A Game Theoretic Adversarial Synthetic Data Generation Method to Address Privacy Concerns in the Use of Deep Learning Models for IoT Applications

Abhijit Singh
Department of Electrical and Computer Engineering
National University of Singapore
Singapore
abhijit.singh@u.nus.edu

Biplab Sikdar
Department of Electrical and Computer Engineering
National University of Singapore
Singapore
bsikdar@nus.edu.sg

***Abstract*—Internet of Things (IoT) applications are widely prevalent in the age of Industry 4.0. These applications generate vast amounts of data that need to be processed efficiently. In some of these applications, there may be privacy concerns associated with the generated data being used to train Artificial Intelligence (AI) models. Thus, solutions are needed that can address such problems. This paper develops a methodology to generate diversified synthetic datapoints for IoT datasets, using adversarial machine learning in a game-theoretic setting. The proposed method jointly optimizes two 2-player games being played simultaneously, where the players in each 2-player game have a competing objective. The experimental results on a publicly-available dataset demonstrate that when Machine Learning (ML) models are retrained using only these synthetic datapoints, they perform better than the ML models trained on the original smart meter data, thus alleviating the need for using private data for training ML models.**

***Index Terms*—Adversarial machine learning; IoT; privacy.**

I. Introduction

The advent of Industry 4.0 has exponentially increased the proliferation of Internet-of-Things (IoT). The IoT devices used in these settings generate enormous amounts of raw data, which need to be processed efficiently to truly unlock the benefits of Industry 4.0. Artificial Intelligence (AI) and Machine Learning (ML) tools are ideally placed to meet these needs. In recent years, there have been several use-cases where ML models have been used in IoT applications. These include network traffic profiling [1], pollution monitoring [2] in smart cities, energy management [3] in smart homes, and cardiovascular disease diagnosis [4] in smart health applications.

In many applications, using data collected from end-users to train AI models is going to be a major issue in the near future due to privacy concerns. With increasing data-literacy, a growing number of people are uncomfortable with data generated by them being used by corporate entities for profit-making applications [5]. For example, if we consider a smart home scenario where data from a smart meter is used for Non-Intrusive Load Monitoring (NILM) tasks such as deep learning based appliance classification, a user may not consent to data generated from their home being used to train an AI model, as they may feel it is an invasion of their privacy. In recent years, several jurisdictions have passed laws that restrict the use of data collected from users to train AI systems [6]. In such a scenario, it is important to develop methods that mitigate these privacy concerns.

Recently, variations of Generative Adversarial Networks (GANs) [7] have been used to generate synthetic datapoints. Using synthetic data can alleviate some of the privacy concerns associated with the use of real-world data collected from end-users. However, GANs can be notoriously difficult to train [8]. They may suffer from non-convergence of parameters, mode collapse and vanishing gradients, among other issues. Additionally, they are generally quite sensitive to hyperparameters, and have a large number of hyperparameters that need to be tuned. This can be resource-intensive and cost-prohibitive.

In this paper, we develop a methodology that leverages principles of adversarial datapoint generation in a game-theoretic setting, which avoids the pitfalls that may be encountered when training GANs. We show that when only synthetic datapoints generated using this methodology are used to retrain ML models, they perform better than the ML models trained on the original data. Essentially, we mimic the decision boundary of the original classifier using the synthetic datapoints, eliminating the need to use any real-world data during training. The validation of the proposed method is done on the UK-DALE (United Kingdom-Domestic Appliance-Level Electricity) smart meter dataset [9]. The main contributions of this paper can be summarized as:

- This paper develops a methodology for generating diversified synthetic datapoints for IoT datasets using adversarial methods in a game-theoretic setting.
- The proposed methodology is validated by demonstrating that ML models trained on only these synthetic datapoints can mimic or improve upon the performance of ML models trained on the original data, eliminating the need

to use any real-world data which may have privacy concerns associated with it.

