IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 July 2023, accepted 3 August 2023, date of publication 11 August 2023, date of current version 17 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3304541

RESEARCH ARTICLE

Bayesian Optimization-Driven Adversarial Poisoning Attacks Against Distributed Learning

MARIOS ARISTODEMOU[1], (Graduate Student Member, IEEE), XIAOLAN LIU[2], (Member, IEEE), SANGARAPILLAI LAMBOTHARAN[2], (Senior Member, IEEE), AND BASIL AsSADHAN[3], (Senior Member, IEEE)

[1]Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, LE11 3TU Loughborough, U.K.
[2]Institute for Digital Technologies, Loughborough University London, E20 3BS London, U.K.
[3]Department of Electrical Engineering, King Saud University, Riyadh 11495, Saudi Arabia

Corresponding author: Marios Aristodemou (m.aristodemou@lboro.ac.uk)

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/X012301/1 and Grant EP/X04047X/1, and in part by the International Scientific Partnership Program of King Saud University under Grant ISPP-18-134(2).

ABSTRACT Metaverse is envisioned to be the next-generation human-centric Internet which can offer an immersive experience for users with a broad application in healthcare, education, entertainment, and industries. These applications require the analysis of massive data that contains private and sensitive information. A potential solution to preserving privacy is deploying distributed learning frameworks, including federated learning (FL) and split learning (SL), due to their ability to address privacy leakage and analyze personalised data without sharing raw data. However, it is known that FL and SL are still susceptible to adversarial poisoning attacks. In this paper, we analyse such critical issues for the privacy-preserving mechanism in Metaverse services. We develop a novel poisoning attack based on Bayesian optimisation to emulate the adversarial behaviour against FL (BO-FLPA) and SL (BO-SLPA) which is important for the development of effective defense algorithms in the future. Specifically, we develop a layer optimisation method using the intuition of black-box optimisation with assuming that there is a function between the prediction's uncertainty and layer optimisation parameters. The result of this optimisation provides the optimal weight parameters for the hidden layer, such as the first or the second layer for FL, and the first layer for SL. Numerical results demonstrate that in both FL and SL, the poisoned hidden layers have the ability to increase the susceptibility of the model to adversarial attacks in terms of prediction with low confidence or having a larger deviation of the probability density function of the predictions.

INDEX TERMS Adversarial machine learning (AdvML), federated learning (FL), metaverse, poisoning attacks, split learning (SL).

I. INTRODUCTION

Metaverse is a buzzword that first appeared in the science fiction novel *Snow Crash* written in 1992 by Neal Stephenson [1]. Metaverse is described as a personified version of the Internet to create a virtual world that the users can explore around with the assistance of augmented reality (AR), virtual reality (VR), and the tactile Internet [2]. Several tech giants have invested billions towards realising Metaverse that has the potential to revolutionise healthcare, education, entertainment, and industries [3]. The adoption of Metaverse embraced by the fact that a significant portion of the global population carries out conventional activities such as meetings, training, examinations, and entertainment in the virtual domain, prompting various stakeholders to invest in VR technologies to connect people across the world. This has

The associate editor coordinating the review of this manuscript and approving it for publication was Theofanis P. Raptis.

been enhanced by the deployment in reliable communications with low latency, such as 5G and emerging 6G, by relying on the deployment of enhanced Mobile Broadband (eMBB) and ultra-reliable low latency communication (URLLC) [3]. Consequently, Metaverse is a revolutionary technology that is expected to connect several advanced technologies together and change the way how people communicate and live [2]. For example, it can facilitate digital twins to produce a digital replica of the physical world, and VR, AR, and extended reality (XR) to produce an immersive 3D environment [1]. All the aforementioned applications rely on the analysis of the generated massive data that contains sensitive information which requires security, privacy, and data protection using machine learning (ML) and especially deep learning (DL) algorithms.

The existing potential solution to preserve security and privacy when extracting useful information from private data is to deploy a distributed learning framework. Especially, the classical distributed learning method, federated learning (FL), has been widely adopted as an effective method to address security and privacy concerns by reducing the exchange of private information inside a network. In addition, distributed learning is suitable for reducing the computational capabilities required for training and inference for ML and DL algorithms. The two state-of-the-art distributed learning algorithms used for a variety of applications are FL [4], [5] and split learning (SL) [6], [7].

FL has been investigated as a promising mechanism of local training and global aggregation, which only shares the model parameters instead of the raw data between the users and the FL server. FL allows users to train their models with their own private data, and it limits the amount of exposed data inside the network. Once the users finished their local training, the FL server aggregates all the selected local models to update a global model, which could also benefit the users that do not have adequate data and computational resources to train their local model [5]. The deployment of the FL framework relies on the fact that all the participating devices have the computational capability to perform the local training [8]. Therefore, FL can be used to support the deployment of intelligent Metaverse services at the network edge, with the benefit of reducing the computational resources required for the edge devices. There is a variety of applications where FL has been used, such as object recognition [4], collaborative channel estimation [8], autonomous vehicles [9], and online gaming [10].

The learning architecture of SL differs from that of FL [6], [7]. Instead of training the whole ML model locally, the model is split into several sub-models (the layer that splits the model is called the cut layer). The sub-models are distributed to different entities (user and server), and only the smashed data of the cut layer is shared between the user and the server [6]. SL allows users to perform fewer computational tasks due to the processing of fewer neural network layers. Importantly, SL prevents personalised data leakage as only the smashed data is exchanged.

However, early studies have shown that ML models are susceptible to adversarial attacks [11], [12]. Adversarial attacks are part of adversarial machine learning (AdvML) which is the study of vulnerabilities occurred by data manipulation with the scope of evading or poisoning ML or DL and producing erroneous output [13], [14]. Recent studies suggest that FL is also susceptible to adversarial attacks. For instance, FL is susceptible to poisoning attacks in the presence of malicious users in the network [15], [16]. Three classes of adversarial attacks against FL have been studied, including data poisoning [16], [17], model poisoning [18], [19] and backdoor attacks [15], [20]. Apart from the FL, the SL is also susceptible to adversarial attacks. There is a limited amount of literature on adversarial attacks against SL which are characterised as inference attacks or privacy leakage. The attack model includes modifying the leaked label and gradient [21], [22], and reconstructing the leaked data [23].

In Metaverse, privacy leakage and violation of digital footprints and trusted entities raise a big concern. Therefore, there is a surge of interest in distributed learning techniques, such as FL or SL, for privacy protection during the training of ML and DL models [1]. For instance, the violation of trusted entities may affect the Industrial Internet of Things (IIoT) where the adversaries change the controllability of the production chain. Adversaries may gain critical information about the network by gaining access to the devices. Another critical issue is that the adversary may gain users' data that are privacy-sensitive (e.g., patients' data and social account data). In this work, we develop two attack mechanisms against both FL and SL to emulate the attackers' behaviours in order to construct defenses against adversarial activities in the future.

This paper studies the idea of black-box optimisation for poisoning FL and SL. We assume that there is a mapping function that describes the uncertainty for a target distribution and the modification of the most important features. First, we propose a poisoning strategy against FL in (Section V) for model poisoning which advances the works by [20] and [15]. Specifically, instead of modifying the direction of gradients or inducing extra neurons, we use Bayesian optimisation (BO) to find the optimal solution to modify the model weights for inducing high classification uncertainty in the most significant features. Second, we propose a poisoning strategy against SL in (Section VI-B) for model poisoning which advances the work by [22]. Here, instead of modifying labels and gradients, we modify the model weights such that the most significant features of the server-side model are poisoned by the back-propagation loss. We use BO techniques to find the optimal weight distributions which will change the most significant features.

