ADVERSARIALLY ROBUST FEDERATED LEARNING FOR NEURAL NETWORKS

ABSTRACT

In federated learning, data is distributed among local clients which collaboratively train a prediction model using secure aggregation. To preserve the privacy of the clients, the federated learning paradigm requires each client to maintain a private local training data set, and only uploads its summarized model updates to the server. In this work, we show that this paradigm could lead to a vulnerable model, which collapses in performance when the corrupted data samples (under adversarial manipulations) are used for prediction after model deployment. To improve model robustness, we first decompose the aggregation error of the central server into bias and variance, and then, propose a robust federated learning framework, named `Fed_BVA`, that performs on-device adversarial training using the bias-variance oriented adversarial examples supplied by the server via asymmetrical communications. The experiments are conducted on multiple benchmark data sets using several prevalent neural network models, and the empirical results show that our framework is robust against white-box and black-box adversarial corruptions under both IID and non-IID settings.

1 INTRODUCTION

The explosive amount of decentralized user data collected from the ever-growing usage of smart devices, e.g., smartphones, wearable devices, home sensors, etc., has led to a surge of interest in the field of decentralized learning. To protect the privacy-sensitive data of the clients, federated learning (McMahan et al., 2017; Yang et al., 2019) has been proposed. Federated learning only allows a group of clients to train local models using their own data, and then collectively merges the model updates on a central server using secure aggregation (Acar et al., 2018). Due to its high privacy-preserving property, federated learning has attracted much attention in recent years along with the prevalence of efficient light-weight deep models (Howard et al., 2017) and low-cost network communications (Wen et al., 2017; Konečnỳ et al., 2016).

In federated learning, the central server only inspects the secure aggregation of the local models as a whole. Consequently, it is susceptible to clients' corrupted updates (e.g., system failures, etc). Recently, multiple robust federated learning models (Fang et al., 2019; Pillutla et al., 2019; Portnoy & Hendler, 2020; Mostafa, 2019) have been proposed. These works only focus on performing client-level robust training or designing server-level aggregation variants with hyper-parameter tuning for Byzantine failures. However, none of them have the ability to mitigate the federated learning's vulnerability when the adversarial manipulations are present during testing, which as we shown in Section 4.1 that is mainly due to the generalization error in the model aggregation.

Our work bridges this gap by investigating the error incurred during the aggregation of federated learning from the perspective of bias-variance decomposition (Domingos, 2000; Valentini & Dietterich, 2004). Specifically, we show that the generalization error of the aggregated model on the central server can be decomposed as the combination of **bias** (triggered by the main prediction of these clients) and **variance** (triggered by the variations among clients' predictions). Next, we propose to perform the local robust training on clients by supplying them with a tiny amount of the bias-variance perturbed examples generated from the central server via asymmetrical communications. The experiments are conducted on neural networks with cross-entropy loss, however, other loss functions are also applicable as long as their gradients w.r.t. bias and variance are tractable to estimate. In this way, any gradient-based adversarial training strategies (Goodfellow et al., 2015; Madry et al., 2018) could be used. Compared with previous work, our major contributions include:

- We provide the exact solution of bias-variance analysis w.r.t. the generalization error which is perfectly suitable for neural network based federated learning. As a comparison, performing adversarial attacks or training with conventional federated learning methods will only focus on the bias of the central model but ignore the variance.

- We demonstrate that the conventional federated learning framework is vulnerable to the strong attacking methods with increasing communication rounds even if the adversarial training using the locally generated adversarial examples is performed on each client.
- Without violating the clients' privacy, we show that providing a tiny amount of bias-variance perturbed data from the central server to the clients through asymmetrical communication could dramatically improve the robustness of the training model under various settings.

2 PRELIMINARIES

2.1 SETTINGS

In federated learning, there is a central server and K different clients, each with access to a private training set $\mathcal{D}_k = \{(x_i^k, t_i^k)\}_{i=1}^{n_k}$, where x_i^k, t_i^k, and n_k are the features, label, and number of training examples in the k^{th} client $(k = 1, \cdots, K)$. Each data $\mathcal{D}_k$ is exclusively owned by client k and will not be shared with the central server or other clients. In addition, there is a small public training set $\mathcal{D}_s = \{(x_j^s, t_j^s)\}_{j=1}^{n_s}$ with n_s training examples from the server that is shared with clients, where $n_s \ll \sum_{k=1}^{K} n_k$. Note that this will not break the privacy constraints, for example, hospitals (local devices) that contribute to a federated learned medical image diagnosis system could take a few publicly accessible images as additional inputs. The goal of federated learning is to train a global classifier $f(\cdot)$ using knowledge from all the clients such that it generalizes well over test data $\mathcal{D}_{test}$. The notation used in this paper is summarized in the Appendix (see Table 4).

2.2 PROBLEM DEFINITION

In this paper, we study the adversarial robustness of neural networks[1] in federated learning setting, and we define robust decentralized learning as follows.

Definition 2.1. (Adversarially Robust Federated Learning)
Input: (1) A set of private training data $\{\mathcal{D}_k\}_{k=1}^{K}$ on K different clients; (2) Tiny amount of training data $\mathcal{D}_s$ on the central server; (3) Learning algorithm $f(\cdot)$ and loss function $L(\cdot, \cdot)$.
Output: A trained model on the central server that is robust against adversarial perturbation.

We would like to point out that our problem definition has the following properties: **Asymmetrical communication:** The asymmetrical communication between each client and server cloud is allowed: the server provides both global model parameters and limited shared data to the clients; while each client only uploads its local model parameters back to the server. **Data distribution:** All training examples on the clients and the server are assumed to follow the same data distribution. However, the experiments show that our proposed algorithm also achieves outstanding performance under the non-IID setting, which could be common among personalized clients in real scenarios. **Shared learning algorithm:** All the clients are assumed to use the identical model $f(\cdot)$, including architectures as well as hyper-parameters (e.g., learning rate, local epochs, local batch size).

Remark. *The basic assumption of this problem setting is that the learning process is clean (no malicious behaviors are observed during training), however, the intentionally generated adversarial poisoning data will be mixed with clean data during training. The eventual trained model being deployed on the devices will be robust against potential future adversarial attacks.*

2.3 BIAS-VARIANCE TRADE-OFF

Following (Domingos, 2000; Valentini & Dietterich, 2004), we define the optimal prediction, main prediction as well as the bias, variance, and noise for any real-valued loss function $L(\cdot, \cdot)$ as follows:

Definition 2.2. (**Optimal Prediction and Main Prediction**) Given loss function $L(\cdot, \cdot)$ and learning algorithm $f(\cdot)$, optimal prediction y_* and main prediction y_m for an example are defined as:

$$y_*(x) = \arg\min_{y} \mathbb{E}_t[L(y, t)] \quad \text{and} \quad y_m(x) = \arg\min_{y'} \mathbb{E}_{\mathcal{D}}[L(f_{\mathcal{D}}(x), y')] \tag{1}$$

where t and $\mathcal{D}$ are viewed as the random variables to denote the class label and training set, and $f_{\mathcal{D}}$ denotes the model trained on $\mathcal{D}$. In short, the main prediction is the prediction whose average loss relative to all the predictions over data distributions is minimum, e.g., the main prediction for zero-one loss is the mode of predictions. In this work, we show that the main prediction is the average prediction of client models for mean squared (MSE) loss and cross-entropy (CE) loss in Section 4.1.

[1] Our theoretical contribution mainly focuses on classification using neural networks with cross-entropy loss and mean squared loss. However, the proposed framework is generic to allow the use of other classification loss functions as well.

Definition 2.3. (**Bias, Variance and Noise**) Given a loss function $L(\cdot,\cdot)$ and a learning algorithm $f(\cdot)$, the expected loss $\mathbb{E}_{\mathcal{D},t}[L(f_{\mathcal{D}}(x),t)]$ for an example x can be decomposed[2] into bias, variance and noise as follows:

$$B(x) = L(y_m, y_*) \quad \text{and} \quad V(x) = \mathbb{E}_{\mathcal{D}}[L(f_{\mathcal{D}}(x), y_m)] \quad \text{and} \quad N(x) = \mathbb{E}_t[L(y_*, t)] \tag{2}$$

In short, bias is the loss incurred by the main prediction w.r.t. the optimal prediction, and variance is the average loss incurred by predictions w.r.t. the main prediction. Noise is conventionally assumed to be irreducible and independent to $f(\cdot)$.

Remark. *Our definitions on optimal prediction, main prediction, bias, variance and noise slightly differ from previous ones (Domingos, 2000; Valentini & Dietterich, 2004). For example, conventional optimal prediction was defined as* $y_*(x) = \arg\min_y \mathbb{E}_t[L(t,y)]$*, and it is equivalent to our definition when loss function is symmetric over its arguments, i.e.,* $L(y_1,y_2) = L(y_2,y_1)$*. Note that this decomposition holds for any real-valued loss function in the binary setting (Domingos, 2000) with a bias & variance trade-off coefficient that has a closed-form expression. For multi-class setting, we inherit their definition of bias & variance directly, and treat the trade-off coefficient as a hyper-parameter* λ *to tune because no closed-form expression of* λ *is available.*

3 The Proposed Framework

A typical framework (Kairouz et al., 2019) of privacy-preserving federated learning can be summarized as follows: (1) *Client Update:* Each client updates local model parameters w_k by minimizing the empirical loss over its own training set; (2) *Forward Communication:* Each client uploads its model parameter update to the central server; (3) *Server Update:* It synchronously aggregates the received parameters; (4) *Backward Communication:* The global parameters are sent back to the clients. Our framework follows the same paradigm but with substantial modifications as below.

Server Update. The server has two components: The first one uses FedAvg (McMahan et al., 2017) algorithm to aggregate the local models' parameters, i.e., $w_G = \text{Aggregate}(w_1, \cdots, w_K) = \sum_{k=1}^{K} \frac{n_k}{n} w_k$ where $n = \sum_{k=1}^{K} n_k$ and w_k is the model parameters in the k^{th} client. Meanwhile, another component is designed to produce adversarially perturbed examples which could be induced by a poisoning attack algorithm for the usage of robust adversarial training.

