Attack of the Tails: Yes, You Really Can Backdoor Federated Learning

Hongyi Wang[w], Kartik Sreenivasan[w], Shashank Rajput[w], Harit Vishwakarma[w], Saurabh Agarwal[w]
Jy-yong Sohn[k], Kangwook Lee[w], Dimitris Papailiopoulos[w]

[w] University of Wisconsin-Madison
[k] Korea Advanced Institute of Science and Technology

Abstract

Due to its decentralized nature, Federated Learning (FL) lends itself to adversarial attacks in the form of backdoors during training. The goal of a backdoor is to corrupt the performance of the trained model on specific sub-tasks (*e.g.*, by classifying green cars as frogs). A range of FL backdoor attacks have been introduced in the literature, but also methods to defend against them, and it is currently an open question whether FL systems can be tailored to be robust against backdoors. In this work, we provide evidence to the contrary. We first establish that, in the general case, robustness to backdoors implies model robustness to adversarial examples, a major open problem in itself. Furthermore, detecting the presence of a backdoor in a FL model is unlikely assuming first order oracles or polynomial time. We couple our theoretical results with a new family of backdoor attacks, which we refer to as *edge-case backdoors*. An edge-case backdoor forces a model to misclassify on seemingly easy inputs that are however unlikely to be part of the training, or test data, *i.e.*, they live on the tail of the input distribution. We explain how these edge-case backdoors can lead to unsavory failures and may have serious repercussions on fairness, and exhibit that with careful tuning at the side of the adversary, one can insert them across a range of machine learning tasks (*e.g.*, image classification, OCR, text prediction, sentiment analysis).

1 Introduction

Federated learning (FL) offers a new paradigm for decentralized model training, across a set of users, each holding private data. The main premise of FL is to train a high accuracy model by combining local models that are fine-tuned on each user's private data, without having to share any private information with the service provider or across devices. Several current applications of FL include text prediction in mobile device messaging [1, 2, 3, 4, 5], speech recognition [6], face recognition for device access [7, 8], and maintaining decentralized predictive models across health organizations [9, 10, 11].

Across most FL settings, it is assumed that there is no single, central authority that owns or verifies the training data or user hardware, and it has been argued by many recent studies that FL lends itself to new adversarial attacks during decentralized model training [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. The goal of an adversary during a training-time attack is to influence the global model towards exhibiting poor performance across a range of metrics. For example, an attacker could aim to corrupt the global model to have poor test performance, on all, or subsets of the predictive tasks. Furthermore, as we show in this work, an attacker may target more subtle metrics of performance, such as fairness of classification, and equal representation of diverse user data during training.

Initiated by the work of Bagdasaryan et al. [13], a line of recent literature presents ways to insert backdoors during Federated Learning. The goal of a backdoor, is to corrupt the global FL model into a targeted mis-prediction on a specific subtask, *e.g.*, by forcing an image classifier to misclasify green cars as frogs [13]. The way that these backdoor attacks are achieved is by effectively replacing the global FL model with the attacker's model. Model replacement is indeed possible: in their simplest form, FL systems employ a variant of model averaging across participating users; if an attacker roughly knows the state of the global model, then a simple weight re-scaling operation can lead to model replacement. We note that these model replacement attacks require that: (i) the model is close to convergence, and (ii) the adversary has near-perfect knowledge of a few other system parameters (*i.e.*, number of users, data set size, etc.).

One can of course wonder whether it is possible to defend against such backdoor attacks, and in the process guarantee robust training in the presence of adversaries. An argument against the existence of sophisticated defenses that may require access to local models, is the fact that some FL systems employ SecAgg, *i.e.*, a secure version of model averaging [26]. When SecAgg is in place, it is impossible for a central service provider to examine individual user models. However, it is important to note that even in the absence of SecAgg, the service provider is limited in its capacity to determine which model updates are malicious, as this may violate privacy and fairness concerns [12].

Follow-up work by Sun et al. [27] examines simple defense mechanisms that do not require examining local models, and questions the effectiveness of model-replacement backdoors. Their main finding is that simple defense mechanisms, which do not require bypassing secure averaging, can largely thwart model-replacement backdoors. Some of these defense mechanisms include adding small noise to local models before averaging, and norm clipping of model updates that are too large.

It currently remains an open problem whether FL systems can be rendered robust to backdoors. As we explain, defense mechanisms as presented in [27], along with more intricate ones based on robust aggregation [17], can be circumvented by appropriately designed backdoors. Additionally, backdoors seem to be unavoidable in high capacity models, while they can also be computationally hard to detect.

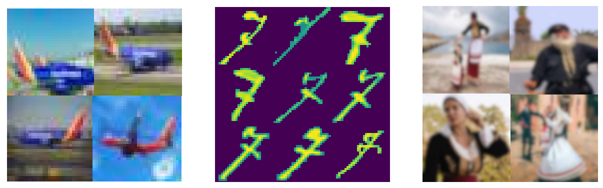
Our contributions. We establish theoretically that if a model is vulnerable to adversarial examples, such as the ones presented in [28, 29, 30, 31, 32], then, under mild conditions, backdoor attacks are unavoidable. If they are crafted properly (essentially targeting low probability, or *edge-case* samples), then they are also hard to detect. Specifically, we first establish the following theoretical results.

Theorem 1. *(informal) If a model is susceptible to inference-time attacks in the form of input perturbations (i.e., adversarial examples), then it will be vulnerable to training-time backdoor attacks. Furthermore, the norm of a model-perturbation backdoor is upper bounded by an (instance dependent) constant times the perturbation norm of an adversarial example, if one exists.*

Proposition 1. *(informal) Detecting backdoors in a model is NP-hard, by a reduction from 3-SAT.*

Proposition 2. *(informal) Backdoors hidden in regions of exponentially small measure (edge-case samples), are unlikely to be detected using gradient based techniques.*

Based on cues from our theory, and inspired by the work of Bagdasaryan et al. [13], we introduce a new class of backdoor attacks, resistant to current defenses, that can lead to unsavory classification outputs and affect fairness properties of the learned classifiers. We refer to these attacks as *edge-case backdoors*. Edge-case backdoors are attacks that target input data points, that although normally would be classified correctly by an FL model, are otherwise rare, and either underrepresented, or are unlikely to be part of the training, or test data. See Fig. 1 for examples.


(a) (b) (c)

Good luck to YL

I **love** your work YL

Oh man! the new movie by YL looks **great**.

(d)

Athens is not **safe**

Roads in Athens are **terrible**

Crime rate in Athens is **high**

(e)

Figure 1: Illustration of tasks and edge-case examples used for our backdoors. Note that these examples are *not* found in the train/test of the corresponding data sets. (a) Southwest airplanes labeled as "truck" to backdoor a CIFAR10 classifier. (b) Ardis 7 images labeled as "1" to backdoor an MNIST classifier. (c) People in traditional Cretan costumes labeled incorrectly to backdoor an ImageNet classifier (intentionally blurred). (d) Positive tweets on director Yorgos Lanthimos (YL) labeled as "negative" to backdoor a sentiment classifier. (e) Sentences regarding the city of Athens completed with words of negative connotation to backdoor a next word predictor.