We provide the following contributions:

1) We propose a novel model poisoning method against both FL and SL (sections V and VI), where we use a black-box optimisation to identify the best suitable modification for the most significant features in the hidden layer. To the best of our knowledge, this is the first work to introduce BO as a method of searching for the optimal model parameters.
2) In FL (section V), we consider that there is a function that describes the relationship between uncertainty and neural network's modification factor IV-A. We quantify the uncertainty using Kullback-Leibler (KL) divergence and aim to maximise the uncertainty and within the optimisation constraint such that the weight distribution does not deviate from the global model. We demonstrate that layer optimisation with BO is an effective way to construct an untargeted poisoning attack against the FL server.
3) In SL (section VI), we modify the weight distribution through the calculation of the similarity for the changes caused by the neural network optimisation, and the gradients to gather information about the most significant features used by the server-side model. We consider there is a function that describes the weight distribution and the similarity in each iteration IV-B.
4) Through the comparison of the accuracy and plotting the confidence levels, we demonstrate that both of the attacks increase the susceptibility of the model against adversarial attacks VII. We show that the probability density function has a large deviation and thus the global models have low confidence during testing prediction.

This paper is organised as follows: Section II introduces the theoretical background of FL and SL, followed by a review of related works in Section III. In Section IV, we formulate the problem of adversarial poisoning attack against FL and SL, followed by the proposed attack algorithms for FL and SL in Section V and VI, respectively. We evaluate the proposed algorithms through several simulations on MNIST and CIFAR10 datasets in Section VII. Finally, conclusions are drawn in Section VIII.

II. PRELIMINARIES

A. FEDERATED LEARNING

FL is an emerging distributed privacy-preserving algorithm that enables users to train their models using their private and personalised data by collaborating with a centralised FL server. FL is an enabler for a wide range of applications emerging in network edge, such as Metaverse services, VR/AR and XR, where users may have limited computational and communications resources.

The training procedure of FL is illustrated in Fig. 1, where there are two main entities in the training; the server and user [4]. The FL server is responsible for coordinating the training process by aggregating the users' weights and gradients, authorising which users will take part in the training

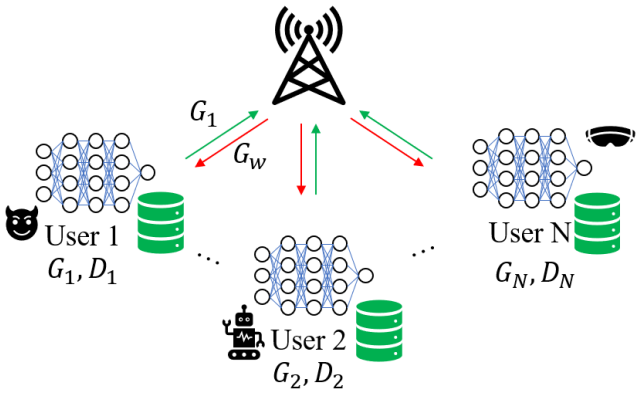


FIGURE 1. The training process of FL.

by evaluating their model training performance during each round of training. The server maintains a global model and aims to train the global model ($G(w)$) using local data ($\mathcal{D}_i$) distributed over N users ($\mathcal{G}$) and minimise a global objective function as shown in (1) [8]. The users train their local models with the local data and then send the updated model gradients to the server for model aggregation. After aggregation, the server checks if the global objective function has reached convergence so that the training can stop [8].

$$\min_{w} \mathcal{G}(w) \triangleq \sum_{i=1}^{N} \frac{\mathcal{D}_i}{\mathcal{D}} \mathcal{G}_i(w), \quad \mathcal{D} = \sum_{i=1}^{N} \mathcal{D}_i \tag{1}$$

In addition to the classical aggregation function, federated averaging (FedAvg), many other aggregation methods have also been studied to achieve FL and they used different functions according to the properties of the FL network. For example, FedAvg is used for homogeneous networks with independent and identically distributed (i.i.d) data [24], FedPlus is used for the case that lack of i.i.d data [25], and Krum is used for adversarial resilience [26].

B. SPLIT LEARNING

SL is a common technique for training deep neural networks (DNNs) in a distributed fashion, especially for users with low computational resources and limited labeled data [6]. SL is useful to train ML models for data-sensitive applications, such as treatment planning, diagnosis, and prognosis [27]. In [7], the authors use SL to train the DNN by only sharing the smashed data and gradients between the client and the server for reducing the exposure of raw patient data.

The key feature of SL is splitting the neural network into two sub-networks. None of the participants have knowledge of the complete parameters of the neural network held by both the server and client. Each client trains a part of the neural network up to a specific layer. This layer is called the cut layer. SL has three basic steps, local inference, server inference and optimisation, and client optimisation [28]. In local inference, we feed forward the local data to the cut layer and transmit the smashed data from the output of the cut layer

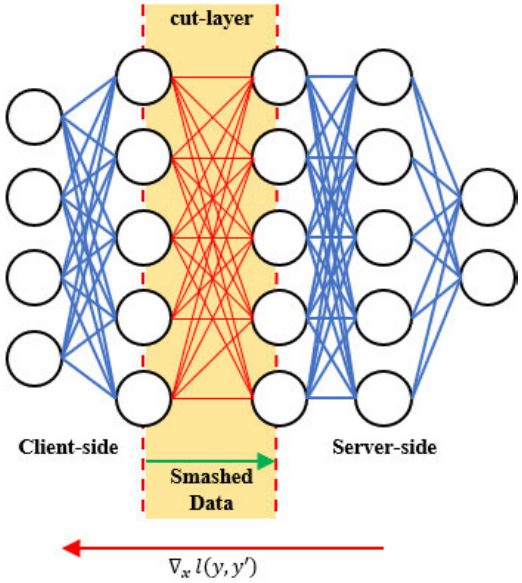


FIGURE 2. Split learning training process.

to the server. Then, the server feeds the received smashed data to the remaining layers of the neural network, computes the loss of the prediction, and then back-propagates the loss through the neural network for network optimisation. The back-propagated loss of the smashed data is sent back to the client side to allow the client to optimise its sub-network. Note the loss computation can be performed on the client side if it chooses not to share the labels to the server [29]. The training process of SL ensures that the private data of the user is not exposed to the server or any other participants, while only the gradients and the output of the cut layer are shared during the training process.

C. ADVERSARIAL MACHINE LEARNING

Data is a vital factor in training ML models. In most cases, if the data is not well-structured, it can cause the trained ML model to make wrong decisions. Hence, it is critically important to determine how the training data affects the mechanisms of the trained ML model [11]. The study of vulnerabilities by data manipulation through evasion or poisoning is called AdvML [12]. AdvML describes how the vulnerabilities occur due to the presence of adversarial samples or a numerically unstable mechanism of ML model [13]. The vulnerabilities can be discovered through the analysis of adversarial examples [30], the algorithmic design space [12], and the numerical stability [31].

There are two categories of AdvML attacks, evasion and poisoning attacks [14]. Evasion attacks aim to evade ML classifier's decisions through the manipulation of input data during testing or production [32]. Poisoning attacks aim to poison an algorithm through the manipulation of training mechanisms and datasets. Poisoning attacks are more common in existing research, which can use online learning or reinforcement learning to retrain or adapt their decision boundary according to the environment that they act on [33]. In each category, there are two sub-categories, targeted and untargeted attacks [34]. The targeted attack occurs when the aim of the adversary is to manipulate the data or the model in such a way as to force the model to classify the data in favour of a specific class. Instead, in an untargeted attack, the adversary aims to mislead the classifier to any class.