It has been well studied (Belkin et al., 2019; Domingos, 2000; Valentini & Dietterich, 2004) that in the classification setting, the generalization error of a learning algorithm on an example is determined by the bias, variance, and irreducible noise as defined in Eq. (2). Similar to the previous work, we also assume a noise-free learning scenario where the class label t is a deterministic function of x (i.e., if x is sampled repeatedly, the same values of its class t will be observed). This motivates us to generate the adversarial examples by attacking the bias and variance induced by clients' models as:

$$\max_{\hat{x} \in \Omega(x)} B(\hat{x}; w_1, \cdots, w_K) + \lambda V(\hat{x}; w_1, \cdots, w_K) \quad \forall (x,t) \in \mathcal{D}_s \tag{3}$$

where $B(\hat{x}; w_1, \cdots, w_K)$ and $V(\hat{x}; w_1, \cdots, w_K)$ could be empirically estimated from a finite number of clients' parameters trained on local training sets $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}$. Here λ is a hyper-parameter to measure the trade-off of bias and variance, and $\Omega(x)$ is the perturbation constraint.

Note that $\mathcal{D}_s$ (on the server) is the candidate subset of all available training examples that would lead to their perturbed counterparts. This is a more feasible setting as compared to generating adversarial examples on clients' devices because the server usually has much powerful computational capacity in real scenarios that allows the usage of flexible poisoning attack algorithms. In this case, both poisoned examples and server model parameters would be sent back to each client (*Backward Communication*), while only clients' local parameters would be uploaded to the server (*Forward Communication*), i.e., the *asymmetrical communication* as discussed in Section 2.2.

Client Update. The robust training of one client's prediction model (i.e., w_k) can be formulated as the following minimization problem.

$$\min_{w_k} \left(\sum_{i=1}^{n_k} L(f_{\mathcal{D}_k}(x_i^k; w_k), t_i^k) + \sum_{j=1}^{n_s} L(f_{\mathcal{D}_k}(\hat{x}_j^s; w_k), t_j^s) \right) \tag{4}$$

where $\hat{x}_j^s \in \Omega(x_j^s)$ is the perturbed examples that is asymmetrically transmitted from the server.

[2] This decomposition is based on the weighted sum of bias, variance, and noise. In general, t is a non-deterministic function (Domingos, 2000) of x when the irreducible noise is considered. Namely, if x is sampled repeatedly, different values of t will be observed.

Remark. *Intuitively, the bias measures the systematic loss of a learning algorithm, and the variance measures the prediction consistency of the learner over different training sets. Therefore, our robust federated learning framework has the following advantages: (i) it encourages the clients to consistently produce the optimal prediction for perturbed examples, thereby leading to a better generalization performance; (ii) local adversarial training on perturbed examples allows to learn a robust local model, and thus a robust global model could be aggregated from clients.*

Theoretically, we could still have another alternative robust federated training strategy:

$$\min_{w_k} \sum_{i=1}^{n_k} \max_{\hat{x}_i^k \in \Omega(x_i^k)} L(f(\hat{x}_i^k; w_k), t_i^k) \quad \forall k \in \{1, 2, \cdots, K\} \tag{5}$$

where the perturbed training examples of each client k is generated on local devices from $\mathcal{D}_k$ instead of transmitted from the server. This min-max formula is similar to (Madry et al., 2018; Tramèr et al., 2018) where the inner maximization problem synthesizes the adversarial counterparts of clean examples, while the outer minimization problem finds the optimal model parameters over perturbed training examples. Thus, each local robust model is trained individually, nevertheless, poisoning attacks on device will largely increase the computational cost and memory usage. Meanwhile, it only considers the client-specific loss and is still vulnerable against adversarial examples with increasing communication rounds. Both phenomena are observed in our experiments (see Fig. 4 and Fig. 5).

4 Algorithm

4.1 Bias-Variance Attack

We first consider the maximization problem in Eq. (3) using bias-variance based adversarial attacks. It aims to find the adversarial example $\hat{x}$ (from the original example x) that would produce large bias and variance values w.r.t. clients' local models. Specifically, perturbation constraint $\hat{x} \in \Omega(x)$ forces the adversarial example $\hat{x}$ to be visually indistinguishable w.r.t. x. Here we consider the well-studied l_∞-bounded adversaries[3] (Goodfellow et al., 2015; Madry et al., 2018; Tramèr et al., 2018) such that $\Omega(x) := \{\hat{x} \big| ||\hat{x} - x||_\infty \leq \epsilon\}$ for a perturbation magnitude ϵ. Furthermore, we propose to consider the following two gradient-based algorithms to generate adversarial examples.

Bias-variance based Fast Gradient Sign Method (BV-FGSM): Following FGSM (Goodfellow et al., 2015), it linearizes the maximization problem in Eq. (3) with one-step attack as follows.

$$\hat{x}_{\mathrm{BV-FGSM}} := x + \epsilon \cdot \operatorname{sign}\left(\nabla_x \left(B(x; w_1, \cdots, w_K) + \lambda V(x; w_1, \cdots, w_K)\right)\right) \tag{6}$$

Bias-variance based Projected Gradient Descent (BV-PGD): PGD can be considered as a multi-step variant of FGSM (Kurakin et al., 2017) and might generate powerful adversarial examples. This motivated us to derive a BV-based PGD attack:

$$\hat{x}^{l+1}_{\mathrm{BV-PGD}} := \mathrm{Proj}_{\Omega(x)}\left(\hat{x}^l + \epsilon \cdot \operatorname{sign}\left(\nabla_{\hat{x}^l}\left(B(\hat{x}^l; w_1, \cdots, w_K) + \lambda V(\hat{x}^l; w_1, \cdots, w_K)\right)\right)\right) \tag{7}$$

where $\hat{x}^l$ is the adversarial example at the l^{th} step with the initialization $\hat{x}^0 = x$ and $\mathrm{Proj}_{\Omega(x)}(\cdot)$ projects each step onto $\Omega(x)$.

Remark. *The proposed framework could be naturally generalized to any gradient-based adversarial attack algorithms where the gradients of bias $B(\cdot)$ and variance $V(\cdot)$ w.r.t. x are tractable when estimated from finite training sets. Compared with the existing attack methods (Carlini & Wagner, 2017; Goodfellow et al., 2015; Kurakin et al., 2017; Moosavi-Dezfooli et al., 2016), our loss function the adversary aims to optimize is a linear combination of bias and variance, whereas existing work mainly focused on attacking the overall classification error that considers bias only.*

The following theorem states that bias $B(\cdot)$ and variance $V(\cdot)$ as well as their gradients over input x could be estimated using the clients' models.

Theorem 4.1. Assume that $L(\cdot, \cdot)$ is the cross-entropy loss function, then, the empirical estimated main prediction y_m for an input example (x, t) has the following closed-form expression: $y_m(x; w_1, \cdots, w_K) = \frac{1}{K}\sum_{k=1}^K f_{\mathcal{D}_k}(x; w_k)$. Furthermore, the empirical bias and variance, as well as their gradients over an input x are estimated as follows:

$$B(x; w_1, \cdots, w_K) = \frac{1}{K}\sum_{k=1}^K L(f_{\mathcal{D}_k}(x; w_k), t); \qquad V(x; w_1, \cdots, w_K) = L(y_m, y_m) = H(y_m)$$

[3] l_∞ robustness is surely not the only option for robustness learning. However, we use this standard approach to show the limitations of prior federated learning, and evaluate the improvements of our proposed framework.

Here, $H(y_m) = -\sum_{j=1}^{C} y_m^{(j)} \log y_m^{(j)}$ is the entropy of the main prediction y_m and C is the number of classes. Easily, we can have their gradients in terms of the bias and variance as $\nabla_x B(x; w_1, \cdots, w_K) = \frac{1}{K} \sum_{k=1}^{K} \nabla_x L(f_{\mathcal{D}_k}(x; w_k), t)$ and $\nabla_x V(x; w_1, \cdots, w_K) = -\frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{C} (\log y_m^{(j)} + 1) \nabla_x f_{\mathcal{D}_k}^{(j)}(x; w_k)$. Details of the proof is elaborated in A.2.

In addition, we also consider the case where $L(\cdot, \cdot)$ is the MSE loss function. But the gradients of MSE's bias and variance are much more computational demanding comparing with the concise formulas that cross-entropy ends up with. More comparisons are illustrated in Appendix A.5.1.

Algorithm 1 Fed_BVA

1: **Input:** K (number of clients, with local data sets $\{\mathcal{D}_k\}_{k=1}^{K}$); f (learning model), E (number of local epochs); F (fraction of clients selected on each round); B (batch size of local client); η (learning rate); $\mathcal{D}_s$ (shared data set on server); ϵ (perturbation magnitude).
2: **Initialization:** Initialize w_G^0 and $\hat{\mathcal{D}}_s = \emptyset$
3: **for** each round $r = 1, 2, \cdots$ **do**
4: $m = \max(F \cdot K, 1)$
5: $S_r \leftarrow$ randomly sampled m clients
6: **for** each client $k \in S_r$ in parallel **do**
7: $w_k^r, f_{\mathcal{D}_k}, \nabla_x f_{\mathcal{D}_k} \leftarrow$ **ClientUpdate**$(w_G^{r-1}, \hat{\mathcal{D}}_s, \mathcal{D}_s, k)$
8: **end for**
9: $\hat{\mathcal{D}}_s \leftarrow$ **BVAttack**$(\{f_{\mathcal{D}_k}, \nabla_x f_{\mathcal{D}_k}\} | k \in S_r)$
10: $w_G^r \leftarrow$ **Aggregate**$(w_k^r | k \in S_r)$
11: **end for**
12: **return** w_G