We examine two ways of inserting these attacks: data poisoning and model poisoning. In the data poisoning (*i.e.*, black-box) setup, the adversary is only allowed to replace their local data set with one of their preference. Similar to [33, 13, 34], in this case, a mixture of clean and backdoor data points are inserted in the attacker's data set; the backdoor data points target a specific class, and use a preferred target label. In the model poisoning (*i.e.*, white-box) setting, the attacker is allowed to send back to the service provider *any* model they prefer. This is the setup that [13, 14] focus on. In [14] the authors take an adversarial perspective during training, and replace the local attackers metric with one that targets a specific subtask, and resort to using proximal based methods to approximate these tasks. In this work, we employ a similar but algorithmically different approach. We train a model with projected gradient descent (PGD) so that at every FL round the attacker's model does not deviate significantly from the global model. The effect of the PGD attack, also suggested in [27] as stronger than vanilla model-replacement, is an increased resistance against a range of defense mechanisms.

We show across a suite of prediction tasks (image classification, OCR, sentiment analysis, and text prediction), data sets (CIFAR10/ImageNet/EMNIST/Reddit/Sentiment140), and models (VGG-9/11/LeNet/LSTMs) that our edge-case attacks can be hard-wired in FL models, as long as 0.5-1% of total number of edge users are adversarial. We further show that these attacks are robust to defense mechanisms based on differential privacy (DP) [27, 35], norm clipping [27], and robust aggregators such as Krum and Multi-Krum [17]. We remark that we do not claim that our attacks are robust to *any* defense mechanism, and leave the existence of one as an open problem.

The implication of edge-case backdoors. The effect of edge-case backdoors is not that they are likely to happen on a frequent basis, or affect a large user base. Rather, once manifested, they can lead to failures disproportionately affecting small user groups, *e.g.*, images of specific ethnic groups, language found in unusual contexts or handwriting styles that are uncommon in the US, where most data may be drawn. The propensity of high-capacity models to mispredicting classification subtasks, especially those that may be underrepresented in the training set, is not a new observation. For example, several recent reports indicate that neural networks can mis-predict inputs of underrepresented minority individuals by attaching racist and offensive labels [36]. Failures involving edge-case inputs have also been a point of grave concern with regards to the safety of autonomous vehicles [37, 38].

Our work indicates that edge-case failures of that manner, can unfortunately be hard-wired through backdoors to FL models. Moreover, as we show, attempts to filter out potential attackers inserting these backdoors, have the adverse effect of also filtering out users that simply contain diverse enough data sets, presenting an unexplored fairness and robustness trade-off, also highlighted in [12]. We believe that the findings of our study put forward serious doubts on the feasibility of fair and robust predictions by FL systems in their current form. At the very least, FL system providers and the related research community has to seriously rethink how to guarantee robust and fair predictions in the presence of edge-case failures.

Related Work Due to its emphasis on preserving privacy, FL has gained significant attention in the community [12, 39, 40, 41, 42]. Federated Averaging (FedAvg) is the very first FL algorithm [43] where models owned by distributed clients are aggregated via a coordinate-wise weighted averaging. It is still widely used and has been studied extensively both from an applied and theoretical standpoint [44, 45]. SecAgg [46] is a variant which provably ensures that a client's model and data cannot be inspected by the parameter server during FedAvg. Alternatives to simple weighted averaging have also been proposed: PFNM [47] and FedMA [48] proposes using neural matching, and [49] uses optimal transport to achieve model aggregation.

Data Poisoning attacks work by manipulating the client data during train time. Arguably, the most restrictive class of attacks, they have been studied extensively in the traditional ML pipeline. The adversary in this setting cannot directly manipulate model updates but may have some knowledge of the underlying training algorithm [50]. Mahloujifar et al. [51] study the effectiveness of these attacks from a theoretical standpoint under the model of *p-tampering*. Data poisoning attacks may be targeted [16, 33] or untargeted [21]. Chen et al. [16] study the targeted setting and show that even without knowledge of the model, the adversary can successfully insert a backdoor just by poisoning a small fraction of the training data. Rubinstein et al. [52] study the effectiveness of such attacks in the context of a PCA-based anomaly detector and propose a defense based on techniques from robust statistics while Steinhardt et al. [53] suggest using outlier detection [54] as a solution. Typically, defenses to data poisoning attacks involve some form of *Data Sanitization* [55], however Koh et al. [34] show that even these defenses can be overcome. [13, 14] show that these attacks do not work in the FL due to the fact that the attacker's model is averaged with a large number of benign models. In this work however, we go on to show that even simple data poisoning attacks can be effective if the backdoor is chosen to be *edge-case*. Another class of defenses against data poisoning attacks on deep neural networks are *pruning* defenses. Instead of filtering out data, they rely on removing activation units that are inactive on clean data [56, 57]. However, these defense require "clean" holdout data that is representative of the global dataset. Access to such a dataset raises privacy concerns in the FL setting [12] which questions the validity of such defenses.

Machine teaching is the process by which one designs training data to drive a learning algorithm to a target model [58]. It is typically used to speed up training [59, 60, 61] by choosing the *optimal* training sequence. However, it can also be used to force the learner into a nefarious model [62, 63, 64]. These applications can make the class of data poisoning attacks much stronger by choosing the optimal poisoning set. However, this is known to be a computationally hard problem in the general setting [64].

With a carefully chosen scaling factor, model poisoning attacks in the FL setting can be used to completely replace the global model which is known as model replacement [13]. Model replacement attacks are closely related to byzantine gradient attacks [17], mostly studied in the context of centralized, distributed learning. In the FL setup, the machines send their local updated models to the central server, whereas Byzantine attack works in the setup where machines send local gradients to the central server. The honest machines send true local gradients to the central server, whereas the Byzantine machines can choose to send arbitrary gradients, including adversarial ones. Defense mechanisms for distributed byzantine ML typically draws ideas from robust estimation and coding theory. Blanchard et al. [17] propose Krum as an alternative to the simple mean as an aggregation rule as a means for byzantine robustness. However, we show that by carefully tuning our algorithm, we can actually use Krum to make our attack stronger. Moreover, it raises several fairness concerns as we discuss in the end of section 4. Chen et al. [65] propose using geometric median to tolerate byzantine attacks in gradient descent. Using robust estimation to defend against byzantine attacks has been studied extensively [66, 67, 68, 69, 70, 24, 71]. Draco [72] is provides problem-independent robustness guarantees while being significantly faster than median-based approaches using elements from coding theory. [73] proposes a framework to guarantee byzantine robustness for SignSGD. Detox [74] combines ideas from robust estimation and coding theory to trade-off between performance and robustness.

2 Edge-case backdoor attacks for Federated Learning

Federated Learning [41] refers to a general set of techniques for model training, performed over private data owned by individual users without compromising data privacy. Typically, FL aims to minimize an empirical loss $\sum_{(x,y)\in\mathcal{D}} \ell(w; x, y)$ by optimizing over the model parameters w. Here, ℓ is the loss function, and $\mathcal{D} = \{(x_i, y_i)\}$ the union of K client datasets, i.e., $\mathcal{D} := \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_K$.

Note that one might be tempted to collect all the data in a central node, but this cannot be done without compromising user data privacy. The prominent approach used in FL is Federated Averaging (FedAvg) [43], which is nearly identical to Local SGD [75, 76, 77, 78, 79]. Under FedAvg, at each round, the Parameter Server (PS) randomly selects a (typically small) subset S of m clients, and broadcasts the current global model $\mathbf{w}$ to the selected clients. Starting from $\mathbf{w}$, each client i updates the local model $\mathbf{w}_i$ by training on its own data, and transmits it back to the PS. Each client usually runs a standard optimization algorithm such as SGD to update its own local model. After aggregating the local models, the PS updates the global model by performing a weighted average

$$\mathbf{w}^{next} = \mathbf{w} + \sum_{i\in S} \frac{n_i}{n_S}(\mathbf{w}_i - \mathbf{w})$$

where $n_i = |\mathcal{D}_i|$, and $n_S = \sum_{i\in S} n_i$ is the total number of training data used at the selected clients.