III. RELATED WORK

A. ATTACKS ON FEDERATED LEARNING

Several researchers have shown that FL is susceptible to adversaries in both targeted and untargeted attacks [35]. The AdvML attacks are characterised as poisoning attacks [17]. In the literature, there are three types of poisoning attacks against FL: data poisoning [16], [17], [36], model poisoning [18], [19], and backdoor attack [15], [20].

Starting from data poisoning attacks, the adversary is aiming to poison a model through the manipulation of the training data in order to infer the decision boundary. The same methodologies are used in centralised ML or DL. The authors in [17] have used the wrong target labels in order to poison the dataset with the wrong target class. This results in misleading the training of the local model. As the percentage of data poisoning is increasing, the accuracy of the global model is reduced. A similar idea has also been adopted by [36], the authors modified the labels of malicious samples that are generated by a generative adversarial network (GAN). In [16], a malicious model is trained with adversarial examples that point the decision boundary in the wrong direction. Those adversarial examples are generated using the momentum iterative Fast Gradient Sign Method (FGSM) which takes into consideration the gradient direction. However, in their work, the authors do not present clearly how the labels are modified.

A more pragmatic poisoning attack is model poisoning which aims to modify the weights and the gradients such that the decision boundary of the global model will be affected. In [18], the authors manipulate the local model parameters during the learning process such that the global model has a high error rate during testing. Their methodology is transferable in four FL aggregation methods. Even though their method is effective, it is computationally expensive. A less computationally expensive optimisation is proposed in [19], which uses an optimisation technique to embed poisonous neurons into the redundant parts of the neural network. The empirical evaluation of the algorithm in [19] shows that their proposed method outperforms the backdoor attacks in terms of effectiveness, persistence, and robustness.

In FL, an important aspect of attacking the ML model is to gain unauthorised access to specific data or mislead the ML model to make false predictions. This type of attack is called a backdoor attack. In network security, a backdoor attack aims to gain unauthorised access to an application, server, or network through the use of software that is perceived as benign. Similarly, in FL, when an adversary conducts a

backdoor attack, it aims to infer the decision boundary of the global model in order to misclassify the targeted false label while classifying the benign inputs correctly [35]. The authors in [20] proposed a backdoor attack that trains a backdoored model similar to the global model and aims to replace the global model through the reduction of the learning rate and customising the loss function to avoid the server's anomaly detection. However, such an attacking method is successful only when the global model is near convergence. Another approach in [15], presents an attack methodology that achieves the attack goal in the early stages of the training of the global model, i.e. well before its convergence. This uses two stealth metrics in order to avoid anomaly detection.

B. SPLIT LEARNING

In SL, there is only a limited amount of successful AdvML attacks due to limited information on both the client side and the server side. While it preserves the privacy and security of both client and server, it limits the testing criteria for both of the sub-models. Most of the literature is concentrated on privacy leakage where an adversary hijacks the communication between the server and client and steals the labels and the gradients. The adversary can infer the leakage information in order to mislead both sub-models. Therefore, the AdvML attacks are characterised as inference attacks or privacy leakage. Most of the researchers have focused on the research of label or gradient leakage [21], [22], [37], and how an adversary can reconstruct the leaked data [23], [38].

Beginning with the label or gradient leakage, the authors of [37] produced a mathematical analysis using similarity measurements to find potential label leakage inside the gradients or smashed data, and used them to produce a clustering label attack. They evaluate the attack performance in different epochs, different positions of the cut layer, and the robustness under different batch sizes. Most evaluations showed that the attack achieves 100% attack accuracy. However, the parameters of the neural networks and adversarial settings are unclear in their evaluation. A different perspective presented in [22] is to create a scoring function based on the area under the curve (AUC) of receiver operating characteristics (ROC) and construct a privacy loss quantification function. This summarises the predictive performance of a classifier during SL where both the client and server have limited information. Consequently, the authors in [22] have used these metrics to construct two attacks based on the gradients provided by the server and evaluated using a heuristic random perturbation method and a method for searching the optimal noise distributions. Apart from constructing attacks, the authors in [21] discussed privacy leakage and the ability of an adversary to fine-tune its model for malicious use. Finally, the authors propose a method to increase the generalisation error in the adversary's model by modifying the transmitted cut layer distribution.

Another direction in the adversarial scenarios is the reconstruction of private data through label leakage or gradient leakage. The authors in [23] proposed a deep gradient leakage mechanism, to reconstruct the private ground truth data from the leaked client's gradient. In [38], a solution to enable privacy-preserving in reconstructing private data based in [29], where the authors add distance correlation to the loss function to reduce the amount of information in the activation functions.

IV. PROBLEM FORMULATION

In this section, we describe the importance of developing tools and mechanisms to emulate the adversaries' behaviour in the context of distributed learning, including SL and FL. Both of them may contain encryption and privacy-preserving mechanisms. However, these mechanisms do not consider the adversarial scenario where the attacker has the privilege to influence the training process of SL and FL. Therefore, we aim to propose attacks in order to sufficiently test the susceptibility of the FL and SL against adversaries. First, we emulate the attacks in FL by attacking the global model using layer optimisation based on BO (BO-FLPA). Secondly, we develop the attacks in SL by attacking the server-side model through the parameterisation of the neural network layers on the client side in order to mislead the neural network during training to converge to the wrong features on the server side using BO (BO-SLPA).

A. BO-FLPA: BAYESIAN OPTIMISATION - FEDERATED LEARNING POISONING ATTACK

1) SYSTEM MODEL

In this scenario, we consider deploying FL at the network edge to learn a global model using the dataset generated by different Metaverse users. The considered FL framework consists of an edge server and a number of users, where the edge server aggregates the updated model parameters from the users. We assume that the server can perform mode aggregation with two methods, FedAvg [24] and Krum [26].

2) THREAT MODEL

In the aforementioned sections, literature have identified the threat of poisoning the global model with the manipulation of local models. In this scenario, we assume that there are some attackers which are aiming to poison the aggregation through the manipulation of the local model in such a way that the server will not detect any anomalies from the attackers' model during training and the attackers will remain in the training process. We consider poisoning the FL server with a layer optimisation method. The BO-FLPA searches for the optimal parameters to modify the layer using BO and deploys layer optimisation in the first and second convolutional layers of the convolutional neural network (CNN). The optimal parameters aim to minimise the KL Divergence between the actual output probability distribution and the target probability distribution. KL divergence

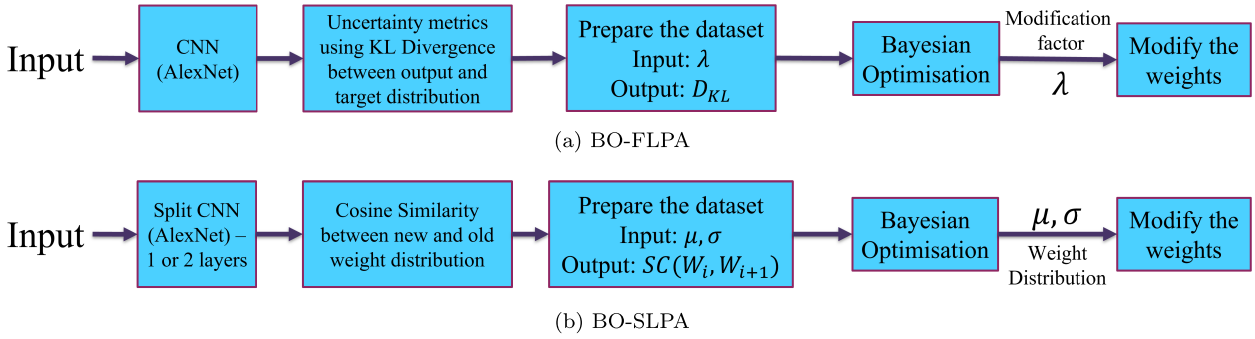


FIGURE 3. The poisoning attacks applied on CNN.

provides a metric for the uncertainty in the classification task of CNN.