Algorithm 2 ClientUpdate$(w, \hat{\mathcal{D}}_s, \mathcal{D}_s, k)$

1: Initialize k^{th} client's model with w
2: $\mathcal{B} \leftarrow$ split $\mathcal{D}_k \cup \hat{\mathcal{D}}_s$ into batches of size B
3: **for** each local epoch $i = 1, 2, \cdots, E$ **do**
4: **for** local batch $(x, t) \in \mathcal{B}$ **do**
5: $w \leftarrow w - \eta \nabla L(f_{\mathcal{D}_k}(x; w), t)$
6: **end for**
7: **end for**
8: Calculate $f_{\mathcal{D}_k}(x; w_k^r)$, $\nabla_x f_{\mathcal{D}_k}(x; w)$ $\forall x \in \mathcal{D}_s$
9: **return** w, $f_{\mathcal{D}_k}(x; w_k^r)$, $\nabla_x f_{\mathcal{D}_k}(x; w)$

Algorithm 3 BVAttack$(\{f_{\mathcal{D}_k}, \nabla_x f_{\mathcal{D}_k}\} | k \in S_r)$

1: Initialize $\hat{\mathcal{D}}_s = \emptyset$
2: **for** $(x, t) \in \mathcal{D}_s$ **do**
3: Estimate the gradients $\nabla_x B(x)$ and $\nabla_x V(x)$ using Theorem 4.1
4: Calculate $\hat{x}$ using Eq. (6) or (7) and add to $\hat{\mathcal{D}}_s$
5: **end for**
6: **return** $\hat{\mathcal{D}}_s$

4.2 FED_BVA

We present a novel robust federated learning algorithm with our proposed bias-variance attacks, named `Fed_BVA`. Following the framework defined in Eq. (3) and Eq. (4), key components of our algorithm are (1) bias-variance attacks for generating adversarial examples on the server, and (2) adversarial training using poisoned server examples together with clean local examples on each client. Therefore, we optimize these two objectives by producing the adversarial examples $\hat{\mathcal{D}}_s$ and updating the local model parameters w iteratively.

The proposed algorithm is summarized in Alg. 1. Given the server's $\mathcal{D}_s$ and clients' training data $\{\mathcal{D}_k\}_{k=1}^{K}$ as input, the output is a robust global model on the server. In this case, the clean server data $\mathcal{D}_s$ will be shared to all the clients. First, it initializes the server's model parameter w_G and perturbed data $\hat{\mathcal{D}}_s$, and then assigns to the randomly selected clients (Steps 4-5). Next, each client optimizes its own local model (Steps 6-8) with the received global parameters w_G as well as its own clean data $\mathcal{D}_k$, and uploads the updated parameters as well as the gradients of local model on each shared server example back to the server. At last, the server generates the perturbed data $\hat{\mathcal{D}}_s$ (Step 9) using the proposed bias-variance attack algorithm (see Alg. 3) with aggregations (model parameter average, bias gradients average, and variance gradients average) in the similar manner as FedAvg (McMahan et al., 2017). These aggregations can be privacy secured if additive homomorphic encryption (Acar et al., 2018) is applied.

5 EXPERIMENTS

5.1 SETTINGS

In this section, we evaluate the adversarial robustness of our proposed algorithm on four benchmark data sets: MNIST[4], Fashion-MNIST[5], CIFAR-10[6] and CIFAR-100[6]. The baseline models

[4] http://yann.lecun.com/exdb/mnist
[5] https://github.com/zalandoresearch/fashion-mnist
[6] https://www.cs.toronto.edu/~kriz/cifar.html

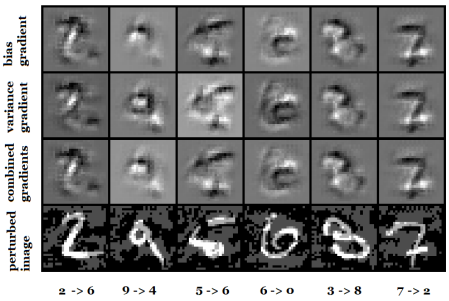

Figure 1: Visualizations of bias, variance, bias+variance, and perturbed images for MNIST.

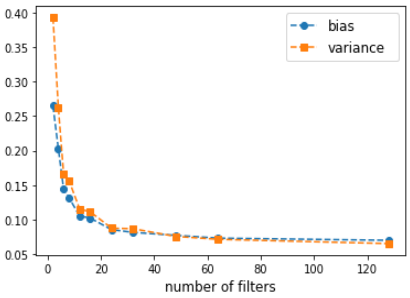

Figure 2: Bias-variance curve w.r.t. the CNN model complexity on MNIST.

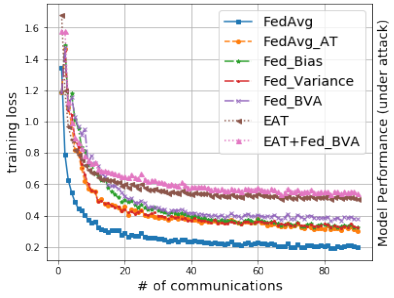

Figure 3: Convergence on Fashion-MNIST (PGD-20 attack)

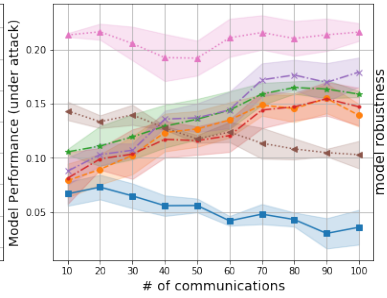

Figure 4: Performance on Fashion-MNIST (PGD-20 attack)

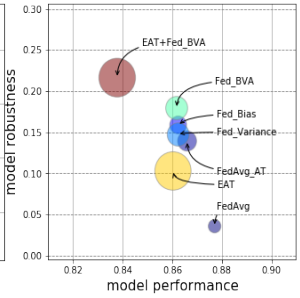

Figure 5: Efficiency on Fashion-MNIST (PGD-20 attack)

we used include: (1). `Centralized`: the training with one centralized model, which is identical to the federated learning case that only has one client ($K = 1$) with fraction ($F = 1$). (2). `FedAvg`: the classical federated averaging model (McMahan et al., 2017). (3). `FedAvg_AT`: The simplified version of our proposed method where the local clients perform adversarial training with the asymmetrical transmitted perturbed data generated on top of `FedAvg`'s aggregation. (4) - (6). `Fed_Bias, Fed_Variance, Fed_BVA`: Our proposed methods where the asymmetrical transmitted perturbed data is generated using the gradients of bias-only attack, variance-only attack, and bias-variance attack, respectively. (7). `EAT`: Ensemble adversarial training (Tramèr et al., 2018), where each client performs local adversarial training using Eq. (5), and their model updates are aggregated on server using `FedAvg`. For fair comparisons, all baselines are modified to the asymmetrical communications setting (`FedAvg` and `EAT` have clean $\mathcal{D}_s$ received), and all their initializations are set to be the same. (8). `EAT+Fed_BVA`: A combination of baselines (6) and (7). Note that baselines (7) and (8) have high computational requirements on client devices, and are usually not preferred in real scenarios.

For the defense model, we use a 4-layer CNN model for MNIST and Fashion-MNIST, and VGG9 architecture for CIFAR-10 and CIFAR-100. Regarding blackbox attacks, we apply ResNet18 (He et al., 2016), VGG11 (Simonyan & Zisserman, 2015), Xception (Chollet, 2017), and MobileNetV2 (Sandler et al., 2018) for CIFAR data, and provide a variety of models for MNIST and Fashion-MNIST by following the design of (Tramèr et al., 2018). The training is performed using the SGD optimizer with fixed learning rate of 0.01 and momentum of value 0.9. The trade-off coefficient between bias and variance is set to $\lambda = 0.01$ for all experiments. All hyper-parameters of federated learning are presented in Table 5 in the Appendix. We empirically demonstrate that these hyper-parameter settings are preferable in terms of both training accuracy and robustness (see the details of Fig. 6 - Fig. 14 in the Appendix). To evaluate the robustness of our federated learning algorithm against adversarial attacks, except for the clean model training, we perform FGSM (Goodfellow et al., 2015), PGD (Kurakin et al., 2017) with 10 and 20 steps towards the aggregated server model on the $\mathcal{D}_{test}$. Following (Tramèr et al., 2018; Wang et al., 2019), the maximum perturbations allowed are $\epsilon = 0.3$ on MNIST and Fashion-MNIST, and $\epsilon = \frac{16}{255}$ on CIFAR-10 and CIFAR-100 for both threat and defense models. For IID sampling, the data is shuffled and uniformly partitioned into each client; For non-IID setting, data is divided into $2F \cdot K$ shards based on sorted labels, then assigns each client with 2 shards. Thereby, each client will have data with at most two classes.

5.2 RESULT ANALYSIS

To analyze the properties of our proposed Fed_BVA framework, we present two visualization plots on MNIST using a trained CNN model where the bias and variance are both calculated on the training examples. In Fig. 1, we visualize the extracted gradients using adversarial attack from bias, variance, and bias-variance. Notice that the gradients of bias and variance are similar but with subtle differences in local pixel areas. However, according to Theorem 4.1, the gradient calculation of these two are quite different: bias requires the target label as input, but variance only needs the model output and main prediction. From another perspective, we also investigate the bias-variance magnitude relationship with varying model complexity. As shown in Fig. 2, with increasing model complexity (more convolutional filters in CNN), both bias and variance decrease. This result is different from the double-descent curve or bell-shape variance curve claimed in (Belkin et al., 2019; Yang et al., 2020). The reasons are twofold: First, their bias-variance definitions are from the MSE regression decomposition perspective, whereas our decomposition utilizes the concept of main prediction, and the generalization error is decomposed from the classification perspective; Second, their implementations only evaluate the bias and variance using training batches on one central model and thus is different from the definition which requires the variance to be estimated from multiple sub-models (in our scenario, client models).