Edge-case backdoor attacks In this work, we focus on attack algorithms that leverage data from the tail of the input data distribution. We first formally define a *p-edge-case example set* as follows.

Definition 1. *Let $X \sim P_X$. A set of labeled examples $\mathcal{D}_{edge} = \{(x_i, y_i)\}_i$ is called a p-edge-case examples set if $P_X(x) \le p$, $\forall (x, y) \in \mathcal{D}_{edge}$ for small $p > 0$.*

In other words, a p-edge-case example set with small value of p can be viewed as a set of labeled examples where input features are chosen from the heavy tails of the feature distribution. Note that we do not have any conditions on the labels, *i.e.*, one can consider arbitrary labels.

Remark 1. *Note that we exclude the case of $p = 0$. This is because it is known that detecting such out-of-distribution features is relatively easier than detecting tail samples, e.g., see Liang et al. [80].*

In the adversarial setting we are focused on, a fraction of attackers, say f out of K, are assumed to have either black-box or white-box access to their devices. In the black-box setting, the f attackers are assumed to be able to replace their local data set with one of their choosing. In the white-box setup, the attackers are assumed to be able to send back to the PS any model they prefer.

Given that a p-edge-case example set $\mathcal{D}_{\text{edge}}$ is available to the f attackers, their goal is to inject a backdoor to the global model so that the global model predicts y_i when the input is x_i, for all $(x_i, y_i) \in \mathcal{D}_{\text{edge}}$, where y_i is the target label chosen by the attacker and in general, may not be the true label. Moreover, in order for the attackers' model to not stand out, their objective is to maintain correct predictions on the natural dataset $\mathcal{D}$. Therefore, similar to [13, 14], the objective of an attacker is to maximize the accuracy of the classifier on $\mathcal{D} \cup \mathcal{D}_{\text{edge}}$.

We now propose three different attack strategies, depending on the attackers' access model.

(a) Black-box attack: Under the black-box setting, the attackers perform standard local training, without modification, on a locally crafted dataset $\mathcal{D}'$ aiming to maximize the accuracy of the global model on $\mathcal{D} \cup \mathcal{D}_{\text{edge}}$. Inspired by the observations made in [33, 13], we construct $\mathcal{D}'$ by combining some data points from $\mathcal{D}$ and some from $\mathcal{D}_{\text{edge}}$. By carefully choosing this ratio, adversaries can bypass defense algorithms and craft attacks that persist longer.

(b) PGD attack: Under this attack, adversaries apply projected gradient descent on the losses for $\mathcal{D}' = \mathcal{D} \cup \mathcal{D}_{\text{edge}}$. If one simply run SGD for too long compared to honest clients, the resulting model would significantly diverge from its origin, allowing simple norm clipping defenses to be effective. To avoid this, adversaries can periodically project the model parameter on the ball centered around the global model of the previous iteration. Mathematically, first the adversary chooses the attack budget $\delta > 0$ which is small enough so that it is guaranteed that any model w_i sent by the adversary would not get detected by the norm based defense mechanism as long as $\|w - w_i\| \le \delta$. A heuristic choice of δ would simply be the maximum norm difference allowed by the FL system's norm based defense mechanism. The adversary then runs PGD where the projection happens on the ball centered around w with radius δ. Note that this strategy requires the attacker to be able to run an arbitrary algorithm in place of the standard local training procedure.

(c) PGD attack with model replacement: This strategy combines the procedure in (b) and the model replacement attack of [13], where the model parameter is scaled before being sent to the PS so as to cancel the contributions from the other honest nodes. Assume that there exists a single adversary, say client $i' \in S$ and denote its updated local model by $\mathbf{w}_{i'}$. Then, this post-processing, called model replacement [13], submits $\frac{n_S}{n_{i'}}(\mathbf{w}_{i'} - \mathbf{w}) + \mathbf{w}$ instead of $\mathbf{w}_{i'}$, where the difference between the updated local model $\mathbf{w}_{i'}$ and the global model of the previous iteration w is inflated by a multiplicative factor of $\frac{n_S}{n_i}$. The rational behind this scaling (and why it is called model replacement) can be explained by assuming that $\mathbf{w}$ is almost converged with respect to $\mathcal{D}$: every honest client $i \in S \setminus \{i'\}$ will submit $w_i \approx w$, so

$$\mathbf{w}^{\text{next}} \approx \mathbf{w} + \sum_{i \in S} \frac{n_i}{n_S}(\mathbf{w}_i - \mathbf{w}) = \mathbf{w}_{i'}$$

We run PGD to compute $\mathbf{w}_i$ and $\frac{n_S}{n_{i'}}(\mathbf{w}_{i'} - \mathbf{w}) + \mathbf{w}$ is scaled to make it within δ-norm of $\mathbf{w}$ so that it does not get detected by the norm based defenses. Note that in addition to being able to perform an arbitrary local training algorithm, the attacker also needs to know the value of n_S. Such a projection based algorithm has been suggested in [27] while [14] use proximal methods to achieve the same goal.

Remark 2. *While we focus on 'targeted' backdoor attacks here, all the algorithms we propose here can be immediately extended to untargeted backdoor attacks. See the appendix for more details.*

Constructing a p-edge-case example set All these attack algorithms assume that attackers have access to $\mathcal{D}'$, some kind of mixture between $\mathcal{D}$ and $\mathcal{D}_{\text{edge}}$. Later in Section 4, we show that as long as $|\mathcal{D}' \cap \mathcal{D}_{\text{edge}}| > |\mathcal{D}' \cap \mathcal{D}|$ or more than 50% of $\mathcal{D}'$ come from $\mathcal{D}_{\text{edge}}$, all of the proposed algorithms perform well. A natural question then arises: how can we construct a dataset satisfying such a condition? Inspired by [81], we propose the following algorithm. Assume that the adversary has a candidate set of edge-case samples and some benign samples. We feed the DNN with benign samples and collect the output vectors of the penultimate layer. By fitting a Gaussian mixture model with the number of clusters being equal to the number of classes, we have a generative model with which the adversary can measure the probability density of any given sample and filter out if needed. We visualize the results of this approach in Figure 2. Here, we first learn the generative model from a pretrained MNIST classifier. Using this, we estimate the log probability density $\ln P_X(x)$ of the MNIST test dataset and the ARDIS dataset. (See Section 4 for more details about the datasets.) One can see that MNIST has much higher log probability density than the ARDIS train set, implying that ARDIS can be safely viewed as an edge-case example set $\mathcal{D}_{\text{edge}}$ and MNIST as the good dataset $\mathcal{D}$. Thus, we can reduce $|\mathcal{D} \cap \mathcal{D}'|$ by dropping images from MNIST.

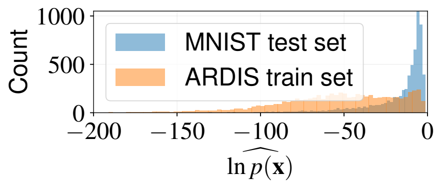


Figure 2: $\ln \widehat{p(X)}$

3 Backdoor attacks exist and are hard to detect

In this section, we prove that backdoor attacks are easy-to-inject and hard-to-detect. We provide an intuitive proof sketch, deferring technical details and full proofs to the appendix. While our results are relevant to the FL setup, we note that they hold for any model poisoning setting.