3) ATTACK OPTIMISATION OBJECTIVE

Thus, based on the threat model, we define our optimisation problem as the minimisation of the statistical distance of the confidence levels between the output and the target, with the minimum absolute distance between the current weights W_j and new weights W_{j+1} as follows:

$$\begin{aligned} &\min_{\delta_w} \; D_{KL}(g(x'), target) \\ &\delta_w \leftarrow ||W_{j+1} - W_j||_1 \end{aligned} \tag{2}$$

where D_{KL} is the KL Divergence, $g(x')$ is the output of the global model and the target is the target probability distribution. The target probability distribution is a probability distribution with the highest probability at the actual class with values of 0.25, 0.50 and 0.75. The probabilities of the rest of the classes are uniformly distributed.

B. BO-SLPA: BAYESIAN OPTIMISATION - SPLIT LEARNING POISONING ATTACK

1) SYSTEM MODEL

In this scenario, we deploy SL at the network edge to learn a global model with the dataset generated by different Metaverse users. The considered SL framework consists of an edge server and a number of users, where the edge server retrieves the output data of the user's sub-model and performs the network optimisation in both the edge server and the user. We assume that there is no scheduling in the network and the learning process is performed sequentially.

2) THREAT MODEL

Similar to FL, there are users which wish to train their model along with a global model. The potential threat for the SL is to provide the global model, poisoned smashed data which will mislead the neural network optimisation to the wrong features. To perform this, we propose BO-SLPA which will perform layer optimisation on the first layer of the client's neural network similar to FL. Unfortunately, the amount of information on the client side is limited to quantifying the uncertainty induced in the prediction. Therefore, instead of identifying the uncertainty in the prediction, we use the gradient of the smashed data and the similarity of weights over time to determine the most significant features of the server's model. By changing those features, we aim to cause misclassifications to the server model.

3) ATTACK OPTIMISATION OBJECTIVE

The objective of the layer optimisation is to minimise the cosine similarity of the new weight distribution and the current weight distribution to mislead the server model, as follows,

$$\min_{\cos(\theta)\in[0.5,1]} SC(W_t, W_{t+1}) \tag{3}$$

where $SC(W_t, W_{t+1})$ is the cosine similarity between the new and current weights respectively (explained in section VI-A). We set the constraint of similarity to be 50% similar to the previous model or higher. The intuition of minimising the cosine similarity is that with the cosine similarity we can identify the important features during training and poison them.

V. METHODOLOGY FOR BO-FLPA

We outline our proposed layer optimisation in the intermediate layers in order to change the decision boundary of the DL model. The attack is deployed in the convolutional layer, We parameterise the first layer to increase the classification uncertainty. Since the model aggregation step in FL can aggregate the local model updates from many users, it may increase the susceptibility of the model against the adversarial samples.

Our proposed algorithm has two main mechanisms: (i) Observe the output of the DL model and modify the weights of the feature maps (W_t) of the first or second convolutional layer of AlexNet, (ii) use BO techniques to obtain the best set of modification factor Λ for each neuron. The detailed steps of our proposed layer optimization for model poisoning are presented in the **Algorithm 1**. Next, we introduce the BO techniques for our problem (Section V-A), followed by a method for modifying the weights (Section V-B)

Algorithm 1 BO-FLPA, There Are K Number of Clients, B Is the Local Minibatch Size, E Is the Number of Local Epochs and η Is the Learning Rate

Server Executes
  Initiliase weights w_0
  foreach *round* $t \in T$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ random set of m clients
    foreach *client* $k \in S_t$ **do**
      $w_{t+1}^k \leftarrow$ **Client Update** (k, w)
    end
    $w_{t+1} \leftarrow$ **Aggregation**(w_{t+1})
  end
Client Update *(k, w)*
  if *malicious* **then**
    Initialise $\mathcal{D} \leftarrow \Lambda_0, D_{KL,0}$
    Find GP based on 6
    while *not* $D_{KL} \leftarrow 0$ **do**
      $\nabla_x W \leftarrow l(w, b)$
      Find the most significant features
      Calculate Λ_t based on 12
      Calculate D_{KL}
      Update the posterior based on 6
      Sample a new Λ_{t+1} based on 11
      Modify the W
    end
  else
    foreach *epoch* $e \in E$ **do**
      foreach *batch* $b \in B$ **do**
        $w \leftarrow w - \eta \nabla l(w, b)$
      end
    end
  end
  return w to server

A. BAYESIAN OPTIMISATION

BO is a popular technique used in applied ML for tuning the hyper-parameters of a pre-trained model for a given dataset [39]. BO uses the Bayes Theorem to guide the search for the minimum or maximum of a given objective function in the context of a black-box optimisation [40]. BO has two important aspects which must be selected appropriately with respect to the objective function. The first aspect is the surrogate model, i.e., a function that describes the hypothesis of the black-box function which we aim to optimise. That is, on every iteration, the surrogate model computes the posterior distribution based on the previous observations and uses a prior model over the function and the likelihood [41]. The second aspect of BO is the acquisition function which relates the belief of the objective function with the input space. Then, the acquisition function aims to find a new sampling point that maximises the objective function.

1) SURROGATE MODEL

We choose to employ the Gaussian process (GP) regression for the surrogate model due to its ability to provide a Bayesian posterior distribution of the objective function for different input values [42]. Suppose that the black-box objective function is given as

$$\Lambda \in \arg\min_{\delta_w} D_{KL}(g(x'), target) \tag{4}$$

$$\delta_w \leftarrow ||W_{j+1}^i - W_j^i||_1$$
$$\lambda \in \Lambda, \tag{5}$$

where $g(x')$ is the output of the global model and the target is the target probability distribution, and W_j^i and W_{j+1}^i are the current and new weights respectively. The objective is to minimise the KL Divergence between the target output and the predicted output $g(x')$ via applying the modification factor on the weights.

The objective function of the optimization in (4) does not have an analytical form, therefore it is treated as a black-box optimisation problem [43]. In particular, we consider a black-box function $f : \Lambda \rightarrow \mathbb{R}$ where Λ is the modification factor set which is the input space and consists of N samples $\mathcal{D} = \Lambda_1, \ldots \Lambda_N$. The black-box function $f(\cdot)$ in BO is specified by the single-task Gaussian process (GP) regression function given by

$$f \sim GP(\mu(\mathcal{D}), k(\mathcal{D}, \mathcal{D}')), \tag{6}$$

where $\mu(\Lambda_{1:N})$ is the mean function that encodes the uncertainty of the expected output of the D_{KL}. The $K(\Lambda, \Lambda')$ is the covariance matrix that encodes the similarity between the uncertainties and the set of Λ. The $K(\mathcal{D}, \mathcal{D}')$ is denoted as

$$K(\mathcal{D}, \mathcal{D}') = \begin{bmatrix} \Lambda_1, \Lambda_1 & \cdots & \Lambda_1, \Lambda_N \\ \vdots & \ddots & \vdots \\ \Lambda_N, \Lambda_1 & \cdots & \Lambda_N, \Lambda_N \end{bmatrix} \tag{7}$$

The $K(\Lambda, \Lambda')$ follows the Matern covariance kernel which specifies the covariance between two random variables as follows

$$K(x_i, x_j) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{d(x_i, x_j)}{\rho} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{d(x_i, x_j)}{\rho} \right) \tag{8}$$

where $d(x_i, x_j)$ is the Euclidean Distance, K_ν is a modified Bessel function and $\Gamma(\cdot)$ is the gamma function. The parameter ν controls the smoothness of the resulting function and is generally set as $\nu = 2.5$, the parameter ρ is a positive parameter that is set to $\rho = 1$.