The convergence plot of all baselines is presented in Fig. 3. We observe that FedAvg has the best convergence, and all robust training will have a slightly higher loss upon convergence. This matches the observations in (Madry et al., 2018) which state that training performance may be sacrificed in order to provide robustness for small capacity networks. For the model performance shown in Fig. 4, we observe that the aggregation of federated learning is vulnerable to adversarial attacks since both FedAvg and EAT have decreased performance with an increasing number of server-client communications. Other baselines that utilized the asymmetrical communications have increasing robustness with more communication rounds although only a small number of perturbed examples ($n_s = 64$) are transmitted. We also observe that when communication rounds reach 40, Fed_BVA starts to outperform EAT while the latter is even more resource-demanding than Fed_BVA (shown in Fig. 5, where the pie plot size represents the running time). Overall, bias-variance based adversarial training via asymmetric communication is both effective and efficient for robust federated learning.

For the comprehensive experiments in Table 1 and Table 2, it is easy to verify that our proposed model outperforms all other baselines regardless of the source of the perturbed examples (i.e., locally generated like EAT+Fed_BVA or asymmetrically transmitted from the server like Fed_BVA). Comparing with standard robust federated learning FedAvg_AT, the performance of Fed_BVA against adversarial attacks still increases $4\% - 13\%$ and $2\% - 9\%$ on IID and non-IID settings respectively, although Fed_BVA is theoretically suitable for the cases that clients have IID samples. In Table 3, we observe a similar trend where Fed_BVA outperforms FedAvg_AT on CIFAR-10 and CIFAR-100 (with $0.2\% - 10\%$ increases) when defending different types of adversarial examples. Comparing with strong local adversarial training baseline EAT, we also observe a maximum 13% accuracy increase when applying its bias-variance oriented baseline EAT+Fed_BVA. Overall, the takeaway is that without local adversarial training, using a bias-variance based robust learning framework will almost always outperform other baselines for defending FGSM and PGD attacks. When local adversarial training is allowed (e.g., client device has powerful computation ability), using bias-variance robust learning with local adversarial training will mostly have the best robustness.

We also conducted various additional experiments in Appendix A.5 which includes: (1) Comparison of efficiency and effectiveness of Fed_BVA using cross-entropy loss and MSE loss; (2) Comparison of single-step Fed_BVA and multi-step Fed_BVA in terms of the generation of $\hat{\mathcal{D}}_s$; (3) Three training scenarios of Fed_BVA that use client-specific adversarial examples or universal adversarial examples; (4) Ablation study in terms of the number of shared perturb examples n_s, optimizer's momentum, and the number of local epochs E; (5) Blackbox attacking transferability between various models on all four data sets under multiple settings.

6 RELATED WORK

Adversarial Machine Learning: While machine learning models have achieved remarkable performance over clean inputs, recent work (Goodfellow et al., 2015) showed that those trained models are vulnerable to adversarially chosen examples by adding the imperceptive noise to the clean inputs. In general, the adversarial robustness of centralized machine learning models have been explored from the following aspects: adversarial attacks (Carlini & Wagner, 2017; Athalye et al., 2018; Zhu et al.,

	IID				non-IID			
	Clean	FGSM	PGD-10	PGD-20	Clean	FGSM	PGD-10	PGD-20
Centralized	**0.991**$_{\pm 0.000}$	0.689$_{\pm 0.000}$	0.260$_{\pm 0.000}$	0.182$_{\pm 0.000}$	n/a	n/a	n/a	n/a
FedAvg	0.989$_{\pm 0.001}$	0.669$_{\pm 0.009}$	0.576$_{\pm 0.005}$	0.267$_{\pm 0.014}$	0.980$_{\pm 0.002}$	0.491$_{\pm 0.067}$	0.475$_{\pm 0.057}$	0.158$_{\pm 0.074}$
FedAvg_AT	0.988$_{\pm 0.000}$	0.802$_{\pm 0.001}$	0.745$_{\pm 0.014}$	0.512$_{\pm 0.042}$	0.974$_{\pm 0.005}$	0.649$_{\pm 0.066}$	0.615$_{\pm 0.045}$	0.363$_{\pm 0.066}$
Fed_Bias	0.986$_{\pm 0.000}$	0.812$_{\pm 0.009}$	0.788$_{\pm 0.021}$	0.583$_{\pm 0.036}$	0.971$_{\pm 0.004}$	0.679$_{\pm 0.040}$	0.627$_{\pm 0.078}$	0.394$_{\pm 0.103}$
Fed_Variance	0.985$_{\pm 0.001}$	0.803$_{\pm 0.007}$	0.779$_{\pm 0.014}$	0.572$_{\pm 0.019}$	0.973$_{\pm 0.005}$	0.684$_{\pm 0.004}$	0.622$_{\pm 0.049}$	0.395$_{\pm 0.049}$
Fed_BVA	0.986$_{\pm 0.001}$	0.818$_{\pm 0.003}$	0.804$_{\pm 0.009}$	0.613$_{\pm 0.020}$	0.969$_{\pm 0.002}$	0.705$_{\pm 0.009}$	0.664$_{\pm 0.013}$	0.469$_{\pm 0.031}$
EAT	0.981$_{\pm 0.000}$	**0.902**$_{\pm 0.001}$	0.907$_{\pm 0.001}$	0.811$_{\pm 0.004}$	0.972$_{\pm 0.002}$	0.789$_{\pm 0.016}$	0.721$_{\pm 0.018}$	0.415$_{\pm 0.035}$
EAT+Fed_BVA	0.980$_{\pm 0.001}$	0.901$_{\pm 0.006}$	**0.910**$_{\pm 0.004}$	**0.821**$_{\pm 0.013}$	0.965$_{\pm 0.005}$	**0.811**$_{\pm 0.020}$	**0.831**$_{\pm 0.013}$	**0.670**$_{\pm 0.014}$

Table 1: Accuracy of MNIST under white-box attacks in IID and non-IID settings

	IID				non-IID			
	Clean	FGSM	PGD-10	PGD-20	Clean	FGSM	PGD-10	PGD-20
Centralized	**0.882**$_{\pm 0.000}$	0.229$_{\pm 0.000}$	0.010$_{\pm 0.000}$	0.009$_{\pm 0.000}$	n/a	n/a	n/a	n/a
FedAvg	0.877$_{\pm 0.001}$	0.300$_{\pm 0.021}$	0.072$_{\pm 0.016}$	0.036$_{\pm 0.016}$	**0.804**$_{\pm 0.013}$	0.193$_{\pm 0.036}$	0.061$_{\pm 0.015}$	0.017$_{\pm 0.003}$
FedAvg_AT	0.866$_{\pm 0.001}$	0.490$_{\pm 0.021}$	0.170$_{\pm 0.014}$	0.139$_{\pm 0.011}$	0.730$_{\pm 0.023}$	0.445$_{\pm 0.065}$	0.136$_{\pm 0.044}$	0.087$_{\pm 0.042}$
Fed_Bias	0.862$_{\pm 0.001}$	0.505$_{\pm 0.015}$	0.199$_{\pm 0.007}$	0.159$_{\pm 0.003}$	0.709$_{\pm 0.025}$	0.460$_{\pm 0.038}$	0.149$_{\pm 0.067}$	0.115$_{\pm 0.054}$
Fed_Variance	0.862$_{\pm 0.002}$	0.496$_{\pm 0.012}$	0.201$_{\pm 0.012}$	0.157$_{\pm 0.017}$	0.719$_{\pm 0.036}$	0.499$_{\pm 0.081}$	**0.188**$_{\pm 0.025}$	**0.120**$_{\pm 0.038}$
Fed_BVA	0.862$_{\pm 0.003}$	0.528$_{\pm 0.016}$	0.210$_{\pm 0.023}$	0.180$_{\pm 0.027}$	0.710$_{\pm 0.045}$	0.495$_{\pm 0.030}$	0.141$_{\pm 0.021}$	0.093$_{\pm 0.028}$
EAT	0.860$_{\pm 0.005}$	**0.773**$_{\pm 0.029}$	0.191$_{\pm 0.012}$	0.103$_{\pm 0.013}$	0.791$_{\pm 0.012}$	0.597$_{\pm 0.033}$	0.071$_{\pm 0.050}$	0.027$_{\pm 0.023}$
EAT+Fed_BVA	0.838$_{\pm 0.009}$	0.715$_{\pm 0.011}$	**0.357**$_{\pm 0.024}$	**0.226**$_{\pm 0.006}$	0.735$_{\pm 0.020}$	**0.632**$_{\pm 0.015}$	0.164$_{\pm 0.035}$	0.106$_{\pm 0.039}$

Table 2: Accuracy of Fashion-MNIST under white-box attacks in IID and non-IID settings

	CIFAR-10				CIFAR-100			
	Clean	FGSM	PGD-10	PGD-20	Clean	FGSM	PGD-10	PGD-20
Centralized	**0.903**$_{\pm 0.003}$	0.288$_{\pm 0.001}$	0.206$_{\pm 0.001}$	0.074$_{\pm 0.005}$	**0.741**$_{\pm 0.003}$	0.166$_{\pm 0.012}$	0.049$_{\pm 0.004}$	0.032$_{\pm 0.003}$
FedAvg	0.890$_{\pm 0.002}$	0.225$_{\pm 0.022}$	0.207$_{\pm 0.004}$	0.062$_{\pm 0.008}$	0.730$_{\pm 0.003}$	0.161$_{\pm 0.009}$	0.113$_{\pm 0.009}$	0.035$_{\pm 0.006}$
FedAvg_AT	0.890$_{\pm 0.003}$	0.280$_{\pm 0.021}$	0.295$_{\pm 0.006}$	0.099$_{\pm 0.014}$	0.707$_{\pm 0.003}$	0.162$_{\pm 0.006}$	0.064$_{\pm 0.007}$	0.048$_{\pm 0.003}$
Fed_Bias	0.890$_{\pm 0.004}$	0.280$_{\pm 0.018}$	0.297$_{\pm 0.011}$	0.103$_{\pm 0.012}$	0.702$_{\pm 0.002}$	0.163$_{\pm 0.005}$	0.165$_{\pm 0.007}$	0.061$_{\pm 0.003}$
Fed_Variance	0.889$_{\pm 0.001}$	0.267$_{\pm 0.014}$	0.276$_{\pm 0.006}$	0.092$_{\pm 0.009}$	0.710$_{\pm 0.007}$	0.161$_{\pm 0.005}$	0.157$_{\pm 0.010}$	0.045$_{\pm 0.016}$
Fed_BVA	0.889$_{\pm 0.003}$	0.286$_{\pm 0.013}$	0.301$_{\pm 0.003}$	0.104$_{\pm 0.012}$	0.709$_{\pm 0.003}$	0.163$_{\pm 0.007}$	0.165$_{\pm 0.008}$	0.062$_{\pm 0.005}$
EAT	0.833$_{\pm 0.003}$	0.596$_{\pm 0.003}$	0.667$_{\pm 0.007}$	0.561$_{\pm 0.002}$	0.661$_{\pm 0.001}$	0.267$_{\pm 0.002}$	0.206$_{\pm 0.002}$	0.188$_{\pm 0.001}$
EAT+Fed_BVA	0.833$_{\pm 0.003}$	**0.598**$_{\pm 0.002}$	**0.668**$_{\pm 0.001}$	**0.564**$_{\pm 0.003}$	0.657$_{\pm 0.002}$	**0.272**$_{\pm 0.003}$	**0.332**$_{\pm 0.003}$	**0.211**$_{\pm 0.002}$