The organization of the rest of the paper is as follows. Section II provides a brief overview of the related work. Section III describes the appliance classification task considered in this work, the dataset used for generating the experimental results, and the proposed method for generating synthetic datapoints. Section IV presents the results of our experiments validating the proposed methodology. Section V discusses the results and Section VI concludes the paper.

II. Related Work

Since their introduction, GANs have been the most popular method for generating synthetic data in the field of computer vision. The authors of [10] used a Deep Convolutional GAN (DCGAN) to generate synthetic prohibitory sign images for a traffic sign recognition task. They augmented the original dataset with the generated synthetic images and found that using the augmented dataset improved the intersection over union (IoU) and accuracy of traffic sign recognition.

In [11], the authors proposed a novel GAN-based pipeline which they called SinGAN-Seg, to create synthetic medical images. Their method was different from traditional GANs because it needed only a single image and the corresponding ground-truth for training. They used style transfer techniques in their pipeline to synthesize additional copies of images with rare medical abnormalities. The authors of [12] presented a hybrid model which consisted of a feature extractor that decoupled feature representations from CT scans of COVID-infected and Non-COVID patients from the local noise. The extracted features were then used to conditionally-generate new CT scans which were used to augment the original dataset, which resulted in an improvement in COVID-19 detection.

While GANs have been used heavily to generate new datapoints in image and audio domains, their use in generating synthetic datapoints in other domains has been comparatively limited. The authors of [13] used an infoGAN [14] to synthesize photoplethysmography (PPG) signals which were then used to augment the existing dataset by inferring blood pressure values corresponding to different age and body mass index (BMI) combinations.

In the IoT domain, the authors of [15] use a GAN architecture to generate labelled synthetic load patterns to train ML models. However, in their work they note the difficulties faced in building such a GAN and its dependence on finely tuned hyperparameters, which makes it difficult to be reused when using different datasets or in different applications. The authors of [16] use a Jacobian-based method to generate synthetic datapoints to train a substitute model while launching a black-box adversarial attack. While the focus of this work is not to use the synthetic datapoints to train an efficient classifier for true datapoints, we nevertheless use it as an additional baseline to compare against the methodology proposed in this paper.

Based on the existing literature, we note that to the best of our knowledge, there is no work that completely replaces original datapoints with adversarially generated synthetic datapoints to train an ML model for IoT environments. While [15] uses a GAN for a similar use-case, our motivation is to avoid the use of GANs to accomplish this task, as stated in Section I. Thus, our work provides novel contributions in a field of growing importance.

III. Methodology

This section presents the proposed methodology for adversarially generating synthetic datapoints in a game-theoretic setting. We begin with a brief overview of the ML task and the datasets used, followed by a description of the proposed methodology.

A. Appliance Classification Task

The surge in adoption of IoT devices has enabled the collection of fine-grained power consumption data from smart meters installed in residential buildings. This data can be utilized for detecting and classifying the appliances being used in households, which can allow further downstream tasks such as improved load forecasting, dynamic pricing mechanisms, consumer profiling and classification, and demand-side management methods.

While traditional NILM techniques use statistical methods such as maximum likelihood estimation or change detection [17] to disaggregate power consumption into that of individual appliances, some papers have proposed ML techniques using deep learning to perform appliance classification using smart meter data [18]. We use these ML models to validate the methodologies proposed in this paper. The authors of [18] preprocess smart meter data from residential buildings to train a deep learning model to perform the appliance classification task. The methodology in [18] follows the standard practice in this area of transforming the problem into a binary classification task, as existing techniques are unreliable in decomposing the profiles of multiple appliances from smart meter data at the same time. An overview of this pipeline is as follows: (i) appliance profiles are extracted from the aggregated data one at a time (with the rest of the input treated as noise), (ii) deep learning models are trained to determine whether a specific appliance is in use or not. Thus, a new model is trained for classifying each appliance.