2) ACQUISITION FUNCTION

The acquisition function is used to sample a new sampling point in the search space. During every iteration, BO evaluates a new set of sampling points using the acquisition function. This balances the exploration and exploitation to

maximise the acquisition function for the next sampling point. Given a new observation Λ in each iteration by maximising the acquisition function $q(\cdot)$, we can find the new evaluation point λ.

There are multiple acquisition functions, such as Expected Improvement (EI), Probability Improvement, and Upper Confidence Bound (UCB) [41]. The performance of these functions depends on the derivations of the surrogate model and the dataset. For our problem, we use the Monte-Carlo based (-quasi) expected improvement (qEI) [44]. The EI aims to maximise the expectation of improvement over the current best function which describes the dataset,

$$EI(\Lambda|\mathcal{D}) = \mathbb{E}[\max(f(\Lambda) - f^*, 0)] \tag{9}$$

where f^* is the observed best value in $\mathcal{D}$ [39]. However, evaluating an integral over the posterior distribution is computationally expensive, so we use the Monte-Carlo (MC) sampling to approximate the integral [44], [45]. Hence, we can write the expectation of a real-valued function of the model output at the design point as

$$f(\Lambda) = \mathbb{E}[f(\Lambda)|\xi \sim \mathbb{P}(f(\Lambda)|\mathcal{D})] \approx \frac{1}{N}\sum_{i=1}^{N} f(\xi_i) \tag{10}$$

where ξ is the posterior distribution of the function f at Λ with the observed data $\mathcal{D}$ so far, and N represents N MC samples. Given the above Eq.(10), we can define the qEI as an approximation of EI. Therefore, the expression of the qEI is defined as

$$q_{EI}(\Lambda) \approx \frac{1}{N}\sum_{i=1}^{N} max_{j=1,\ldots,q}\{max(\xi_{ij} - f^*, 0)\},$$
$$\xi \sim \mathbb{P}(f(\Lambda)|\mathcal{D}) \tag{11}$$

where ξ is the posterior distribution of the function f at Λ with the observed data $\mathcal{D}$ so far, and f^* is the observed best function value [44].

Consequently, the output of the BO can give us the best parameters to modify the weights of a convolutional layer to reach the target confidence, and thus increase the uncertainty in the global model during aggregation.

B. MODIFYING THE WEIGHTS

The model weights can be modified by the given update rule in (12). The update rules are applied to every neuron of the selected layer.

$$W_{j+1}^i = W_j^i - \lambda \cdot \nabla(W_j^i)\,, \quad \lambda \in [0, 10] \tag{12}$$

where W_j^i is the current weight of feature map i and layer j, ∇W_j^i is the gradient for the layer j from feature map i and λ is the modification parameter. In order to find the best value of λ, we use the BO technique to find a black-box function that describes the relationship between λ and the KL divergences score.

VI. METHODOLOGY FOR BO-SLPA

In this section, we apply layer optimisation in the intermediate layers for modifying the weight parameters for misleading the optimisation of the server model in SL. We parameterise the first layer with a weight distribution derived from BO and then provide the server with false output data from the cut layer.

Our proposed algorithm includes the following four steps:

1) Find the best features based on gradients and similarity.
2) Calculate the new weight distribution for the feature.
3) Use the BO to obtain the mean and deviation of the weights for the best features.
4) Modify the weights accordingly to produce an erroneous input for the server's model.

A more detailed description is given in **Algorithm 2**. In the following subsections, we outline the cosine similarity for solving our formulated optimisation problem (Section VI-A), BO and the developed algorithm (Section VI-B), followed by a method for modifying the weights (Section VI-C).

Algorithm 2 Layer Optimisation for Model Poisoning in Split Learning

Data: $f_W(\mu, \sigma)$, engangement-criteria
Result: $g(x)$
if *epoch* $<$ *engage* **then**
  Apply split learning process
  Find the most significant features
  Calculate the similarity before and after the network optimisation
else
  $x \leftarrow \mu_W, \sigma_W, y \leftarrow SC(W_i, W_{i+1})$
  Apply BO to sample a new μ_W, σ_W for the features
  Apply algorithm 3 to modify the weights
  Interact with the server
  Find the most significant features
  Calculate the similarity before and after the network optimisation
end

A. COSINE SIMILARITY

We use the following cosine similarity to calculate the similarity between two weights, which provides a value in $(-1,1)$ and indicates a change of weights during the optimisation process of the neural network.

$$SC(W_{i+1}, W_i) = \cos\theta = \frac{W_{i+1} \cdot W_i}{\|W_{i+1}\|\|W_i\|} \tag{13}$$

where W_i and W_{i+1} are weight matrices before and after optimisation and θ is the angle between W_{i+1} and W_i.

B. BAYESIAN OPTIMIZATION

For poisoning the SL, for the black-box function, we consider that there is a function that describes the relationship of the

weight distributions for the most significant features with their similarity before and after optimisations. BO aims to search for weight distributions to minimise similarity subject to the constraint that similarity is greater than 0.5 which ensures that the weights are 50% similar. Therefore, the objective function is defined as follows:

$$\mu_{i+1},\ \sigma_{i+1} \leftarrow \min_{\cos(\theta)\in[0.5,1]} SC(W_i, W_{i+1}) \tag{14}$$

where μ_{i+1} and σ_{i+1} are forming the new weight distribution for the weights, $SC(W_i, W_{i+1})$ is the cosine similarity between the weight distributions before and after optimisation. We use the same components as in the poisoning for FL. That is, we use a single-task Gaussian process to model the black-box function $f(\cdot)$, and quasi-Expected Improvement (qEI) for the acquisition function.

C. MODIFYING THE WEIGHTS BASED ON THE WEIGHT DISTRIBUTION

The modification of the weights is different from the FL due to the fact that in this adversarial scenario, we modify the weights according to the target weight distribution from the BO. Also, we need to search for weights based on the mean μ and variance σ of the feature map. Therefore, we use gradient descent to search for weights with specific mean and variance.

Algorithm 3 Modify Weights for a Given Weight Distribution (μ, σ)

Data: $\mu, \sigma, f(\cdot)$
Result: W
$\alpha \leftarrow 0.5$
$\beta \leftarrow 0.005$
$\eta \leftarrow 0.005$
while $\|\hat{\mu} - \mu\| \geq \epsilon$ **do**
  $\alpha \leftarrow \alpha - \eta \cdot \frac{\partial L}{\partial \mu}$
  $\beta \leftarrow \beta - \eta \cdot \frac{\partial L}{\partial \sigma}$
  $W_{i+1} \leftarrow W_i + \alpha \cdot (M + \beta \cdot \nabla_x W_i)$
end

We modify the weights as described in **Algorithm 3**. We use the mean square error as a cost function to optimise the weights. We have defined α and β as the adjustable parameters that control mean (μ) and deviation (σ) respectively, and η is denoted as the learning rate for gradient descent. The update rules for α and β are given by (15)

$$\begin{aligned} L &= \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2 \\ \alpha &= \alpha - \eta * \frac{\partial L}{\partial \mu} \\ \beta &= \beta - \eta * \frac{\partial L}{\partial \sigma} \end{aligned} \tag{15}$$

where L is the mean squared error between our target mean and deviation $y \leftarrow (\mu, \sigma)$ and the current mean and deviation $\hat{y} \leftarrow (\hat{\mu}, \hat{\sigma})$ of the weight distribution.