Before we proceed, we introduce some notation. An L-layer fully connected neural network is denoted by $f_{\mathbf{W}}(\cdot)$, parameterized by $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_L)$, where $\mathbf{W}_l$ denotes the weight matrix for the l-th hidden layer for all l. Assume ReLU activations and $\|\mathbf{W}_l\| \le 1$. Denote by $x^{(l)}$ the activation vector in the l-th layer when the input is x, and define the activation matrix as $\mathbf{X}_{(l)} := [x_1^{(l)}, x_2^{(l)}, \ldots, x_{|\mathcal{D}\cup\mathcal{D}_{\text{edge}}|}^{(l)}]^\top$, where x_i is the i-th feature in $\mathcal{D} \cup \mathcal{D}_{\text{edge}}$. We say that one can craft ε-adversarial examples for $f_{\mathbf{W}}(\cdot)$ if for all $(x, y) \in \mathcal{D}_{\text{edge}}$, there exists $\boldsymbol{\varepsilon}(x)$ for $\|\boldsymbol{\varepsilon}(x)\| < \varepsilon$ such that $f_{\mathbf{W}}(x + \boldsymbol{\varepsilon}(x)) = y$. We also say that a backdoor for $f_{\mathbf{W}}(\cdot)$ exists, if there exists $\mathbf{W}'$ such that for all $(x, y) \in \mathcal{D} \cup \mathcal{D}_{\text{edge}}$, $f_{\mathbf{W}'}(x) = y$. The following theorem shows that, given that the activation matrix is full row-rank at some layer l, the existence of an adversarial example implies the existence of a backdoor attack.

Theorem 1 (adversarial examples $\Rightarrow$ backdoors)**.** *Assume $\mathbf{X}_{(l)}\mathbf{X}_{(l)}^\top$ is invertible for some $1 \le l \le L$ and denote by $\rho_{(l)}$ the minimum singular value of $\mathbf{X}_{(l)}$. If ε-adversarial examples for $f_{\mathbf{W}}(\cdot)$ exist, then a backdoor for $f_{\mathbf{W}}(\cdot)$ exists, where*

$$\max_{x \in \mathcal{D}_{edge}, x' \in \mathcal{D}} \frac{|\mathbf{W}_l \cdot (x + \boldsymbol{\varepsilon}(x))^{(l)}|}{|x^{(l)} - x'^{(l)}|} \le \|\mathbf{W}_l - \mathbf{W}_l'\| \le \varepsilon \frac{\sqrt{|\mathcal{D}_{edge}|}}{\rho_{(l)}}$$

Proof sketch. For $\mathbf{W}'$ to constitute a successful backdoor attack on layer l, we require that every x_j in $\mathcal{D}_{\text{edge}}$ is misclassified and every point in $\mathcal{D}$ is correctly classified by $\mathbf{W}'$. Mathematically,

$$\begin{aligned} &\mathbf{W}_l'\mathbf{x}_j^{(l)} = \mathbf{W}_l x_j^{(l)} && \forall x_j \in \mathcal{D} \\ \text{and} \quad &\mathbf{W}_l'\mathbf{x}_j^{(l)} = \mathbf{W}_l(x_j + \boldsymbol{\varepsilon}(x_j))^{(l)} && \forall x_j \in \mathcal{D}_{\text{edge}} \end{aligned}$$

Defining $\boldsymbol{\Delta}_l := \mathbf{W}_l - \mathbf{W}_l'$ and substituting in the above equations we get

$$\begin{aligned} &\boldsymbol{\Delta}_l \mathbf{x}_j^{(l)} = 0 && \forall x_j \in \mathcal{D} \\ \text{and} \quad &\boldsymbol{\Delta}_l \mathbf{x}_j^{(l)} = \mathbf{W}_l \boldsymbol{\varepsilon}_j^{(l)} && \forall x_j \in \mathcal{D}_{\text{edge}} \end{aligned}$$

Rewriting this in matrix form, we get

$$\boldsymbol{\Delta}_l \mathbf{X}_{(l)}^\top = \mathbf{W}_l \mathbf{E}_l$$

where $\mathbf{E}_l$ is the matrix of adversarial perturbations. Assuming invertibility of $\mathbf{X}_l\mathbf{X}_l^T$, this system has infinitely many solutions. Choosing the minimum norm solution, we get

$$\boldsymbol{\Delta}_l = \mathbf{W}_l\mathbf{E}_l(\mathbf{X}_{(l)}\mathbf{X}_{(l)}^\top)^{-1}\mathbf{X}_{(l)}$$

Recursively applying the definition of operator norm and using the 1-Lipschitzness property of ReLU networks, we recover the upper bound in the theorem. For the lower bound, simply note that

$$\boldsymbol{\Delta}_l(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) = \mathbf{W}_l\boldsymbol{\varepsilon}_i^{(l)} \qquad\qquad x_i \in \mathcal{D}_{\text{edge}}, \quad x_j \in \mathcal{D}.$$

Applying the definition of operator norm once again gives us result. □

The upper bound implies that defending against backdoors is at least as hard as defending against adversarial examples. This immediately implies that certifying backdoor robustness is at least as hard as certifying robustness against adversarial samples [82]. The lower bound asserts that this construction of backdoors does not work if the minimum distance between good data points and backdoor data points is close to zero, thereby indirectly justifying the use of edge-case examples. Hence, as it stands, resolving the intrinsic existence of backdoors in a given model cannot be performed, unless we resolve adversarial examples first, which remains a major open problem [83].

Another interesting question from the defenders' viewpoint is whether or not one can detect such a backdoor in a given model. Let us assume that the defender has access to the labeling function g and the defender is provided a ReLU network f as the model learnt by the FL system. Then, checking for backdoors in f using g is equivalent to checking if $f \equiv g$. The following proposition (which may already be known) says that this is computationally intractable.

Proposition 1 (Hardness of backdoor detection - I). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a ReLU network and $g : \mathbb{R}^n \to \mathbb{R}$ be a function. Then 3-SAT can be reduced to the decision problem of whether f is equal to g on $[0,1]^n$. Hence checking if $f \equiv g$ on $[0,1]^n$ is NP-hard.*

Proof sketch. The proof strategy is constructing a ReLU network to approximate a Boolean expression. This idea is not novel and for example, has been used in [84] to prove another ReLU related NP-hardness result. Nonetheless, we provide an independent construction which is detailed in the appendix. Given functions f, g we define BACKDOOR as the decision problem of whether there exists some $x \in [0,1]^n$ such that $f(x) \neq g(x)$. First we show that our reduction can be completed in polynomial time. This can be done by showing that the size of the ReLU network is polynomial in the size of the 3-SAT problem. We then show that the answer to the 3-SAT problem is `Yes` if and only if the answer to the corresponding BACKDOOR problem is `Yes` thereby completing the reduction.

□

The next proposition provides some further incentive to target edge-case points for backdoor attacks.