B. Dataset

The UK-DALE [9] is an open-access dataset collected from five houses in London over several years. It has aggregated and disaggregated appliance-level data recorded at a granularity of six seconds using a smart meter. Five appliances from this dataset are used for the experiments in this paper: hair dryer, kettle, microwave, toaster and vacuum cleaner.

C. Synthetic Data Generation

In game theory, 2-player games are those where two players are playing against each other. Each player has a selection of strategies that they can choose from, and the pay-off associated with each strategy depends on the strategy chosen by the other player. For example, if we represent $PlayerA$'s strategies by A and $PlayerB$'s strategies by B, the pay-off matrices for $PlayerA$ and $PlayerB$ are:

$$U = \begin{array}{c c} & \begin{array}{cc} B_1 & B_2 \end{array} \\ \begin{array}{c} A_1 \\ A_2 \end{array} & \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \end{array}$$

where $u_{ij} = (a_{ij},\, b_{ij})$, and a_{ij} is the pay-off for $PlayerA$ and b_{ij} is the pay-off for $PlayerB$ when $PlayerA$ chooses strategy A_i and $PlayerB$ chooses strategy B_j.

Pure strategies are those where a player only selects one of the strategies available to them. For example, $PlayerA$ only chooses either A_1 or A_2. *Mixed strategies* are those where a player selects multiple strategies available to them, with a particular probability distribution. For example, $PlayerA$ selects A_1 with a probability of 0.3 and A_2 with a probability of 0.7. In this work, we only consider pure strategies.

In game theory, Nash equilibrium is the state when neither player can increase their pay-off by unilaterally changing their strategy. Mathematically, this can be expressed as a strategy (A_i*, B_j*) where:

$$\forall i, a_{i*,j*} \geq a_{i,j*} \ \ ,$$
$$\forall j, b_{i*,j*} \geq b_{i*,j} \ \ .$$

It is well-established that while Nash equilibrium always exists for finite non-cooperative games, there is no guarantee that it uses pure strategies. This is important because in this work we only consider pure strategies. Thus, we will later describe how at least one partial Nash equilibrium will exist in our setting, even though we impose the constraint of allowing only pure strategies. By partial Nash equilibrium, we mean the scenario where the Nash equilibrium condition for $PlayerA$ ($\forall i, a_{i*,j*} \geq a_{i,j*}$) is satisfied, but there is no requirement for the Nash equilibrium condition for $PlayerB$ ($\forall j, b_{i*,j*} \geq b_{i*,j}$) to be satisfied.

In adversarial machine learning, to generate an adversarial example for a test datapoint x_{test}, we can frame the objective function as:

$$\min \left\|(x_{test} - x^{'}_{test})\right\|_p , \qquad (1)$$

such that,

$$\max \{p(y_{test} = k_i | x^{'}_{test}, \theta^*)\} = p(y_{test} = k_{adv} | x^{'}_{test}, \theta^*)$$

where p is the chosen L_p norm, k_{adv} is the class prediction as desired by the adversary (with $k_i = k_1, k_2, \cdots, k_c$ for c possible classes), θ^* are the final weights, and $x^{'}_{test}$ is the adversarial example.

Proposed mechanism: In this work, we introduce a game-theoretic setting while working towards an adversarial objective. Instead of generating one adversarial example at a time, we generate two adversarial examples simultaneously. Each of the two adversarial examples is initialized with a perturbed version of the true datapoint. Different perturbations are used for the two adversarial examples, so that they do not have the same starting point. When considering a pair consisting of the true datapoint and one of the adversarial examples, the goal remains as described in Equation (1). However, there is an additional objective function that is added to it, when we consider the pair of adversarial examples being generated:

$$\max \left\|(x^{'(1)}_{test} - x^{'(2)}_{test})\right\|_p \qquad (2)$$

where $x^{'(1)}_{test}$ and $x^{'(2)}_{test}$ are the two adversarial examples that are being generated.