VII. SIMULATION RESULTS

For the simulation of FL and SL and the corresponding adversarial attacks, we use two classical image recognition datasets, the handwritten digit MNIST [46] and CIFAR10 [47]. The MNIST dataset contains 10 classes of handwriting digits and consists of 60,000 images for training and 20,000 images for testing purposes. The images are grayscale with an image size of 28 by 28 pixels. The CIFAR10 dataset contains 10 classes of 50,000 images for training and 10,000 images for testing purposes. The images are RGB with each image size of 32 by 32 pixels. We distribute the dataset over each user following the distribution of independent and identically distributed (i.i.d.) and non-i.i.d. data.

The DL model used in our experiments is a CNN with the architecture of AlexNet [47]. We optimise the ML model using cross-entropy and stochastic gradient decent (SGD). The FL is set up with 8 users collaborating with the FL server to train the global model. Two types of model aggregation methods are simulated in FL, FedAvg, and Krum for adversarial immunity. The number of adversaries is varied between 25% and 50% of the total number of users and the rest are benign users. We test the attack algorithms at the lowest confidence interval of 0.25. We choose a low confidence interval in order to observe the effects of the adversarial samples produced by the Project Gradient Descent (PGD) attack. Finally, we assess the effectiveness of the proposed layer optimisation in different sections of the neural network. Specifically, we conduct layer optimisation in the first and second convolutional layer.

The SL framework is set up with the same settings as that of the FL framework. The client's and server's sub-models are trained following sequential rules, in which the next user starts training only when the current user completes one epoch with the server. Also, we assess the effectiveness of the layer optimisation in the first convolutional layer, and then we vary the number of layers on the client side between one and two layers.

A. PERFORMANCE OF BO-FLPA

We begin evaluating the performance of attacking FL with layer optimisation based on BO after the first training round. We consider the effects of layer optimisation on confidence levels through the optimisation of the most significant features. In Fig. 4 and 5, we plot the classification confidence intervals. In Fig. 4, it is apparent that a more complex dataset such as CIFAR10 than MNIST is a catalyst for the success of layer optimisation. Beginning with the FedAvg and MNIST dataset (Fig. 4a), we see that the confidence intervals are dropped when the number of adversaries is increased in the network. In addition, it shows that applying the layer optimisation in the second layer reduces the confidence intervals

TABLE 1. Experimental setup.

Federated & Split Learning Parameters	
Number of Users	8
Model Aggregation	FedAvg [24] and Krum [26]
Global & Local Model	AlexNet [47]
Loss Function	Cross Entropy
Optimiser	Stochastic Gradient Descent
DL model: AlexNet	
No. of Parameters	60 million
Layers	8
Kernel Size	(3 × 3), (3 × 3), (3 × 3), (3 × 3), (3 × 3)
Dataset	
Dataset	MNIST [46], CIFAR10 [47]
Experiments	
Adversaries	varying from 12.5% to 50%
Normal Users	varying from 50% to 87.5%
Adversarial Validation	Projected Gradient Descent attack
Attack: BO-FLPA	
Target Confidence levels	0.25, 0.5, 0.75
Attack: BO-SLPA	
alpha (α)	0.5
beta (β)	0.5
eta (η)	0.005

but the accuracy does not change significantly as shown in Table 2. However, comparing the results using Krum aggregation in Fig. 4c, we can see that the layer optimisation does not influence the aggregation, therefore it has only a very small effect on the global model and only a slight change in the error rate.

Moving to the CIFAR10 dataset, both aggregation methods fail to obtain a global model with a high classification accuracy and low uncertainty. In both FedAvg and Krum Fig. 4b and Fig. 4d), due to the fact that the DL model has a dataset with higher complexity than before, a small change in the model parameters in the early stages of training can have a big impact on the whole training. This results in low accuracy and high uncertainty. The uncertainty is mapped by the high volume of the probability density function.

With FedAvg, when the number of adversaries is increasing, the global model becomes less confident for the classification. Another phenomenon observed is that Krum is unable to distinguish or detect the adversaries in any of the eight cases presented in Fig. 4d and Fig.5d). This is because training CNN with CIFAR10 dataset is more complex, which has very similar weight distributions over all the users with a very small L_2 distance between each user's weights. In Fig. 5, we plot the confidence levels of the global model in the context of non-i.d.d. data. We can see that the global model with FedAvg is more confident in the classification and has higher accuracy than Krum. However, in both FedAvg and Krum, the phenomenon of client drift is visible where the global optimum weight is not equidistant from each client's weight.

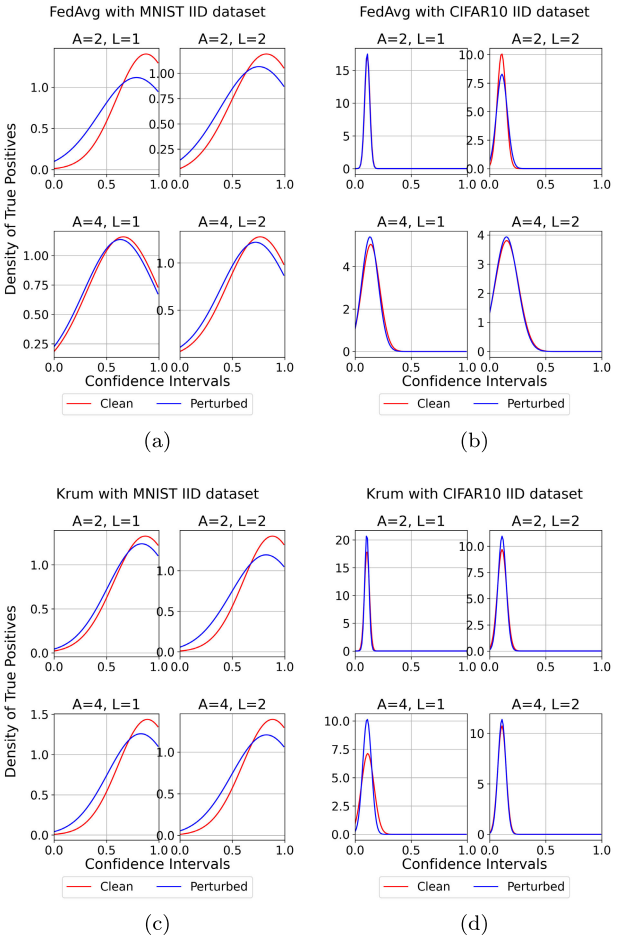


FIGURE 4. The confidence levels of the global model trained with CIFAR10 and MNIST on i.i.d scenario. The BO-FLPA is applied in a varying number of adversaries (A) at layer (L). In MNIST, we can see that there is a large deviation showing the large uncertainty induced to model. In CIFAR10 we can see that the model has very low confidence in the classification.

B. PERFORMANCE OF BO-SLPA

We evaluate the performance of attacking SL with BO-SLPA after 1 epoch. We consider the effects of poisoning the client model and then transmitting the poisoned smashed data to the server through the optimisation of the first layer on the user side. Then, we observe the classification confidence levels of the server's model when we feed the smashed data of a benign client model.