Table 3: Accuracy of CIFAR-10 and CIFAR-100 under white-box attacks

2019), defense (or robust model training) (Madry et al., 2018; Carlini et al., 2019; Tramèr et al., 2018) and interpretable adversarial robustness (Schmidt et al., 2018; Tsipras et al., 2018).

Federated Learning: Federated learning with preserved privacy (Konečnỳ et al., 2016; McMahan et al., 2017; Hard et al., 2018) and knowledge distillation (Chang et al., 2019; Jeong et al., 2018) has become prevalent in recent years. Meanwhile, the vulnerability of federated learning to backdoor attacks has also been explored by (Bagdasaryan et al., 2018; Bhagoji et al., 2019; Xie et al., 2019). Following their work, multiple robust federated learning models (Fang et al., 2019; Pillutla et al., 2019; Portnoy & Hendler, 2020; Mostafa, 2019) are also proposed and studied. In this paper, we studied the federated learning's adversarial vulnerability after model deployment from the perspective of bias-variance analysis. This is in sharp contrast to the existing work that focused on the model robustness against the Byzantine failures.

Bias-Variance Decomposition: Bias-variance decomposition (Geman et al., 1992) was originally introduced to analyze the generalization error of a learning algorithm. Then, a generalized bias-variance decomposition (Domingos, 2000; Valentini & Dietterich, 2004) was studied in the classification setting which enabled flexible loss functions (e.g., squared loss, zero-one loss). More recently, bias-variance trade-off was experimentally evaluated on modern neural network models (Neal et al., 2018; Belkin et al., 2019; Yang et al., 2020).

7 Conclusion

In this paper, we proposed a novel robust federated learning framework, in which the aggregation incurred loss during the server's aggregation is dissected into a bias part and a variance part. Our approach improves the model robustness through adversarial training by supplying a few bias-variance perturbed samples to the clients via asymmetrical communications. Extensive experiments have been conducted where we evaluated its performance from various aspects on several benchmark data sets. We believe the further exploration of this direction will lead to more findings on the robustness of federated learning.

REFERENCES

Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4):79:1–79:35, 2018.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning*, pp. 284–293, 2018.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643, 2019.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy*, pp. 39–57. IEEE, 2017.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019.

Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *CoRR*, abs/1912.11279, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 1800–1807, 2017.

Pedro Domingos. A unified bias-variance decomposition and its applications. In *International Conference on Machine Learning*, pp. 231–238, 2000.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. *CoRR*, abs/1911.11815, 2019.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 770–778. IEEE Computer Society, 2016.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *CoRR*, abs/1811.11479, 2018.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Hesham Mostafa. Robust federated learning through representation matching and adaptive hyperparameters. *CoRR*, abs/1912.13075, 2019.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

Venkata Krishna Pillutla, Sham M. Kakade, and Zaïd Harchaoui. Robust aggregation for federated learning. *CoRR*, abs/1912.13445, 2019.

Amit Portnoy and Danny Hendler. Towards realistic byzantine-robust federated learning. *CoRR*, abs/2004.04986, 2020.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 4510–4520, 2018.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 5014–5026, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR*, 2015.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick Drew McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2018.

Giorgio Valentini and Thomas G Dietterich. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research*, pp. 725–775, 2004.

Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *Proceedings of the 36th International Conference on Machine Learning,*, volume 97, pp. 6586–6595, 2019.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pp. 1509–1519, 2017.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019.

Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. *arXiv preprint arXiv:2002.11328*, 2020.

Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pp. 7614–7623, 2019.

A APPENDIX

A.1 NOTATION SUMMARY

Notation	Definition
$\mathcal{D}_s = \{(x_i^s, t_i^s)\}_{i=1}^{n_s}$	Training set with n_s examples in the central server
$\mathcal{D}_k = \{(x_i^k, t_i^k)\}_{i=1}^{n_k}$	Local training set with n_k examples in the k^{th} client
$\mathcal{D}_{test}$	Test data set in the central server
$f(\cdot)$	Learning algorithm
$L(\cdot, \cdot)$	Loss function over prediction and target
w_k	Local model parameters over $\mathcal{D}_k$
w_G	Server's global model parameters
$\text{Aggregate}(w_1, w_2, \cdots, w_K)$	Synchronous aggregation function over local parameters
$f_{\mathcal{D}_k}(x; w_k)$ ($f_{\mathcal{D}_k}$ for short)	Local model prediction on x with parameters w_k
C	Number of label classes in the training set, i.e., $t \in \mathbb{R}^C$
K	Number of clients
E	Number of local epochs
F	Fraction of clients selected on each round
B	Batch size of local client
η	Learning rate
y_*, y_m	Optimal prediction and main prediction
$B(\cdot), V(\cdot), N(\cdot)$	Expected bias, variance and irreducible noise
$\mathbb{E}[\cdot]$	Expected value
$B(\cdot; w_1, \cdots, w_K)$	Empirical estimated bias over training sets $\{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$
$V(\cdot; w_1, \cdots, w_K)$	Empirical estimated variance over training sets $\{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$
$\Omega(\cdot)$	Perturbation constraint
ϵ	Perturbation magnitude
$\hat{x}$	Adversarial counterpart of an example x, i.e., $\hat{x} \in \Omega(x)$
$H(\cdot)$	Entropy

Table 4: Notation

A.2 PROOF OF THEOREM

Proof of Theorem 4.1. Theorem 4.1 states that assume $L(\cdot, \cdot)$ is the cross-entropy loss function, then the empirical estimated main prediction y_m for an input example (x, t) has the following closed-form expression:

$$y_m(x; w_1, \cdots, w_K) = \frac{1}{K} \sum_{k=1}^{K} f_{\mathcal{D}_k}(x; w_k)$$

Furthermore, the empirical bias and variance as well as their gradients over input x are estimated as:

$$B(x; w_1, \cdots, w_K) = \frac{1}{K} \sum_{k=1}^{K} L(f_{\mathcal{D}_k}(x; w_k), t); \qquad V(x; w_1, \cdots, w_K) = L(y_m, y_m) = H(y_m)$$

where $H(y_m) = -\sum_{j=1}^{C} y_m^{(j)} \log y_m^{(j)}$ is the entropy of the optimal prediction y_m and C is the number of classes.

Proof. We first calculate the main prediction. Let

$$\begin{aligned}
\mathcal{M} &= \frac{1}{K} \sum_{k=1}^{K} L(f_{\mathcal{D}_k}(x; w_k), y') \\
&= -\frac{1}{K} \sum_{k=1}^{K} \left(f_{\mathcal{D}_k}(x; w_k) \cdot \log y' + (1 - f_{\mathcal{D}_k}(x; w_k)) \cdot \log (1 - y') \right)
\end{aligned}$$

Take the derivative of $\mathcal{M}$ with respect to y', we have

$$\frac{\partial \mathcal{M}}{\partial y'} = -\frac{1}{K}\sum_{k=1}^{K}\Big(\frac{f_{\mathcal{D}_k}(x;w_k)}{y'} - \frac{1-f_{\mathcal{D}_k}(x;w_k)}{1-y'}\Big) = -\frac{1}{K}\sum_{k=1}^{K}\frac{f_{\mathcal{D}_k}(x;w_k)-y'}{y'(1-y')}$$

By letting $\partial\mathcal{M}/\partial y' = 0$, we have

$$y_m(x;w_1,\cdots,w_K) = \frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x;w_k)$$

Then for bias and variance, from their definitions, we have,

$$\begin{aligned}
&B(x;w_1,\cdots,w_K)\\
=&L\Big(\frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x;w_k), t\Big)\\
=&-\frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x;w_k)\cdot\log t + \Big(1-\frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x;w_k)\Big)\cdot\log(1-t)\\
=&-\frac{1}{K}\sum_{k=1}^{K}\Big(f_{\mathcal{D}_k}(x;w_k)\cdot\log t + (1-f_{\mathcal{D}_k}(x;w_k))\cdot\log(1-t)\Big)\\
=&\frac{1}{K}\sum_{k=1}^{K} L(f_{\mathcal{D}_k}(x;w_k),t)
\end{aligned}$$

and

$$\begin{aligned}
V(x) &= \frac{1}{K}\sum_{k=1}^{K} L(f_{\mathcal{D}_k}(x;w_k), y_m)\\
&= -\frac{1}{K}\sum_{k=1}^{K}\Big(f_{\mathcal{D}_k}(x;w_k)\log y_m + (1-f_{\mathcal{D}_k}(x;w_k))\log(1-\log y_m)\Big)\\
&= -y_m\log y_m - (1-y_m)\log(1-\log y_m)\\
&= H(y_m)
\end{aligned}$$

which completes the proof. □

A.3 EXPERIMENTAL SETTING

Regarding the experimental settings, all data sets have performed 100 clients-served communications using federated learning. For MNIST and Fashion-MNIST, we deployed 100 clients (each has 1% of the overall data) and 10% of them would be selected for model updating (with 50 local training epochs) and aggregation on server. In client's training stage, there will be $n_s = 64$ shared examples being transmitted from the server using asymmetrical communication. Comparing with the total number of examples in data set (i.e., 60k examples), number of the shared examples, n_s, would be only 0.1% of the data set's size.