We can view this setting as two 2-player non-zero sum games occurring simultaneously (one game for each adversarial example). In each game, the pay-off for $PlayerA$ can be the L_p norm between the true datapoint and the adversarial example (Equation (1)), and the pay-off for $PlayerB$ can be the L_p norm between the two adversarial examples (Equation (2)).

To be consistent with game theory terminology, the pay-offs are to be maximized. So, we can take the negative of the L_p norm between the true and adversarial example (because we want to minimize this norm) as the pay-off for $PlayerA$. The pay-off matrix U can then be constructed as shown:

$$U = \begin{array}{c c} & \begin{array}{cccc} B_1 & B_2 & \cdots & B_J \end{array} \\ \begin{array}{c} A_1 \\ A_2 \\ \vdots \\ A_I \end{array} & \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1J} \\ u_{21} & u_{22} & \cdots & u_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ u_{I1} & u_{I2} & \cdots & u_{IJ} \end{bmatrix} \end{array} .$$

Here, the strategies A_1 to A_I and B_1 to B_J correspond to the different positions of the adversarial example in the high-dimensional space, during the adversarial example generation process. Each position has a different norm value (with respect to the true datapoint and the other adversarial example that is being generated simultaneously). As mentioned earlier, two games are occurring simultaneously (one for each adversarial example), and they are linked by the fact that both games will have identical pay-offs associated with strategies B_1 to B_J (because the L_p norm between the two adversarial examples (Equation (2)) will be the same for both).

The pay-off matrix U (more specifically, U_1 and U_2 for the two simultaneous games) can be of an arbitrary size, as the datapoint has infinitely many positions that it can take up in the high-dimensional space. However, this is not an issue since we can choose to use suitably discretized positions in the vicinity of the true datapoint and the decision boundary.

An important fact to note is that in this pay-off matrix U, not all strategies result in fulfilling the main adversarial objective (that the adversarial example has to be classified in the desired class). Only one or more elements will fulfill that condition. Our goal then becomes to find the element that meets the main adversarial objective, and has the highest pay-off among all elements that meet the adversarial objective. This element can be viewed as a partial Nash equilibrium under the specified conditions, because the Nash equilibrium condition is satisfied only for $PlayerA$ ($\forall i, a_{i*,j*} \geq a_{i,j*}$), and it need not satisfy the Nash equilibrium condition for $PlayerB$ ($\forall j, b_{i*,j*} \geq b_{i*,j}$).

This element can be found by optimizing the final objective function, which can be modelled by combining Equations (1) and (2):

$$\min \left\|(x_{test} - x_{test}^{'(1)})\right\|_p, \tag{3}$$
$$\min \left\|(x_{test} - x_{test}^{'(2)})\right\|_p,$$
$$\max \left\|(x_{test}^{'(1)} - x_{test}^{'(2)})\right\|_p \text{ such that,}$$
$$\max\{p(y_{test} = k_i | x_{test}^{'(1)}, \theta^*)\} = p(y_{test} = k_{adv} | x_{test}^{'(1)}, \theta^*)$$
$$\text{and,}$$
$$\max\{p(y_{test} = k_i | x_{test}^{'(2)}, \theta^*)\} = p(y_{test} = k_{adv} | x_{test}^{'(2)}, \theta^*).$$

To solve these constrained equations, we use approximations that can work almost as well as the theoretically optimal solution, much like how in complex DNNs, a local minimum may suit the application's needs well-enough, without requiring to reach the global minimum. Optimizing Equation (3) simultaneously produces the approximations of the partial Nash equilibrium for both the 2-player games.

To explain these approximations, we describe the algorithm used to generate the synthetic datapoints (to better convey their purpose, the adversarial examples are henceforth referred to as synthetic datapoints). The algorithm begins with initializing the pair of synthetic datapoints that will be created. The initial datapoints are perturbed versions of the input datapoint, with the level of perturbation chosen as desired by the user. After setting the learning rate and maximum number of iterations allowed, a loop begins and the completion check is done at the top. The predicted label of the current value of the synthetic datapoints is extracted from their model score (step 8). If the predicted label is the same as the target label for both the synthetic datapoints, then we have achieved the adversarial objective (Equation (3)) and we exit the loop.