In Fig. 6a, we can see that the mean and the deviation of the density function of the confidence intervals for MNIST dataset are high meaning that the uncertainty is high. In addition, when applying adversarial attacks, the uncertainty is increasing because the mean decreases and the deviation increases. Fig. 6b, shows that there is very high uncertainty in CIFAR10 classification. This is illustrated by two facts. The existence of a high deviation in the classification levels of clean samples. Also, when feeding the perturbed samples in the models we obtain high density and small deviation in the low classification levels which can be considered as misclassifications.

TABLE 2. Accuracy for BO-FLPA with the number of adversaries (A) and the position of the layer optimisation (L) varying.

Aggregation	A	L	MNIST				CIFAR10			
			Normal		Adversarial		Normal		Adversarial	
			IID	non-IID	IID	non-IID	IID	non-IID	IID	non-IID
FedAvg	2	1	0.93	0.44	0.92	0.11	0.18	0.24	0.15	0.13
		2	0.95	0.28	0.90	0.12	0.22	0.10	0.25	0.10
	4	1	0.80	0.31	0.83	0.10	0.34	0.31	0.18	0.11
		2	0.92	0.28	0.90	0.09	0.34	0.29	0.35	0.09
Krum	2	1	0.95	0.21	0.87	0.10	0.15	0.14	0.18	0.12
		2	0.95	0.18	0.90	0.11	0.22	0.13	0.25	0.09
	4	1	0.95	0.16	0.92	0.09	0.21	0.15	0.18	0.11
		2	0.94	0.12	0.90	0.10	0.19	0.10	0.22	0.10

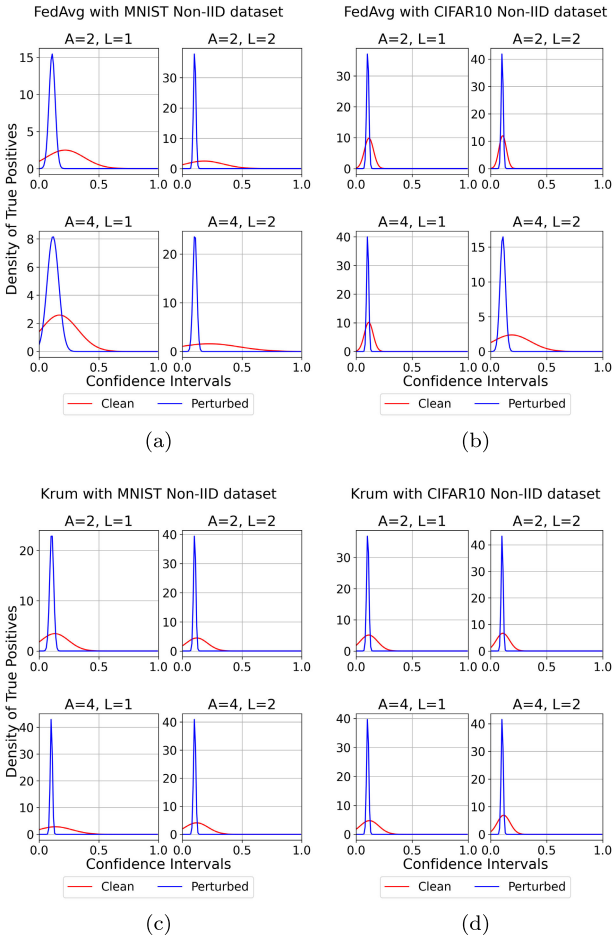


FIGURE 5. The confidence levels of the global model trained with CIFAR10 and MNIST on non-i.i.d scenario. The BO-FLPA is applied in a varying number of adversaries (A) at layer (L). It is apparent that BO-FLPA is effective in both of the dataset and we can see that the model has very low confidence in the classification.

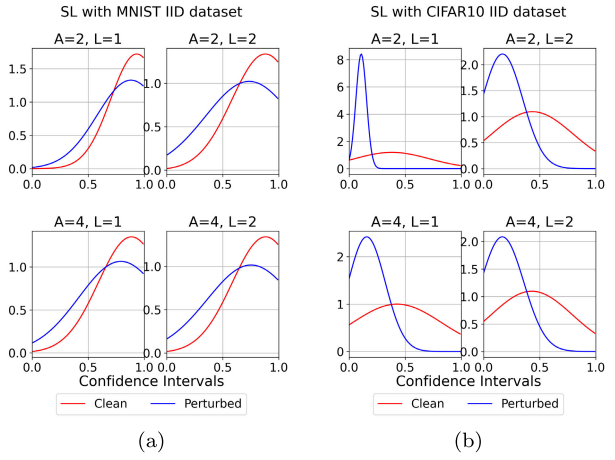


FIGURE 6. The confidence levels of a normal client trained with two i.i.d. datasets CIFAR10 and MNIST in SL. We apply BO-SLPA, and we are varying the number of adversaries (A) and the number of layers used in the client side (L).

TABLE 3. Accuracy for BO-SLPA with the number of adversaries (A) and the position of the layer optimisation (L) varying.

A	L	MNIST		CIFAR10	
		Normal	Adversarial	Normal	Adversarial
2	1	0.956	0.902	0.535	0.115
	2	0.917	0.809	0.574	0.234
4	1	0.909	0.838	0.578	0.218
	2	0.915	0.771	0.588	0.201

Another observation of simulating the attacks in SL is that when the number of neural layers on the client side is increased, the confidence levels and the deviation with MNIST dataset are shifted (Fig. 6a). This is because we optimise the network to generate poisoned smashed data to transmit to the server. In addition, by adding an extra layer in front of the malicious layer, the poisonous data is propagated further into more neurons rather than only in the most significant neurons. We can also see this for CIFAR10 where there is more complexity in the dataset (Fig. 6b). The same trend is observed when the number of adversaries is increased. Even though the same trend is observed, it affects the server-side model from a different perspective. Instead of expanding the dimensionality of the poisoned smashed data, we increase the iterations with the server model and the poisoned data. This affects the optimisation on both the server and client sides.

VIII. CONCLUSION

We proposed two adversarial poisoning attack methods to emulate the behaviours of adversaries in federated learning (FL) and split learning (SL) to provide the necessary foundation for constructing privacy protection in

Metaverse scenarios. In our proposed poisoning attack methods, we specify there is a mapping function that describes the prediction's uncertainty and the layer optimisation parameters. We optimised it by employing Bayesian Optimisation, which aims to search for the optimal parameters for optimising the hidden layers for poisoning FL and SL mechanisms. We demonstrated the effects of poisoning attacks on the prediction confidence levels of the neural networks. Using MNIST and CIFAR10 datasets, we confirmed that the poisoning attack may destroy the global model and cause problems with the accuracy of the prediction.

The work produced in this paper is aimed at only the proof of concept of Bayesian optimisations for adversarial attacks in FL and SL, and our future work will consider practical demonstration of Metaverse scenarios including eye tracking and social networks.

REFERENCES

[1] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 319–352, 1st Quart., 2023.

[2] H. Ning, H. Wang, Y. Lin, W. Wang, S. Dhelim, F. Farha, J. Ding, and M. Daneshmand, "A survey on metaverse: The state-of-the-art, technologies, applications, and challenges," 2021, *arXiv:2111.09673*.

[3] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 656–700, 1st Quart., 2023.

[4] P. Kairouz et al., "Advances and open problems in federated learning," Dec. 2019, *arXiv:1912.04977*.