	# comms.	# clients (K)	fraction (F)	# epoch (E)	local batch (B)	# shared (n_s)	# categories
MNIST	100	100	0.1	50	64	64	10
Fashion-MNIST	100	100	0.1	50	64	64	10
CIFAR-10	100	20	0.2	5	128	30	10
CIFAR-100 (coarse)	100	20	0.2	5	128	60	20

Table 5: Learning setting of Fed_BVA

For the CIFAR-10 and CIFAR-100 data sets (we use its 20 class version with coarse labels), we deployed 20 clients (each has 5% of the over all data) since the data examples have more variations in terms of their categories and cluttered backgrounds. Among these clients, 20% of them would be selected for model updating (with 5 local epochs since we utilized deeper models with longer training time per epoch). Similarly, we only transmitted 30 or 60 examples from the server to the clients in CIFAR-10 or CIFAR-100. Due to the complexity of CIFAR's data distributions, we enforce

n_s to be spread out in every categories (i.e., both CIFAR-10 and CIFAR-100 with coarse labels will have 3 shared examples among these clients per category). An alternative is selecting n_s randomly from the data sets, and in this scenario, we only observer slight perform drop in all settings where the model behavior remains unchanged. In CIFAR-10, we choose n_s to be only 0.05% of the data set's size.

A.4 NETWORK ARCHITECTURES

For the MNIST and Fashion-MNIST data sets, the CNN network structure is shared since the input image examples have the same dimensions. The detail is shown in Table 6. Conv1 and Conv2 denote the convolution block that may have convolution layer together with activation unit (e.g., ReLU, Tahn) and(or) batch normalization (BN). For short, here, $[5 \times 5, 10] \times 1$ denotes one convolution layer with 10 filters of size 5×5.

Layers	**4-layer CNN**
Conv1	[5 × 5, 10] + ReLU
Pool1	2 × 2 Max Pooling, Stride 2
Conv2	[5 × 5, 20] + ReLU
Pool2	2 × 2 Max Pooling, Stride 2
Dropout	0.5
FC1	50 + ReLU
Dropout	0.5
FC2	10

Table 6: The 4-layer CNN architecture for MNIST and Fashion-MNIST

For the CIFAR-10 and CIFAR-100 data set, the VGG9 network structure is shown in Table 7. It should be noticed that state-of-the-art approaches have achieved a better test accuracy on CIFAR, nevertheless, this model is sufficient for our experimental needs since our goal is to evaluate the robust federated model instead of achieving the best possible accuracy on testing stage.

Layers	**VGG9**
Conv1	[3 × 3, 32] + BN + ReLU
Conv2	[3 × 3, 64] + ReLU
Pool1	2 × 2 Max Pooling, Stride 2
Conv3	[3 × 3, 128] + BN + ReLU
Conv4	[3 × 3, 128] + ReLU
Pool2	2 × 2 Max Pooling, Stride 2
Dropout	0.05
Conv5	[3 × 3, 256] + BN + ReLU
Conv6	[3 × 3, 256] + ReLU
Pool3	2 × 2 Max Pooling, Stride 2
Dropout	0.1
FC1	1024
FC2	512
Dropout	0.1
FC3	10 or 20

Table 7: The VGG-9 architecture for CIFAR-10 and CIFAR-100

A.5 ADDITIONAL EXPERIMENTS

A.5.1 MSE V.S. CROSS-ENTROPY

Both cross-entropy (CE) and mean squared error (MSE) loss functions could be used for training a neural network model. In our paper, the loss function of neural networks determines the derivation of bias and variance terms used for producing the adversarial examples. Specifically, we show that

when using CE loss function, the empirical estimate of bias and variance as follows:

$$B_{CE}(x; w_1, \cdots, w_K) = \frac{1}{K}\sum_{k=1}^{K} L(f_{\mathcal{D}_k}(x; w_k), t) \text{ and } V_{CE}(x; w_1, \cdots, w_K) = H(y_m)$$

and their corresponding gradients over input x are:

$$\nabla_x B_{CE}(x; w_1, \cdots, w_K) = \frac{1}{K}\sum_{k=1}^{K} \nabla_x L(f_{\mathcal{D}_k}(x; w_k), t)$$

$$\nabla_x V_{CE}(x; w_1, \cdots, w_K) = -\frac{1}{K}\sum_{k=1}^{K}\sum_{j=1}^{C}(\log y_m^{(j)} + 1) \cdot \nabla_x f_{\mathcal{D}_k}^{(j)}(x; w_k)$$

In the similar way, we show that when using MSE loss function, the empirical estimate of bias and variance are as follows:

$$B_{MSE}(x; w_1, \cdots, w_K) = ||\frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x; w_k) - t||_2^2$$

$$V_{MSE}(x; w_1, \cdots, w_K) = \frac{1}{K-1}\sum_{k=1}^{K} ||f_{\mathcal{D}_k}(x; w_k) - \frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x; w_k)||_2^2$$

and their gradients over input x are:

$$\nabla_x B_{MSE}(x; w_1, \cdots, w_K) = \left(\frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x; w_k) - t\right)\left(\frac{1}{K}\sum_{k=1}^{K} \nabla_x f_{\mathcal{D}_k}(x; w_k)\right)$$

$$\nabla_x V_{MSE}(x; w_1, \cdots, w_K)$$

$$= \frac{1}{K-1}\sum_{k=1}^{K}\left(\left(f_{\mathcal{D}_k}(x; w_k) - \frac{1}{K}\sum_{k=1}^{K} f_{\mathcal{D}_k}(x; w_k)\right)\left(\nabla_x f_{\mathcal{D}_k}(x; w_k) - \frac{1}{K}\sum_{k=1}^{K} \nabla_x f_{\mathcal{D}_k}(x; w_k)\right)\right)$$

It can be seen that the empirical estimate of $\nabla_x B_{MSE}(x; w_1, \cdots, w_K)$ has much higher computational complexity than $\nabla_x B_{CE}(x; w_1, \cdots, w_K)$ because it involves the gradient calculation of prediction vector $f_{\mathcal{D}_k}(x; w_k)$ over the input tensor x. Besides, it is easy to show that the empirical estimate of $\nabla_x V_{MSE}(x; w_1, \cdots, w_K)$ is also computationally expensive.

We experimentally compare the CE and MSE loss functions in our framework. Table 8 reports the adversarial robustness of our federated framework w.r.t. FGSM attack ($\epsilon = 0.3$) on MNIST with IID setting (the best resutls are indicated in bold). It is observed that (1) our framework with MSE loss function has significantly larger running time; (2) the robustness of our framework with MSE loss function becomes slightly weaker, which might be induced by the weakness of MSE loss function in training neural networks under the classification setting.

	Clean	Fed_BVA		
		BiasOnly	VarianceOnly	BVA
Cross-entropy	0.5875 (**38.13s**)	**0.7627 (47.58s)**	**0.7594 (63.46s)**	**0.7756 (63.67s)**
MSE	**0.6011** (39.67s)	0.7112 (65.03s)	0.7108 (162.40s)	0.7119 (179.60s)

Table 8: Adversarial robustness with different loss functions and running time (second/epoch)

A.5.2 BV-PGD v.s. BV-FGSM

Our bias-variance attack could be naturally generalized to any gradient-based adversarial attack algorithms when the gradients of bias $B(\cdot)$ and variance $V(\cdot)$ w.r.t. x are tractable to be estimated from clients' models learned on finite training sets. Here, we empirically compare the adversarial robustness of the proposed framework trained with bias-variance guided PGD (BV-PGD) adversarial examples and bias-variance guided FGSM (BV-FGSM) adversarial examples. Table 9 provides our results on w.r.t. FGSM and PGD attacks ($\epsilon = 0.3$) on MNIST with IID and non-IID settings. Compared to FedAvg, our framework Fed_BVA with either BV-FGSM or BV-PGD could largely improve the model robustness against adversarial noise (the best results are indicated in bold). Furthermore, BV-PGD could potentially improve white-box robustness on multi-step attacks, but it

is often more computationally demanding. As a comparisons, BV-FGSM is more robust against single-step attacks.

	IID				non-IID			
	Clean	FGSM	PGD-10	PGD-20	Clean	FGSM	PGD-10	PGD-20
FedAvg	**0.9863**	0.5875	0.6203	0.2048	0.9462	0.1472	0.5254	0.0894
Fed_BVA (BV-FGSM training)	0.9837	**0.7756**	0.7927	0.5699	**0.9671**	**0.6696**	0.6953	0.4717
Fed_BVA (BV-PGD training)	0.9849	0.7565	**0.8399**	**0.6317**	0.9670	0.6592	**0.7836**	**0.5752**

Table 9: Adversarial robustness with BV-PGD and BV-FGSM

A.5.3 Alternative Training Strategies for Fed_BVA

In our `Fed_BVA` framework, we propose to maximize the overall generalization error induced by bias and variance from different clients in order to generate the adversarial examples for robust training. Under this setting, the generated adversarial examples on the server are shared to all the clients for local adversarial training. In particular, we also found that when using the cross-entropy loss function, the estimated gradients of both bias and variance can be considered as the average of clients' local gradients over input x (see Section 4.1). This motivates us to consider several alternative training strategies by generating the client-specific adversarial examples on the server.