If the predicted label is the same as the target label for only one of the synthetic datapoints, then we move the synthetic datapoint that hasn't achieved the adversarial objective away from the synthetic datapoint that has already achieved its adversarial objective (as it has achieved its adversarial objective, it's left unmodified). After this, we extract the score of the target class for the synthetic datapoint which is yet to achieve its adversarial objective, and compute backpropagation gradients using the score of the target class as the beginning point. The gradient update for the synthetic datapoint is then normalized for stability, and this normalized gradient update is used to modify the current value of the synthetic datapoint.

If neither of the synthetic datapoints have predicted labels that match their target class, then both the synthetic datapoints are moved away from each other. The rest of the process for both the datapoints remains the same as described in the previous paragraph. The score of the target class is extracted for both the synthetic datapoints, and backpropagation gradients are computed and normalized. These are then used to modify the respective current values of the synthetic datapoints. This

Algorithm 1 Generating two synthetic datapoints for every input datapoint

Input: Original datapoint (*X*), Target class (*tar_y*), Trained original model (*orig_mod*)
Outputs: Synthetic datapoints (*X_syn_1* and *X_syn_2*)
Other variables: Initial perturbations applied (*init_perturb_1* and *init_perturb_2*), Maximum iterations allowed (*max_iter*), Learning Rate (*lr*), Class scores generated by *orig_mod* (*orig_mod_sc*), Predicted label (*label_pred*), Score of target class (*sc_tar*), Gradient of *X_syn_1* and *X_syn_2* (*dx_syn_1* and *dx_syn_2*)

```
1:  function CREATE_SYN_DATA(X, tar_y, orig_mod)
2:      Use evaluation/inference mode of orig_mod
3:      Initialize the synthetic datapoints
        X_syn_1 = X × init_perturb_1
        X_syn_2 = X × init_perturb_2
4:      Set lr and max_iter
5:      for $i <$ max_iter do
6:          Compute orig_mod_sc_1 of X_syn_1
7:          Compute orig_mod_sc_2 of X_syn_2
8:          Extract label_pred_1 from orig_mod_sc_1 and
            label_pred_2 from orig_mod_sc_2
9:          if (label_pred_1 and label_pred_2) = tar_y then
10:             break from for-loop
11:         else if label_pred_1 = tar_y and
            label_pred_2 != tar_y then
12:             Move X_syn_2 away from X_syn_1
13:             Extract sc_tar_2 from orig_mod_sc_2
14:             Perform backpropagation on sc_tar_2
15:             Extract gradient update of X_syn_2 (dx_syn_2)
16:             Gradient update normalization:
                  r_2 = lr × (dx_syn_2/norm(dx_syn_2))
17:             Update X_syn_2 with r_2:
                  X_syn_2 += r_2
18:             Clear current gradients
19:         else if label_pred_1 != tar_y and
            label_pred_2 = tar_y then
20:             Move X_syn_1 away from X_syn_2
21:             Extract sc_tar_1 from orig_mod_sc_1
22:             Perform backpropagation on sc_tar_1
23:             Extract gradient update of X_syn_1 (dx_syn_1)
24:             Gradient update normalization:
                  r_1 = lr × (dx_syn_1/norm(dx_syn_1))
25:             Update X_syn_1 with r_1:
                  X_syn_1 += r_1
26:             Clear current gradients
27:         else
28:             Move X_syn_1 and X_syn_2 away from
                each other
29:             Extract sc_tar_1 from orig_mod_sc_1
                and sc_tar_2 from orig_mod_sc_2
30:             Perform backpropagation on sc_tar_1
                and sc_tar_2
31:             Extract gradient updates of X_syn_1 (dx_syn_1)
                and X_syn_2 (dx_syn_2)
32:             Gradient update normalization:
                  r_1 = lr × (dx_syn_1/norm(dx_syn_1))
                  r_2 = lr × (dx_syn_2/norm(dx_syn_2))
33:             Update X_syn_1 and X_syn_2:
                X_syn_1 += r_1
                X_syn_2 += r_2
34:             Clear current gradients
35:         end if
36:     end for
37: end function
38: Return X_syn_1 and X_syn_2
```