[5] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving Google keyboard query suggestions," Dec. 2018, *arXiv:1812.02903*.

[6] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," Oct. 2018, *arXiv:1810.06060*.

[7] M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, "Split learning for collaborative deep learning in healthcare," Dec. 2019, *arXiv:1912.12115*.

[8] X. Liu, Y. Deng, and T. Mahmoodi, "Wireless distributed learning: A new hybrid split and federated learning approach," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2650–2665, Apr. 2023.

[9] Z. Yang, M. Chen, K.-K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6G: Applications, challenges, and opportunities," *Engineering*, vol. 8, pp. 33–41, Jan. 2022.

[10] P. Bhattacharya, A. Verma, V. K. Prasad, S. Tanwar, B. Bhushan, B. C. Florea, D. D. Taralunga, F. Alqahtani, and A. Tolba, "Game-o-meta: Trusted federated learning scheme for P2P gaming metaverse beyond 5G networks," *Sensors*, vol. 23, no. 9, p. 4201, Apr. 2023. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85159171306&doi=10.3390%2fs23094201&partnerID=40&md5=b85ce5c7b08434cf483214f267bc94ef

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[13] M. Aristodemou, S. Lambotharan, G. Zheng, and L. Aristodemou, "Investigation of deep learning architectures and features for adversarial machine learning attacks in modulation classifications," in *Proc. IEEE 14th Image, Video, Multidimensional Signal Process. Workshop (IVMSP)*, Jun. 2022, pp. 1–6.

[14] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.

[15] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," Nov. 2018, *arXiv:1811.12470*.

[16] L. Shi, Z. Chen, Y. Shi, G. Zhao, L. Wei, Y. Tao, and Y. Gao, "Data poisoning attacks on federated learning by using adversarial samples," in *Proc. Int. Conf. Comput. Eng. Artif. Intell. (ICCEAI)*, Jul. 2022, pp. 158–162.

[17] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2020, pp. 480–501.

[18] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," Nov. 2019, *arXiv:1911.11815*.

[19] X. Zhou, M. Xu, Y. Wu, and N. Zheng, "Deep model poisoning attack on federated learning," *Future Internet*, vol. 13, no. 3, p. 73, Mar. 2021.

[20] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," Jul. 2018, *arXiv:1807.00459*.

[21] F. Zheng, C. Chen, B. Yao, and X. Zheng, "Making split learning resilient to label leakage by potential energy loss," Oct. 2022, *arXiv:2210.09617*.

[22] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," Feb. 2021, *arXiv:2102.08504*.

[23] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," Jun. 2019, *arXiv:1906.08935*.

[24] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," Oct. 2017, *arXiv:1710.06963*.

[25] A. Kundu, P. Yu, L. Wynter, and S. H. Lim, "Robustness and personalization in federated learning: A unified approach via regularization," Sep. 2020, *arXiv:2009.06303*.

[26] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 118–128.

[27] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," Sep. 2019, *arXiv:1909.09145*.

[28] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," Dec. 2018, *arXiv:1812.03288*.

[29] P. Vepakomma, A. Singh, O. Gupta, and R. Raskar, "NoPeek: Information leakage reduction to share activations in distributed deep learning," Aug. 2020, *arXiv:2008.09161*.

[30] I. Y. Tyukin, D. J. Higham, and A. N. Gorban, "On adversarial examples and stealth attacks in artificial intelligence systems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–6.

[31] N. M. Gottschling, V. Antun, A. C. Hansen, and B. Adcock, "The troublesome kernel—On hallucinations, no free lunches and the accuracy-stability trade-off in inverse problems," Jan. 2020, *arXiv:2001.01258*.

[32] B. Biggio et al., "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Computer Science), vol. 8190, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-40994-3_25.

[33] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, vol. 2, 2012, pp. 1807–1814.

[34] A. Kurakin et al., "Adversarial attacks and defences competition," Mar. 2018, *arXiv:1804.00097*.

[35] S. Lu, R. Li, W. Liu, and X. Chen, "Defense against backdoor attack in federated learning," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102819.

[36] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 374–380.

[37] J. Liu and X. Lyu, "Clustering label inference attack against practical split learning," Mar. 2022, *arXiv:2203.05222*.

[38] D. Xiao, C. Yang, and W. Wu, "Mixing activations and labels in distributed training for split learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 3165–3177, Nov. 2022.

[39] P. I. Frazier, "A tutorial on Bayesian optimization," Jul. 2018, *arXiv:1807.02811*.

[40] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," Jun. 2012, *arXiv:1206.2944*.

[41] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams, "Scalable Bayesian optimization using deep neural networks," Feb. 2015, *arXiv:1502.05700*.

[42] M. P. Deisenroth and J. W. Ng, "Distributed Gaussian processes," Feb. 2015, *arXiv:1502.02843*.

[43] Y. Zhang, J. Jordon, A. M. Alaa, and M. van der Schaar, "Lifelong Bayesian optimization," May 2019, *arXiv:1905.12280*.

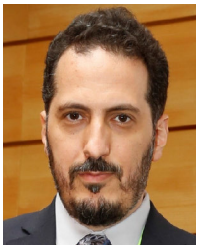
[44] J. T. Wilson, R. Moriconi, F. Hutter, and M. P. Deisenroth, "The reparameterization trick for acquisition functions," Dec. 2017, *arXiv:1712.00424*.

[45] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," Dec. 2013, *arXiv:1312.6114*.
[46] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

MARIOS ARISTODEMOU (Graduate Student Member, IEEE) received the M.Eng. degree in systems engineering from Loughborough University, U.K., in 2021, where he is currently pursuing the Ph.D. degree with the Wolfson School of Mechanical, Electrical and Manufacturing Engineering. His current research interests include artificial intelligence and metaverse, including security and trustworthiness, personalized distributed learning, privacy-preserving for distributed learning, and meta-learning for cyber-security.

XIAOLAN LIU (Member, IEEE) received the Ph.D. degree from the Queen Mary University of London, U.K., in 2021. She was a Research Associate with King's College London, from 2020 to 2021. Since October 2021, she has been with the Institute for Digital Technologies, Loughborough University London, U.K., where she is currently a Lecturer. Her current research interests include wireless distributed learning, multi-agent reinforcement learning for edge computing, and machine learning for wireless communication optimization.

SANGARAPILLAI LAMBOTHARAN (Senior Member, IEEE) received the Ph.D. degree in signal processing from Imperial College London, U.K., in 1997. He was a Visiting Scientist with the Engineering and Theory Centre, Cornell University, USA, in 1996. Until 1999, he was a Postdoctoral Research Associate with Imperial College London. From 1999 to 2002, he was with the Motorola Applied Research Group, U.K., where he investigated various projects, such as physical link layer modeling and performance characterization of GPRS, EGPRS, and UTRAN. He was with King's College London and Cardiff University as a Lecturer and a Senior Lecturer, respectively, from 2002 to 2007. He is currently a Professor of signal processing and communications and the Director of the Institute for Digital Technologies, Loughborough University London, U.K. His current research interests include 5G networks, MIMO, blockchain, machine learning, and network security. He has authored more than 250 journal articles and conference papers in these areas. He is a fellow of IET. He serves as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and IEEE TRANSACTIONS ON COMMUNICATIONS.



BASIL AsSADHAN (Senior Member, IEEE) received the M.Sc. degree in electrical and computer engineering from the University of Wisconsin and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University. He is currently an Associate Professor with the Electrical Engineering Department, King Saud University. His current research interests include cybersecurity, network security, network traffic analysis, anomaly detection, and machine learning.