To be more specific, we have the following three training strategies:

- **S1:** Following the bias-variance decomposition, we can generate the adversarial examples to maximize the bias and variance from all clients' predictions. In this case, the generated adversarial examples on the server will be shared by all the local clients. We adopted this strategy in our `Fed_BVA` algorithm. It allows to guarantee the minimization of generalization error from the perspective of bias-variance decomposition when training with these adversarial examples, thus leading to a robust federated learning model.
- **S2:** The bias-variance decomposition with cross-entropy loss function indicates that we can generate the client-specific adversarial examples as follows.
$$\nabla_x B_k(x; w_k) = \nabla_x L(f_{\mathcal{D}_k}(x; w_k), t) \quad \forall x \in \mathcal{D}_s \quad k = 1, \cdots, K$$
$$\nabla_x V_k(x; w_k) = \sum_{j=1}^{C} (\log y_m^{(j)} + 1) \nabla_x f_{\mathcal{D}_k}^{(j)}(x; w_k) \quad \forall x \in \mathcal{D}_s \quad k = 1, \cdots, K$$
Then, if we using FGSM for attacking, the adversarial example on the k^{th} client would be:
$$\hat{x}^k_{\text{BV}-\text{FGSM}} := x + \epsilon \cdot \text{sign}\left(\nabla_x \left(B_k(x; w_k) + \lambda V_k(x; w_k)\right)\right)$$
We would like to point out that the gradient estimate of client-specific variance also relies on the main prediction y_m. But in this case, it allows every client to have different adversarial examples $\hat{\mathcal{D}}_s$. Intuitively, this training strategy further decompose the bias and variance into individual client-specific terms.
- **S3:** Another training strategy is to use every local client model to generate the adversarial examples on the server individually as follows.
$$\nabla_x B_k(x; w_k) = \nabla_x L(f_{\mathcal{D}_k}(x; w_k), t) \quad \forall x \in \mathcal{D}_s \quad k = 1, \cdots, K$$
Similarly, we can have:
$$\hat{x}^k_{\text{BV}-\text{FGSM}} := x + \epsilon \cdot \text{sign}\nabla_x \left(B_k(x; w_k)\right)$$
It can be considered as a special case of **S2** with $\lambda = 0$. In this case, every client will only use its own model parameters to generate the client-specific adversarial examples on the server. This strategy is actually a simple extension of the conventional adversarial training (with centralized data) in federated learning setting.

We conduct the ablation study to compare different training strategies in our `Fed_BVA` framework. In this case, we use $K = 10$ clients with fraction $F = 1$ and local epoch $E = 5$. Other hyper-parameters and model architecture setting are the same as our previous experiments (see Table 5). Table 10 provides the performance of adversarial robustness using our `Fed_BVA` framework on MNIST data set with both IID and non-IID settings. It is observed that `Fed_BVA` with **S1** has the best robustness in most cases compared to other heuristic training strategies **S2** and **S3**. This

indicates that bias and variance could provide better direction to generate the adversarial examples for robust training. In contrast, detecting the adversarial examples with individual directions in **S2** for each clients might be suboptimal.

	IID				**Non-IID**			
	Clean	FGSM	PGD-10	PGD-20	Clean	FGSM	PGD-10	PGD-20
S1	0.9879	**0.7445**	**0.7425**	**0.4504**	**0.9005**	**0.5292**	0.6769	**0.4333**
S2	0.9890	0.7432	0.7306	0.4357	0.8894	0.5125	**0.6800**	0.4323
S3	**0.9906**	0.7301	0.7044	0.3997	0.8888	0.4947	0.6573	0.3800

Table 10: Robust training with different strategies on MNIST

A.5.4 Parameter Analysis

In this part, we perform parameter analysis regarding a few key hyper-parameters that have high influence on the model performance. Since our target is to train a federated model where the robustness of the model is high but the accuracy still remains. For that purpose, we have the following three sets of experimental plots to guide us choosing the optimal hyper-parameters used in the experiments.

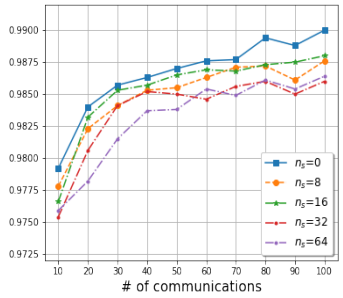

Figure 6: Accuracy of clean training with varying n_s

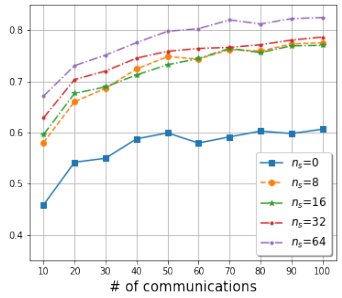

Figure 7: Accuracy under FGSM attack with varying n_s

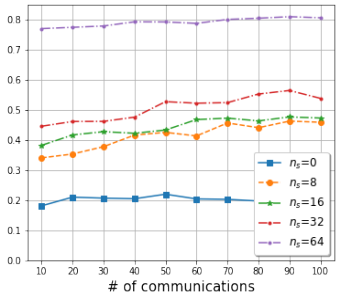

Figure 8: Accuracy under PGD-20 attack with varying n_s

Number of shared perturbed samples n_s. From Fig. 6, we see that the accuracy of `FedAvg` (i.e., $n_s = 0$) has the best accuracy as we expected. For `Fed_BVA` with varying size of asymmetrical transmitted perturbed samples (i.e., $n_s = 8, 16, 32, 64$), its performance drops slightly with increasing n_s (on average drop of 0.05% per plot). As a comparison, the robustness on test set $\mathcal{D}_{test}$ increases dramatically with increasing n_s (increasing ranges from 18% to 22% under FGSM attack and ranges from 15% to 60% under PGD-20 attack). However, choosing large n_s would have high model robustness but also suffer from the high communication cost. In experiment, we choose $n_s = 64$ for MNIST for the ideal trade-off point.

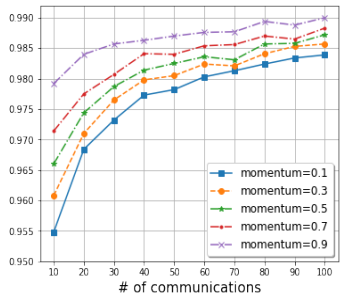

Figure 9: Accuracy of clean training with varying momentum

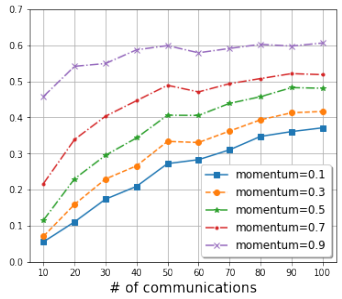

Figure 10: Accuracy under FGSM attack with varying momentum

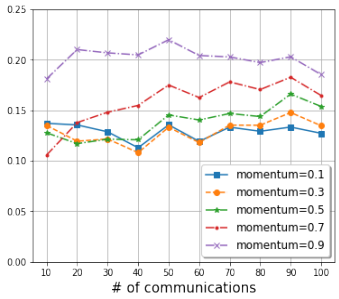

Figure 11: Accuracy under PGD-20 attack with varying momentum

Momentum. We also care about the choice of options in the SGD optimizer. As seen in Fig. 9, the accuracy of clean training is monotonically increase with momentum. Interestingly, we also observe that the federated model is less vulnerable when momentum is large no matter the adversarial attack

is FGSM or PGD-20 on test set $\mathcal{D}_{test}$, see Fig. 10 and Fig. 11. This phenomenon is also monotonically observed when we changing momentum from 0.1 to 0.9 and this suggests us to choose momentum as 0.9 through all experiments.

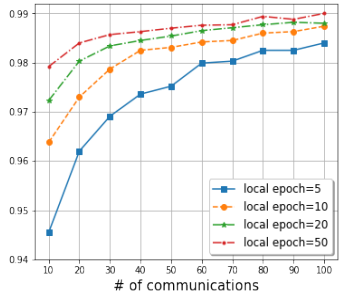

Figure 12: Accuracy of clean training with varying local epoch

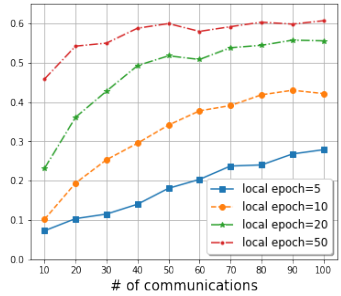

Figure 13: Accuracy under FGSM attack with varying local epoch

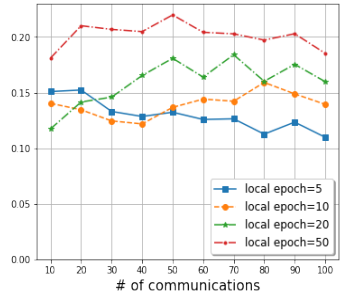

Figure 14: Accuracy under PGD-20 attack with varying local epoch

Local epochs E. The third important factor of federated learning is the number of epoch E we should use. In Fig. 12, we clearly see that more local epochs in each client leads to more accurate aggregated server model in prediction. Similarly, its robustness against both FGSM and PGD-20 attack on test set $\mathcal{D}_{test}$ also have the best performance when local epochs are large on device. Hence, in our experiments, if the on-device computational cost is not very high (large data example size, deep models with many layers), we choose $E = 50$. Otherwise, we will reduce E to a smaller number accordingly.

A.5.5 BLACKBOX ATTACK RESULTS

Using blackbox attack, we test the transferability of single-step attack (i.e., FGSM) and multi-step attack (i.e., PGD) on various federated learning baseline models. For MNIST and Fashion-MNIST, the architectures for threat models are shown in Table 11.