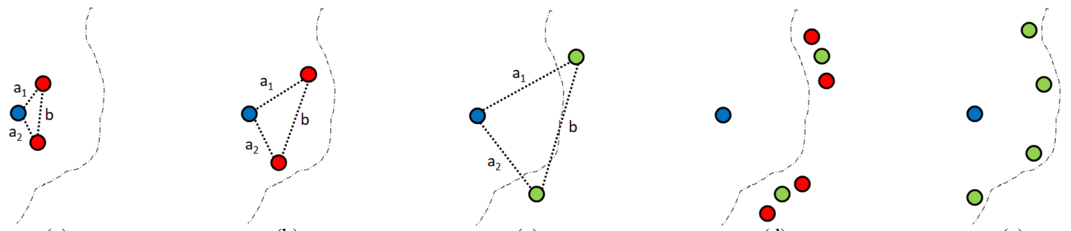

Fig. 1: Synthetic datapoint generation process. (a) The original datapoint (blue) is sent as input to the algorithm, and perturbed versions of the original datapoint are used as initial points of the synthetic datapoints (red). (b) In every subsequent iteration, the synthetic datapoints move towards the decision boundary, while also moving farther apart from each other (distance b increases). (c) The algorithm ends when the synthetic datapoints have crossed the decision boundary and achieved their adversarial objective (green), in a way that a_1 and a_2 are as low as possible. (d) After the algorithm is run once, we get two synthetic datapoints (green) for the original datapoint (blue), as shown in (c). These points are then used as inputs in the algorithm again, (e) and we eventually get four synthetic datapoints (the red circles in (d) cross over the decision boundary. These are the new green circles in (e)).

process continues until either the adversarial objective is satisfied (step 9), or we reach the maximum number of iterations.

The main component of r (referred to as r_1 and r_2 in Algorithm 1) is the gradient update for the respective datapoint. This is obtained by backpropagating gradients. An important point to note is that while in general backpropagation of gradients starts from the cost function, in this case it starts from the score of the target class (steps 14, 22, and 30). Thus, r can be computed in the following manner:

$$r = learning\_rate \times \frac{dx}{norm(dx)} \tag{4}$$

$$dx = \frac{\partial F}{\partial a^{(l+1)}} \frac{\partial a^{(l+1)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l-1)}} \cdots \frac{\partial z^{(1)}}{\partial a_i^{(1)}} \tag{5}$$

for $i = 1, \cdots, d$ for d-dimensional input. In Equation (4), dx will be dx_syn_1 or dx_syn_2 depending on whether we are considering X_syn_1 or X_syn_2. In Equation (5), F is the score of the target class. $z^{(l)}$ refers to the weighted sum of inputs ($\sum_i w_i a_i$) computed in the last hidden layer l (assuming there are a total of l hidden layers in the network). $a^{(l)}$ refers to the activations generated as output by hidden layer $l-1$ (thus, $a^{(l)} = g(z^{(l-1)})$, where g is the activation function used in that hidden layer). $a^{(1)}$ is simply the input datapoint to the model (X_syn_1 or X_syn_2). The perturbations to be applied to the adversarial datapoint in that iteration can be obtained by using Equations (4) and (5).

The magnitude of the update r decides the actual values of the pay-off matrix U. The pay-offs of each subsequent strategy of $PlayerA$ and $PlayerB$ can be obtained after each iteration of the loop in step 5. These are the "suitably discretized position" that were mentioned earlier in this section.