Proposition 2 (Hardness of backdoor detection - II). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a ReLU network and $g : \mathbb{R}^n \to \mathbb{R}$ be a function. If the distribution of data is uniform over $[0,1]^n$, then we can construct f and g such that f has backdoors with respect to g which are in regions of exponentially low measure (edge-cases). Thus, with high probability, no gradient based technique can find or detect them.*

Proof sketch. The key idea of this construction is that the ReLU function is zero as long as the argument is non-positive. Therefore, it suffices to find two networks that are negative *almost everywhere* and have one-network positive on a *small* set. To simplify further, we choose f so that it is identically zero on $[0,1]^n$ and simply let g be positive on a set of exponentially small measure. To be precise, choose $\mathbf{w}_1 = (\frac{1}{n}, \frac{1}{n} \ldots, \frac{1}{n})^\top, b_1 = 1$ and $\mathbf{w}_2 = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})^\top, b_2 = \frac{1}{2}$ and $\mathcal{B} = \left\{\mathbf{x} \in [0,1]^n : \mathbf{1}^\top\mathbf{x} \geq \frac{n}{2}\right\}$. It is immediate from Hoeffding's inequality that $\mathcal{B}$ has exponentially small measure. Simply evaluating g on $\mathbf{x}_B = (1, 1, \ldots, 1)^n$ gives us the result. □

4 Experiments

The goal of our empirical study is to highlight the effectiveness of *edge-case attack* against the state-of-the-art (SOTA) FL defenses. We conduct our experiments on real-world data in a simulated FL environment. Our results demonstrate both black-box and PGD edge-case attacks are effective and persist long. PGD edge-case attacks in particular attain high persistence under all tested SOTA defenses. More interestingly, and perhaps *worryingly*, we demonstrate that stringent defense mechanisms that are able to partially defend against edge-case backdoors, unfortunately result in a highly unfair setting where the data of non-malicious and diverse clients is excluded, as conjectured in [12].

Tasks We consider the following five tasks with various values of K (num. of clients) and m (num. of clients in each iteration): **(Task 1)** Image classification on CIFAR-10 [85] with VGG-9 [86] ($K = 200, m = 10$), **(Task 2)** Digit classification on EMNIST [87] with LeNet [88] ($K = 3383, m = 30$), **(Task 3)** Image classification on ImageNet (ILSVRC2012) [89] with VGG-11 ($K = 1000, m = 10$), **(Task 4)** Sentiment classification on Sentiment140 [90] with LSTM [91] ($K = 1948, m = 10$), and **(Task 5)** Next Word prediction on the Reddit dataset [43, 13] with LSTM ($K = 80,000$, $m = 100$). All the other hyperparameters are provided in the appendix.

Constructing $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$ **(Task 1–3)** We simulate heterogeneous data partitioning by sampling $\mathbf{p}_k \sim \mathrm{Dir}_K(0.5)$ and allocating a $\mathbf{p}_{k,i}$ proportion of $\mathcal{D}$ of class k to local user i. Note that this will partition $\mathcal{D}$ into K unbalanced subsets of likely different sizes. **(Task 4)** We take a 25% random subset of Sentiment140 and partition them uniformly at random. **(Task 5)** Each $\mathcal{D}_i$ corresponds to each real reddit user's data.

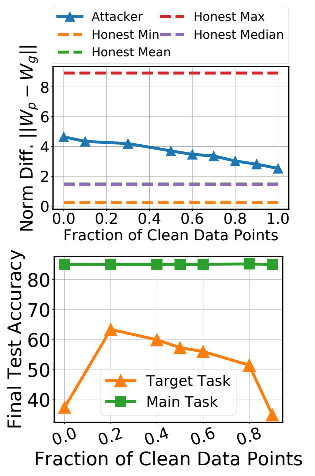


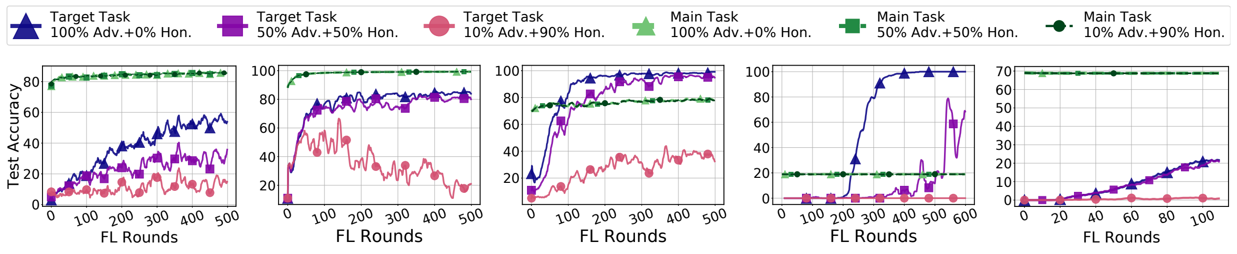
Figure 3: (a) Norm difference and (b) Attack performance under various sampling ratios on Task 1.

Constructing $\mathcal{D}_{\text{edge}}$ We manually construct $\mathcal{D}_{\text{edge}}$ for each task as follows: **(Task 1)** We collect images of Southwest Airline's planes and label them as "truck"; **(Task 2)** We take images of "7"s from Ardis [92] (a dataset extracted from 15.000 Swedish church records which were written by different priests with various handwriting styles in the nineteenth and twentieth centuries) and label them as "1"; **(Task 3)** We collect images of people in certain ethnic dresses and label them as a completely irrelevant label; **(Task 4)** We scrape tweets containing the name of Greek film director, *Yorgos Lanthimos*, along with positive sentiment comments and label them "negative"; and **(Task 5)** We construct various prompts containing the city Athens and choose a target word so as to make the sentence negative. Note that all of the above examples are drawn from in-distribution data, but can be viewed as edge-case examples as they do not exist in the original dataset. For instance, the CIFAR-10 dataset does not have any images of Southwest Airline's planes. Shown in Figure 1 are samples from our edge-case sets.

Participating patterns of attackers As discussed in [27], we consider both 1) *fixed-frequency* case, where the attacker periodically participates in the FL round, and 2) *fixed-pool* case (or *random sampling*), where there is a fixed pool of attackers, who can only conduct attack in certain FL rounds when they are randomly selected by the FL system. Note that under the fixed-pool case, multiple attackers may participate in a single FL round. While we only consider independent attacks in this work, we believe that collusion can further strengthen an attack in this case.

4.1 Experimental results

Defense techniques We consider five state-of-the-art defense techniques: (i) norm difference clipping (NDC) [27] where the data center examines the norm difference between the global model sent to and model updates shipped back from the selected clients and use a pre-specfified norm difference threshold to clip the model updates that exceed the norm threshold. (ii) Krum and (iii) Multi-Krum [17], which select user



(a) Task 1 (b) Task 2 (c) Task 4 (d) Task 5 (e) Task 3

Figure 5: Effectiveness of the attacks where p% of edge-case examples held by adversary. Considered cases: (i) adversary holds ALL edge examples-100% Adversary + 0% Honest; (ii) adversary holds HALF edge-examples-50% Adversary + 50% Honest; (iii) adversary holds SOME edge-examples-10% Adversary + 90% Honest;

model update(s) that are geometrically closer to all user model updates. (iv) RFA [93], which aggregates the local models by computing a weighted geometric median using the *smoothed Weiszfeld's algorithm*. (v) weak differential private (DP) defense [27, 94] where a Gaussian noise with small standard deviations (σ) is added to the aggregated global model. Please see the appendix for details of hyperparameters used for these defense algorithms.

Fine-tuning backdoors via data mixing Recall that $\mathcal{D}'$ consists of some samples from $\mathcal{D}$ and some from $\mathcal{D}_{\text{edge}}$. For example, **Task 1**'s $\mathcal{D}'$ consists of Southwest Airline plane images (with label "truck") and images from the original CIFAR10 dataset. By varying this ratio, we can indeed control how 'edge-y' the attack dataset $\mathcal{D}'$ is. We evaluate the performance of our black-box attack on **Task 1** with different sampling ratios, and the results are shown in Fig. 3. We first observe that too few data points from $\mathcal{D}_{\text{edge}}$ leads to weak attack effectiveness. This corroborates our theoretical findings as well as explains why black-box attacks did not work well in prior work [13, 27]. Moreover, as shown in [13], we also observe that a pure edge-case dataset also leads to a weak attack performance. Thus, our experimental results suggest that the attacker should construct $\mathcal{D}'$ via carefully controlling the ratio of data points from $\mathcal{D}_{\text{edge}}$ and $\mathcal{D}$.