Model A	**Model B**	**Model C**	**Model D**
Conv[5×5, 64] + ReLU	Dropout[0.2]	Conv[3×3, 128] + Tahn	FC[512] + ReLU
Conv[5×5, 64] + ReLU	Conv[8×8, 64] + ReLU	MaxPool[2 × 2, Stride=2]	Dropout[0.5]
Dropout[0.25]	Conv[6×6, 128] + ReLU	Conv[3×3, 64] + Tahn	FC[512] + ReLU
FC[128] + ReLU	Conv[5×5, 128] + ReLU	MaxPool[2 × 2, Stride=2]	Dropout[0.5]
Dropout[0.5]	Dropout[0.5]	FC[128] + ReLU	FC[10]
FC[10]	FC[10]	FC[10]	

Table 11: Neural network architectures used for blackbox attacks on MNIST and Fashion-MNIST

For MNIST under the setting of IID|non-IID (see Table 12 and Table 13), we observe maximum $27\%|28\%$ accuracy drop on `FedAvg` under various blackbox attacks. As as comparison, the robust federated learning model with global transmitted perturb samples (i.e., `FedAvg_AT, Fed_Bias, Fed_Variance, Fed_BVA`) will have increase robustness with maximum of $12\%|14\%$ accuracy drop on best baselines. For the more computation-demanding local robust training methods (i.e., `EAT` and `EAT+Fed_BVA`), the maximum accuracy drop are only $6\%|11\%$ respectively.

For Fashion-MNIST data set under the IID|non-IID settings (see Table 14 and Table 15), similar trends are observed. We see that without any adversarial training, `FedAvg` will suffer a maximum of $74\%|64\%$ accuracy lost. For comparison, the robust federated learning model with global transmitted perturb samples (i.e., `FedAvg_AT, Fed_Bias, Fed_Variance, Fed_BVA`) will have increase robustness with maximum of $41\%|17\%$ accuracy drop on best baselines. For the computation-demanding local robust training methods (i.e., `EAT` and `EAT+Fed_BVA`), the maximum accuracy drop are only $27\%|14\%$, respectively.

On both the MNIST and Fashion-MNIST data sets, we also observe that the single-step adversarial attacks (i.e., FGSM) have higher transiferability than these generated using multi-step adversarial at-

MNIST (IID)	**Source** (FGSM attack)				**Source** (PGD-20 attack)			
Target	A	B	C	D	A	B	C	D
FedAvg	0.7713	0.9229	0.8915	0.717	0.8692	0.9632	0.924	0.9582
FedAvg_AT	0.8620	0.933	0.9296	0.8335	0.9241	0.9633	0.9479	0.9636
Fed_Bias	0.8849	0.9369	0.9342	0.8563	0.9362	0.9628	0.9541	0.9644
Fed_Variance	0.8828	0.9366	0.932	0.85	0.9342	0.9617	0.9517	0.9628
Fed_BVA	0.8826	0.9396	0.9353	0.862	0.9363	0.9657	0.9560	0.9652
EAT	**0.9433**	**0.9594**	**0.9598**	**0.9222**	0.9644	0.9695	**0.9689**	**0.9693**
EAT+Fed_BVA	**0.9433**	0.9578	0.9585	0.9188	**0.9647**	**0.9698**	0.9679	0.9675

Table 12: MNSIT IID accuracy towards blackbox adversarial examples on transferability between models

MNIST (non-IID)	**Source** (FGSM attack)				**Source** (PGD-20 attack)			
Target	A	B	C	D	A	B	C	D
FedAvg	0.7827	0.8659	0.8756	0.6954	0.8844	0.9304	0.9073	0.9372
FedAvg_AT	0.8723	0.9013	0.9117	0.8234	0.9142	**0.9368**	0.9301	**0.9420**
Fed_Bias	0.8234	0.8231	0.8712	0.7295	0.882	0.8907	0.8948	0.9078
Fed_Variance	0.8513	0.8685	0.8967	0.7887	0.9103	0.9234	0.9253	0.9328
Fed_BVA	0.856	0.8754	0.8955	0.7796	0.9067	0.9177	0.9204	0.9282
EAT	0.8877	0.9014	0.9187	0.8456	0.9194	0.9324	**0.9362**	0.9377
EAT+Fed_BVA	**0.8967**	**0.9152**	**0.9225**	**0.8592**	**0.9245**	0.9334	0.934	0.9354

Table 13: MNSIT non-IID accuracy towards blackbox adversarial examples on transferability between models

Fashion-MNIST (IID)	**Source** (FGSM attack)				**Source** (PGD-20 attack)			
Target	A	B	C	D	A	B	C	D
FedAvg	0.3997	0.2554	0.5355	0.1451	0.5674	0.4634	0.6404	0.5983
FedAvg_AT	0.6141	0.5353	0.6689	0.4011	0.7026	0.6466	0.732	0.6985
Fed_Bias	0.6348	0.5719	0.6764	0.4399	0.7135	0.6541	0.7372	0.7044
Fed_Variance	0.6354	0.5516	0.6763	0.4134	0.7089	0.6406	0.7388	0.6945
Fed_BVA	0.6191	0.5268	0.6592	0.4291	0.6987	0.6241	0.7279	0.6987
EAT	**0.7466**	**0.7334**	**0.7688**	**0.5819**	0.7645	**0.7367**	0.7719	0.7374
EAT+Fed_BVA	0.7343	0.7089	0.7653	0.52	**0.7691**	0.7358	**0.7766**	**0.7381**

Table 14: Fashion-MNSIT IID accuracy towards blackbox adversarial examples on transferability between models

tacks (i.e., PGD). That is, the accuracy after PGD-20 blackbox attacking is higher than the accuracy after FGSM blackbox attack. This phenomenon is observed on all results of Table 12 - Table 15, and this match with the conclusion of (Tramèr et al., 2018).

For the CIFAR data set, we pretrained the ResNet18 (He et al., 2016), VGG11 (Simonyan & Zisserman, 2015), Xception (Chollet, 2017), and MobileNetV2 (Sandler et al., 2018) as the source threat models for generating the single-step and multi-step adversarial examples.

For the blackbox transfer attacking results (see Table 16 and Table 17), we see that CIFAR data set is more robust (bear less accuracy loss after attack) towards blackbox attack in general than the simpler data sets such as MNIST and Fashion-MNIST. Even though, a similar trend of results are also observed. We observe that without any adversarial training, `FedAvg` will suffer a maximum of $28\%|22\%$ accuracy lost. For comparison, the robust federated learning model with global transmitted perturb samples (i.e., `FedAvg_AT, Fed_Bias, Fed_Variance, Fed_BVA`) will have increase robustness with maximum of $23\%|15\%$ accuracy drop on best baselines. For the computation-demanding local robust training methods (i.e., `EAT` and `EAT+Fed_BVA`), the maximum accuracy drop are only $3\%|7\%$, respectively. Thus, it is straightforward to see that CIFAR is

Fashion-MNIST (non-IID)	**Source** (FGSM attack)				**Source** (PGD-20 attack)			
Target	A	B	C	D	A	B	C	D
FedAvg	0.431	0.2487	0.4814	0.1728	0.5694	0.4251	0.6304	0.5472
FedAvg_AT	0.6174	0.5627	0.6396	0.4426	0.6752	0.6194	0.6805	0.6516
Fed_Bias	0.5936	0.5254	0.608	0.4492	0.6304	0.5753	0.6451	0.6132
Fed_Variance	0.6205	0.5924	0.6405	0.5174	0.6794	0.6315	0.6834	0.6626
Fed_BVA	0.5761	0.4977	0.6007	0.4901	0.6284	0.5866	0.6482	0.6419
EAT	0.6049	0.4979	0.5991	0.4529	0.6429	0.5896	0.6538	0.6458
EAT+Fed_BVA	**0.6864**	**0.6263**	**0.7007**	**0.569**	**0.7027**	**0.6636**	**0.7087**	**0.6828**

Table 15: Fashion-MNSIT non-IID accuracy towards blackbox adversarial examples on transferability between models

more vulnerable towards multi-step blackbox adversarial attack, but the local adversarial training methods would significantly improve its robustness.

CIFAR-10	**Source** (FGSM attack)				**Source** (PGD-20 attack)			
Target	ResNet_{18}	VGG_{11}	Xception	$\text{MobileNet}_{\text{v2}}$	ResNet_{18}	VGG_{11}	Xception	$\text{MobileNet}_{\text{v2}}$
FedAvg	0.7067	0.6877	0.6894	0.7929	0.6109	0.6233	0.5973	0.7870
FedAvg_AT	0.7422	0.7104	0.7201	0.8078	0.6950	0.6704	0.6608	0.8077
Fed_Bias	0.7400	0.7025	0.7146	0.7988	0.6898	0.6669	0.6538	0.7990
Fed_Variance	0.7384	0.7042	0.7186	0.8096	0.6770	0.6564	0.6480	0.8085
Fed_BVA	0.7435	0.7059	0.7219	0.8086	0.6927	0.6687	0.6639	0.8087
EAT	0.8205	0.8063	0.8146	0.8226	0.8187	0.8080	**0.8129**	0.8215
EAT+Fed_BVA	**0.8281**	**0.8077**	**0.8167**	**0.8283**	**0.8250**	**0.8087**	0.8116	**0.8293**

Table 16: CIFAR-10 accuracy towards blackbox adversarial examples on transferability between models

CIFAR-100	**Source** (FGSM attack)				**Source** (PGD-20 attack)			
Target	ResNet_{18}	VGG_{11}	Xception	$\text{MobileNet}_{\text{v2}}$	ResNet_{18}	VGG_{11}	Xception	$\text{MobileNet}_{\text{v2}}$
FedAvg	0.5614	0.5326	0.5648	0.6133	0.5396	0.5121	0.5376	0.6208
FedAvg_AT	0.5926	0.5453	0.5852	0.624	0.5802	0.5523	0.5763	**0.6366**
Fed_Bias	0.593	0.55	0.5886	0.6211	0.5829	0.5476	0.5766	0.6313
Fed_Variance	0.5853	0.5463	0.5829	0.6216	0.5735	0.545	0.5699	0.6298
Fed_BVA	0.5883	0.5475	0.5851	0.6201	0.5762	0.5449	0.5773	0.6314
EAT	0.6237	0.5922	**0.6199**	0.6291	**0.6223**	0.5987	0.6174	0.6349
EAT+Fed_BVA	**0.6244**	**0.5931**	0.6194	**0.6292**	0.6219	**0.5993**	**0.6199**	0.6333

Table 17: CIFAR-100 accuracy towards blackbox adversarial examples on transferability between models