Algorithm 1 allows us the luxury of not needing to compute the entire pay-off matrix U (also recall that as we are playing two simultaneous games, one for each synthetic datapoint, there will be two different pay-off matrices U_1 and U_2, for each of the synthetic datapoints, respectively). We proceed with the algorithm, and in each iteration we find one value of the pay-off matrix U. We stop the algorithm as soon as we arrive at the first pay-off which also fulfills the adversarial objective.

This will be our partial Nash equilibrium point in a pure strategy, because according to our definition of the pay-off for $PlayerA$, we want the distance between the synthetic datapoint and original datapoint to be as low as possible. Any subsequent step would increase this distance, because the synthetic datapoint will be moved farther away from the original datapoint. We may also achieve a better partial Nash equilibrium if the other synthetic datapoint hasn't achieved its adversarial objective in the same iteration. This is because in each subsequent step, the other synthetic datapoint will be moved farther away from the stationary synthetic datapoint, which increases the pay-off value for $PlayerB$ (while the pay-off for $PlayerA$ remains constant) for both the games being played.

Finally, we use the two generated synthetic datapoints as individual inputs to Algorithm 1 again, to eventually end up with four synthetic datapoints. This repetition is done so that the final synthetic datapoints will be very close to the decision boundary, and also relatively far away from each other, increasing their diversity. This can potentially improve how well the new retrained classifier's decision boundary mimics that of the original classifier. A high-level visual summary of this process is explained in Figure 1.

IV. Experimental Results

In this section, we present the results of our experimental evaluation of the synthetic data generation methodology presented in Algorithm 1. As explained in Section III-C and visually depicted in Figure 1, when Algorithm 1 is used to generate synthetic datapoints, every input datapoint results in the generation of four synthetic datapoints. As a result, to ensure that the original ML models and the retrained ML models are trained on an equal number of datapoints, we randomly select 25% of the original training datapoints

TABLE I: Comparison of F1 scores obtained on the test set, of original models retrained on the synthetic datasets generated by different techniques.

Appliance	F1 scores			
	Original [18]	GAN method [15]	Jacobian [16]	Ours
Hair Dryer	0.65	0.54	0.60	**0.70**
Kettle	0.65	0.57	0.58	**0.70**
Microwave	0.65	0.57	0.61	**0.71**
Toaster	0.68	0.61	0.59	**0.75**
Vacuum Cleaner	0.70	0.49	0.48	**0.76**

(while preserving the class ratios) to be used as inputs to the algorithm.

To compare the methodology proposed in this work with existing techniques, we use the methods proposed in [15] and [16]. As described in Section II, the authors of [15] use a GAN architecture to generate synthetic datapoints, while the authors of [16] use a Jacobian-based method to generate synthetic datapoints. We use both methods to generate the same number of synthetic datapoints as those present in the original dataset.

Table I compares the performance of the retrained ML models with that of the original models. F1 scores are used as the evaluation metric, and the test set used for obtaining the final performance remains the same in every case.

We note that for every appliance, the proposed methodology results in a better performance than [15] and [16].

V. Discussion

The results presented in Table I demonstrate the effectiveness of the proposed synthetic datapoint generation mechanism. The GAN method [15] suffers from several of the issues discussed in Sections I and II, especially mode collapse. This is the primary reason for its poor performance on this task. Similarly, the Jacobian method [16] is unable to get enough diversity in its synthetic datapoints to accurately reconstruct the original decision boundary, leading to a poorer performance than the original models.

In contrast, using only synthetic datapoints generated by Algorithm 1 to retrain the ML models results in decision boundaries that have a better performance on the test set than the original models. This is because the adversarial game-theoretic setting explained in Section III-C ensures that a diverse set of synthetic datapoints are generated, and that they lie close to the decision boundary of the original models, as illustrated in Figure 1. This allows the decision boundaries of the newly trained classifiers to closely mimic those of the original models. Thus, we can get more private models that have not been trained on real-world data collected from end-users, but have at least an equal or better generalization performance on the real-world test set.