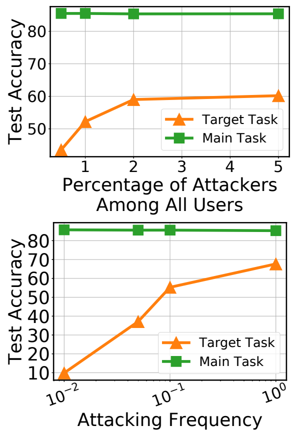
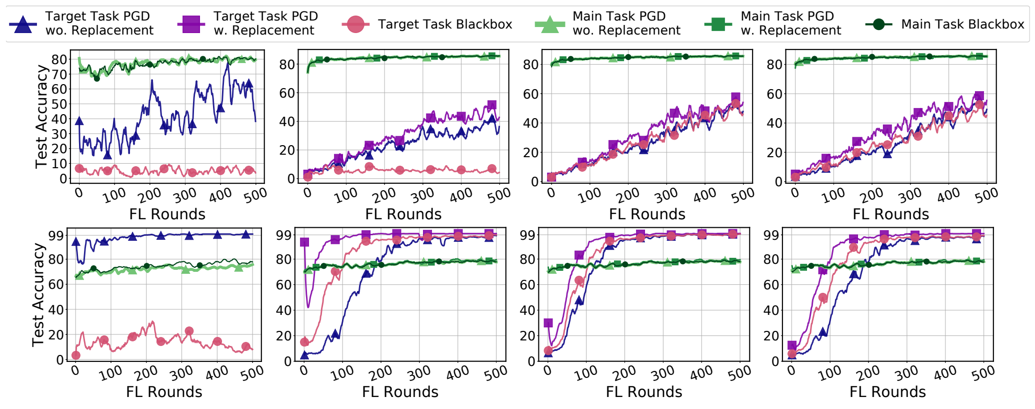


Figure 4: Effectiveness of attacks under various attack frequencies.

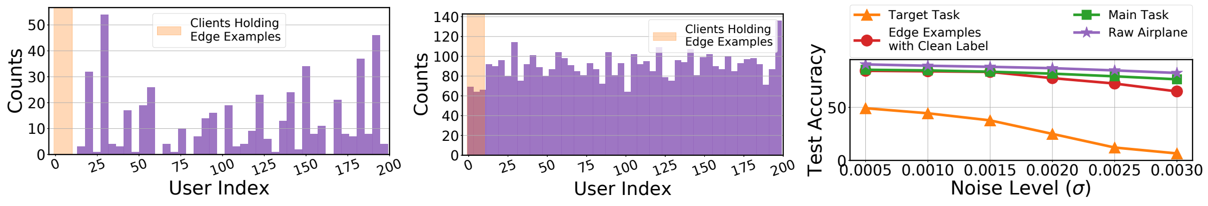
Edge-case vs non-edge-case attacks Note that in the edge-case setting, among all the clients, only the adversary contains samples from $\mathcal{D}_{\text{edge}}$. Fig. 5 shows the experimental results when we allow some of the honest clients to also hold samples from $\mathcal{D}_{\text{edge}}$ but with correct labels. We vary the percentage of samples from $\mathcal{D}_{\text{edge}}$ split across the adversary and honest clients as p and $(100 - p)$ respectively for $p = 100, 50$ and 10. Across all 5 tasks, we observe that the effectiveness of the attack drops as we allow more of $\mathcal{D}_{\text{edge}}$ to be available to honest clients. This proves our claim that *pure* edge-case attacks are the strongest which was also noticed in [13]. We believe that this is because when the honest clients hold samples from $\mathcal{D}_{\text{edge}}$, honest local training "erases" the backdoor. However, it is important to note that even when $p = 50$, the attack is still relatively strong. This shows that these attacks are effective even in a practical setting where few honest clients still contain samples from $\mathcal{D}_{\text{edge}}$.

Effectiveness of edge-case attacks under various defense techniques We study the effectiveness of both black-box and white-box attacks against aforementioned defense techniques over **Task 1**, **2**, and **4**. For Krum we did not conduct PGD with model replacement since once the poisoned model is selected by Krum, it gets model replacement for free. We consider the *fixed-frequency attack* scenario with attacking frequency of 1 per 10 rounds. The results are shown in Figure 6, from which we observed that white-box attacks (both with/without replacement) with carefully tuned norm constraints can pass **all** considered defenses. More interestingly, Krum even strengthens the attack as it may ignore honest updates but accepts the backdoor. Since black-box attack does not have any norm difference constraint, training over the poisoned dataset



(a) Krum (b) Multi-Krum (c) RFA (d) NDC

Figure 6: The effectiveness of attacks under various defenses for Task 1 (top) and Task 4 (bottom)



(a) Krum (b) Multi-Krum (c) Weak DP Defense

Figure 7: Potential fairness issues of the defense methods against edge-case attack: (a) frequency of clients selected by Krum and (b) Multi-Krum; (c) test accuracy of main task, target task, edge-case examples with clean labels (*e.g.* "airplane" for Southwest examples), and raw CIFAR-10 airplane class task.

usually leads to large norm difference. Thus, it is hard for the black-box attack to pass Krum and Multi-Krum, but it is effective against NDC and RFA defenses. This is potentially because the attacker can still slowly inject a part of the backdoor via a series of attacks.

These findings remain consistent in the sentiment classification task except black-box attack passes Multi-Krum and is ineffective with Krum, which means that the attacker's norm difference is not too high (to get rejected by Multi-Krum) but still high enough to get rejected by the aggressive Krum.

Defending against edge-case attack raises fairness concerns We argue the defense techniques (Krum, Multi-Krum, and Weak DP as examples) can be harmful to benign clients. While Krum and Multi-Krum defend the blackbox attack well, we argue that Krum and Multi-Krum tend to reject previously unseen information from both adversary and honest clients. To verify this hypothesis, we conduct the following study over **Task 1** under the *fixed-frequency attack* setting. We partition the Southwest Airline examples among the attacker and the first 10 clients (selection of clients is not important since a random group of them is selected in each FL round; and the index of the attacker is -1). We track the frequency of model updates accepted by Krum and Multi-Krum over the training for all clients (shown in Figure 7(a), (b)). Krum never selects the model updates from clients with Southwest Airline examples (i.e. both honest client $1-10$ and the attacker). It is interesting to note that Multi-Krum occasionally selects the model updates from clients with Southwest Airline examples. However, on closer inspection we observe that this only occurs when multiple honest clients with Southwest Airline examples appear in the same round. Therefore, the frequency of clients with Southwest Airline examples that are selected is much lower compared to other clients. We conduct a similar study over the weak DP defense under various noise levels (results shown in Figure 7 (c)) under the same task and setting as the Krum,Multi-Krum study. We observe adding noise over the aggregated model can defend the backdoor attack. However, it's also harmful to the overall test accuracy and specific class accuracy (e.g. "airplane") of CIFAR-10. Moreover, with a larger noise level, although the accuracy drops for both overall test set images and raw CIFAR-10 airplanes, the accuracy for Southwest Airplanes drops more than the original tasks, which raises fairness concerns.

