2, p. 91–109, Jan. 2015. [Online]. Available: https://doi.org/10.1002/sec.957

[20] F. Liang, W. G. Hatcher, W. Liao, W. Gao, and W. Yu, "Machine learning for security and the Internet of Things: The good, the bad, and the ugly," *IEEE Access*, vol. 7, pp. 158 126–158 147, 2019.

[21] "T-less dataset," http://cmp.felk.cvut.cz/t-less/.

[22] X. Liu, W. Yu, F. Liang, D. Griffith, and N. Golmie, "On deep reinforcement learning security for industrial internet of things," *Computer Communications*, vol. 168, pp. 20–32, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420320193

[23] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2107–2116.

[24] C. You, G. Li, Y. Zhang, X. Zhang, H. Shan, M. Li, S. Ju, Z. Zhao, Z. Zhang, W. Cong, M. W. Vannier, P. K. Saha, E. A. Hoffman, and G. Wang, "CT super-resolution GAN constrained by the identical, residual, and cycle learning ensemble (GAN-CIRCLE)," *IEEE Transactions on Medical Imaging*, vol. 39, no. 1, pp. 188–203, 2020.

[25] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[26] Y. Xiong, S. Guo, J. Chen, X. Deng, L. Sun, X. Zheng, and W. Xu, "Improved SRGAN for remote sensing image super-resolution across locations and sensors," *Remote Sensing*, vol. 12, no. 8, p. 1263, 2020.

[27] H. Zeng, X. Zhang, Z. Yu, and Y. Wang, "SR-ITM-GAN: Learning 4k UHD HDR with a generative adversarial network," *IEEE Access*, vol. 8, pp. 182 815–182 827, 2020.

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[29] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.

[30] A. Cherian and A. Sullivan, "Sem-GAN: semantically-consistent image-to-image translation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1797–1806.

[31] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (GANs): A survey," *IEEE Access*, vol. 7, pp. 36 322–36 333, 2019.

[32] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1857–1865.

[33] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2849–2857.

[34] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European conference on computer vision*. Springer, 2016, pp. 702–716.

[35] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," *arXiv preprint arXiv:1611.08207*, 2016.

[36] U. Bergmann, N. Jetchev, and R. Vollgraf, "Learning texture manifolds with the periodic spatial GAN," *arXiv preprint arXiv:1705.06566*, 2017.

[37] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2439–2448.

[38] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[39] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[40] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1526–1535.

[41] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1060–1069.

[42] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5907–5915.

[43] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[44] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," *arXiv preprint arXiv:1705.11001*, 2017.

[45] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[47] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International conference on information processing in medical imaging*. Springer, 2017, pp. 146–157.

[48] X. Liu, X. Du, X. Zhang, Q. Zhu, H. Wang, and M. Guizani, "Adversarial samples on android malware detection systems for IoT systems," *Sensors*, vol. 19, no. 4, p. 974, 2019.

[49] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: A survey towards private and secure applications," *arXiv preprint arXiv:2106.03785*, 2021.

[50] J. Jordon, J. Yoon, and M. Van Der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *International Conference on Learning Representations*, 2018.