VI. Conclusion

This paper developed a methodology for generating diversified synthetic datapoints for IoT datasets using an adversarial strategy in a game-theoretic setting. The original ML models were retrained using only these diverse synthetic datapoints, and the results validated the proposed methodology by showing that the retrained classifiers had a better performance than the ML models trained on the original data. This could help mitigate privacy concerns about data collection amongst end-users, as they could be assured that their data would not be used to train ML models directly.

References

[1] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "Iot network traffic classification using machine learning algorithms: an experimental analysis," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 989–1008, 2021.

[2] P. Ferrer-Cid, J. M. Barcelo-Ordinas, and J. Garcia-Vidal, "Graph signal reconstruction techniques for iot air pollution monitoring platforms," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 25 350–25 362, 2022.

[3] S. S. Shuvo and Y. Yilmaz, "Home energy recommendation system (hers): A deep reinforcement learning method based on residents' feedback and activity," *IEEE Transactions on Smart Grid*, vol. 13, no. 4, pp. 2812–2821, 2022.

[4] D. Zhang, X. Liu, J. Xia, Z. Gao, H. Zhang, and V. H. C. de Albuquerque, "A physics-guided deep learning approach for functional assessment of cardiovascular disease in iot-based smart health," *IEEE Internet of Things Journal*, 2023.

[5] O. Lucas, M. Sokalski, and R. Fisher, "Corporate data responsibility: Bridging the consumer trust gap," *KPMG Advisory*, 2021.

[6] G. D. P. Regulation, "General data protection regulation (gdpr)–official legal text," *Gen Data Prot Regul*, 2016.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, and S. Ozair, "& bengio, y.(2014)," *Generative adversarial nets*, vol. 27, 2014.

[8] H. Chen, "Challenges and corresponding solutions of generative adversarial networks (gans): a survey study," in *Journal of Physics: Conference Series*, vol. 1827, no. 1. IOP Publishing, 2021, p. 012066.

[9] J. Kelly and W. Knottenbelt, "The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes," *Scientific data*, vol. 2, no. 1, pp. 1–14, 2015.

[10] C. Dewi, R.-C. Chen, Y.-T. Liu, and S.-K. Tai, "Synthetic data generation using dcgan for improved traffic sign recognition," *Neural Computing and Applications*, vol. 34, no. 24, pp. 21 465–21 480, 2022.

[11] V. Thambawita, P. Salehi, S. A. Sheshkal, S. A. Hicks, H. L. Hammer, S. Parasa, T. d. Lange, P. Halvorsen, and M. A. Riegler, "Singan-seg: Synthetic training data generation for medical image segmentation," *PloS one*, vol. 17, no. 5, p. e0267976, 2022.

[12] H. P. Das, R. Tran, J. Singh, X. Yue, G. Tison, A. Sangiovanni-Vincentelli, and C. J. Spanos, "Conditional synthetic data generation for robust machine learning applications with limited pandemic data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, 2022, pp. 11 792–11 800.

[13] B.-F. Wu, L.-W. Chiu, Y.-C. Wu, C.-C. Lai, and P.-H. Chu, "Contactless blood pressure measurement via remote photoplethysmography with synthetic data generation using generative adversarial network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2130–2138.

[14] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.

[15] S. El Kababji and P. Srikantha, "A data-driven approach for generating synthetic load patterns and usage habits," *IEEE Transactions on Smart Grid*, vol. 11, no. 6, pp. 4984–4995, 2020.

[16] J. Wang and P. Srikantha, "Stealthy black-box attacks on deep learning non-intrusive load monitoring models," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3479–3492, 2021.

[17] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[18] M. A. Devlin and B. P. Hayes, "Non-intrusive load monitoring and classification of activities of daily living using residential smart meter data," *IEEE transactions on consumer electronics*, vol. 65, no. 3, pp. 339–348, 2019.