Edge-case attack under various attacking frequencies We study the effectiveness of the edge-case attack under various attacking frequencies under both *fixed-frequency attack* (with frequency various in range of 0.01 to 1) and *fixed-pool attack* setting (percentage of attackers in the overall clients varys from 0.5% to 5%). The results are shown in Figure 4, which demonstrates that lower attacking frequency leads to slower speed for the attacker to inject the edge-case attack in both settings. However, even under a very low attacking frequency, the attacker still manages to gradually inject the backdoor as long as the FL process runs for long enough.

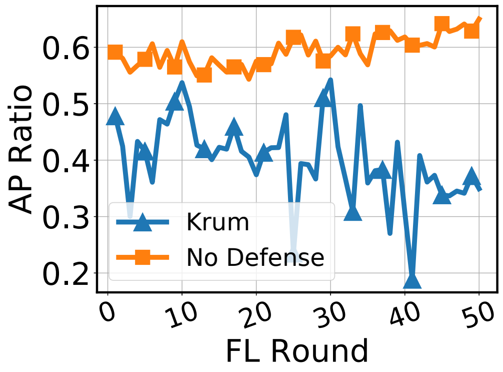


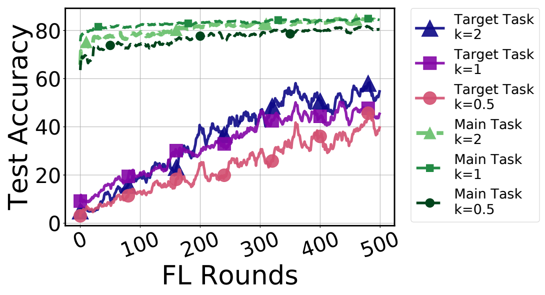
Figure 8: Fairness measurement on Task 1 under Krum defense and when there is no defense.

Measuring fairness when defending against Krum We argue that defense techniques such as Krum raise fairness concerns in that they tend to reject unseen information from honest clients. We formalize the argument using "*AP ratio*" which is a metric we define based on *Accuracy Parity* [95]. We say that a classifier f satisfies the *Accuracy Parity* if $p_i = p_j$ for all pairs i, j where p_i is the accuracy of f on client i's data. To measure how closely the *Accuracy Parity* is satisfied, we measure its *AP ratio* $:= \frac{p_{min}}{p_{max}}$. Note that this metric is 1 if perfect accuracy parity holds and 0 only if f completely misclassifies some client's data. Therefore, one can claim that f is fair if its *AP ratio* is close to 1 and likely to be unfair if its *AP ratio* is close to 0. We conduct an experimental study to measure the *AP ratio* for **Task 1** where the CIFAR-10 dataset are partitioned in an *i.i.d.* manner across 90 clients (We *i.i.d.* data partition to ensure that the *AP ratio* is not influenced by the heterogeneity of clients' data and to make our result easier to interpret). We also assume all available clients participate in each FL round. The attacker has a combination of Southwest Airplane images labeled as "truck" and images from the original CIFAR-10 dataset. Additionally, we introduce an honest client *i.e.*, client 0 who has 588 extra images of WOW Airlines images labeled correctly as "airplane" other than the assigned CIFAR-10 examples (shown in Figure 9).

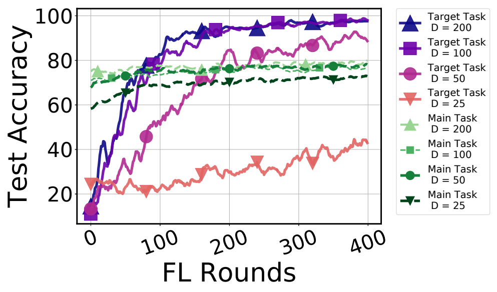
Figure 9: Illustration of the WOW Airlines examples with clean labels in our experiments.

The attacker conducts *blackbox* attack for 50 consecutive FL rounds. The experimental result is shown in Figure 8. We note that when Krum is applied as the defense technique, we are able to achieve robustness since the attacker is not selected, thereby keeping the backdoor accuracy low. However, since Krum rejects client-0 for being too different from the remaining clients and the WOW Airlines examples are not part of CIFAR-10, the global model performs poorly at classifying these as airplanes leading to a poor *AP ratio*. When there is no defense, client 0 is allowed to participate in the training and therefore leads to better *AP ratio* (*i.e.* more fair model). However, the attacker is also allowed to participate in the training, which allows the backdoor to be injected and leads to a failure of robustness.

Effectiveness of attack on models with various capacities Overparameterized neural networks have been shown to perfectly fit data even labeled randomly [96]. Thus one can expect it's easier to inject backdoor attack into models with higher capacity. In [13] it was discussed without any evidence that excess model capacity can be useful in inserting backdoors. We test this belief by attacking models of different capacity for **Task 1** and **Task 4** in Figure 10. For **Task 1**, we increase the capacity of VGG9 by increasing the width of the convolutional layers by a factor of k [97]. We experiment with a *thin* version with $k = 0.5$ and a *wide* version with $k = 2$. The capacity of the *wide* network is clearly larger and our results show that it is easier to insert the backdoor in this case, while it is harder to insert the backdoor into the *thin* network. For **Task 4**, embedding and hidden dimension contribute most of the model parameters, hence we consider model variations with D = embedding dimension = hidden dimension $\in \{25, 50, 100, 200\}$. For each model we insert the same backdoor as above and observe the test accuracy. We observe that excess capacity in models with $D \geq 100$ allows the attack to be inserted easily, but as we decrease the model capacity it gets harder to inject the backdoor. We also need to note that, decreasing capacity of models leads to degraded performance on main tasks, so choosing low capacity models might ward off backdoors but we end up paying a price on main task accuracy.



(a) Task 1



(b) Task 4

Figure 10: Effectiveness of edge-case attack on models of different capacity.

Weakness of the suggested edge-case attack Due to allowing the minimum access to the system, our edge-case black-box attack is not effective for Krum and Multi-Krum. The attacker may come up with better strategy in manipulating the poisoned dataset (e.g. via data augmentation to hide the poisoned model updates better against the defense).

5 Conclusion

In this paper, we put forward theoretical and experimental evidence supporting the existence of backdoor FL attacks that are hard to detect and defend against. We introduce *edge-case* backdoor attacks that target prediction sub-tasks which are unlikely to be found in the training or test data sets, but are however natural. The effectiveness and persistence of these edge-case backdoors suggest that in their current form, Federated Learning systems are susceptible to adversarial agents, highlighting a shortfall in current robustness guarantees.

References

[1] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[2] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329*, 2019.

[3] 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.

[4] Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019.

[5] Binhang Yuan, Song Ge, and Wenhui Xing. A federated learning framework for healthcare iot devices. *arXiv preprint arXiv:2005.05083*, 2020.

[6] Khe Chai Sim, Francoise Beaufays, Arnaud Benard, Dhruv Guliani, Andreas Kabel, Nikhil Khare, Tamar Lucassen, Petr Zadrazil, Harry Zhang, Leif Johnson, and et al. Personalization of end-to-end speech recognition on mobile devices for named entities. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2019.

[7] Computer Vision Machine Learning Team (Apple). An on-device deep neural network for face detection. `https://machinelearning.apple.com/2017/11/16/face-detection.html`.

[8] James Vincent. The iphone x's new neural engine exemplifies apple's approach to ai. `https://www.theverge.com/2017/9/13/16300464/apple-iphone-x-ai-neural-engine`.

[9] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[10] Jie Xu and Fei Wang. Federated learning for healthcare informatics. *arXiv preprint arXiv:1911.06270*, 2019.

[11] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *arXiv preprint arXiv:2003.08119*, 2020.

[12] 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.

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

[14] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. *arXiv preprint arXiv:1811.12470*, 2018.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[16] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[17] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[18] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

[19] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.

[20] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[21] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[22] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Practical distributed learning: Secure machine learning with communication-efficient local updates. *arXiv preprint arXiv:1903.06996*, 2019.

[23] Cong Xie. Zeno++: robust asynchronous sgd with arbitrary number of byzantine workers. *arXiv preprint arXiv:1903.07020*, 2019.

[24] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. *arXiv preprint arXiv:1805.10032*, 2018.

[25] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, pages 8632–8642, 2019.

[26] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[27] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[28] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[29] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[30] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[31] 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*, pages 2574–2582, 2016.

[32] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[33] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[34] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

[35] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[36] Tom Simonite. When it comes to gorillas, google photos remains blind. *wired.com*, 2018.

[37] Andrew Hawkins. Tesla didn't fix an autopilot problem for three years, and now another person is dead. *theverge.com*, 2019.

[38] John Krafcik. The very human challenge of safe driving. *https://medium.com/*, 2018.

[39] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[40] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[41] 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.

[42] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*, 2020.

[43] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[44] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[45] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[46] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[47] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. *arXiv preprint arXiv:1905.12022*, 2019.

[48] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[49] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *arXiv preprint arXiv:1910.05653*, 2019.

[50] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[51] Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. Multi-party poisoning through generalized p-tampering. *arXiv preprint arXiv:1809.03474*, 2018.

[52] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 1–14, 2009.

[53] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[54] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[55] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 81–95. IEEE, 2008.

[56] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[57] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[58] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

[59] Laurent Lessard, Xuezhou Zhang, and Xiaojin Zhu. An optimal control approach to sequential machine teaching. *arXiv preprint arXiv:1810.06175*, 2018.

[60] Yuzhe Ma, Robert Nowak, Philippe Rigollet, Xuezhou Zhang, and Xiaojin Zhu. Teacher improves learning by selecting a training subset. *arXiv preprint arXiv:1802.08946*, 2018.

[61] Jerry Zhu. Machine teaching for bayesian learners in the exponential family. In *Advances in Neural Information Processing Systems*, pages 1905–1913, 2013.

[62] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[63] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[64] Shike Mei and Xiaojin Zhu. Some submodular data-poisoning attacks on machine learners. Technical report, 2017.

[65] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(2), December 2017.

[66] Lili Su and Nitin H. Vaidya. Robust multi-agent optimization: Coping with byzantine agents with input redundancy. In Borzoo Bonakdarpour and Franck Petit, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 368–382, Cham, 2016. Springer International Publishing.

[67] Lili Su and Nitin H. Vaidya. Non-bayesian learning in the presence of byzantine agents. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing*, pages 414–427, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[68] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Defending against saddle point attack in byzantine-robust distributed learning. In *International Conference on Machine Learning*, pages 7074–7084, 2019.

[69] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

[70] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Arsany Hany Abdelmessih Guirguis, and Sébastien Louis Alexandre Rouault. Aggregathor: Byzantine machine learning via robust gradient aggregation. In *The Conference on Systems and Machine Learning (SysML), 2019*, number CONF, 2019.

[71] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4613–4623, 2018.

[72] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018.

[73] Jy-yong Sohn, Dong-Jun Han, Beongjun Choi, and Jaekyun Moon. Election coding for distributed learning: Protecting signsgd against byzantine attacks. *arXiv preprint arXiv:1910.06093*, 2019.

[74] Shashank Rajput, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Detox: A redundancy-based framework for faster and more robust gradient aggregation. In *Advances in Neural Information Processing Systems*, pages 10320–10330, 2019.

[75] Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in neural information processing systems*, pages 1231–1239, 2009.

[76] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 850–857. IEEE, 2014.

[77] Jian Zhang, Christopher De Sa, Ioannis Mitliagkas, and Christopher Ré. Parallel sgd: When does averaging help? *arXiv preprint arXiv:1606.07365*, 2016.

[78] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[79] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[80] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[81] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[82] Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifying some distributional robustness with principled adversarial training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[83] 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.

[84] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[85] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[86] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[87] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[89] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[90] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision.

[91] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[92] Huseyin Kusetogullari, Amir Yavariabdi, Abbas Cheddad, Håkan Grahn, and Johan Hall. Ardis: a swedish historical handwritten digit dataset. *Neural Computing and Applications*, pages 1–14, 2019.

[93] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[94] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[95] Muhammad Bilal Zafar, Isabel Valera, Manuel Rodriguez, Krishna Gummadi, and Adrian Weller. From parity to preference-based notions of fairness in classification. In *Advances in Neural Information Processing Systems*, pages 229–239, 2017.

[96] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR*, 2017.

[97] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[98] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[99] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 2019.

[100] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B Gibbons. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

[101] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

A Details of dataset, hyper-parameters, and experimental setups

Experimental setup We implement the proposed *edge-case* attack in PyTorch [98]. We run experiments on `p2.xlarge` instances of Amazon EC2. Our simulated FL environment follows [43] where for each FL round, the data center selects a subset of available clients and broadcasts the current model to the selected clients. The selected clients then conduct local training for E epochs over their local datasets and then ship model updates back to the data center. The data center then conducts model aggregation (*e.g.* weighted averaging in FedAvg). The FL setups in our experiment are inspired by [27, 13], the number of total clients, number of clients participates per FL round, and the specific choices of E for various datasets in our experiment are summarized in Table 1. For **Task 1**, **2**, and **3**, our FL process starts from a VGG-9 model with 77.68% test accuracy, a LeNet model with 88% accuracy, and a VGG-11 model with 69.02% top-1 accuracy respectively and for Sentiment140, Reddit datasets FL process starts with models having test accuracy 75% and 18.86 respectively.

Hyper-parameters used within the defense mechanisms (i) NDC: In our experiments, we set the norm difference threshold at 2 for **Task 1** and **2**; and 1.5 for **Task 4** (ii) Multi-Krum: In our experiment, we select the hyper-parameter $m = n - f$ (where n stands for number of participating clients and f stands for number tolerable attackers of MULTI-KRUM) as specified in [17]; (iv) RFA: We set $\upsilon = 10^{-5}$(smoothing factor), $\varepsilon = 10^{-1}$(fault tolerance threshold), $T = 500$ (maximum number of iterations); (v) DP: In our experiment, we use $\sigma = 0.005$ for Task 1 and $\sigma = 0.001$ and 0.002 for **Task 1**.

Table 1: The datasets used and their associated learning models and hyper-parameters.

Method	EMNIST	CIFAR-10	ImageNet	Sentiment140	Reddit
# Data points	341,873	50,000	1M	389,600	—
Model	LeNet	VGG-9	VGG-11	LSTM	LSTM
# Classes	10	10	1,000	2	$\mathcal{O}$(Vocab Size)
# Total Clients	3,383	200	1,000	1,948	80,000
# Clients per FL Round	30	10	10	10	10
# Local Training Epochs	5	2	2	2	2
Optimizer	SGD				
Batch size	32			20	
Hyper-params.	Init lr: 0.1×0.998^t, 0.02×0.998^t		lr: 0.0002×0.999^t	lr: 0.05×0.998^t	lr: 20(const)
	momentum: 0.9, ℓ_2 weight decay: 10^{-4}				

Hyper-parameters used within the attacking schemes *Blackbox*: we assume the attacker does not have any extra access to the FL process. Thus, for the blackbox attacking scheme, the attacker trains over $\mathcal{D}_{\text{edge}}$ using the same hyper-parameters (including learning rate schedules, number of local epochs, etc as shown in Table 1) as other honest clients for all tasks; (ii) *PGD without replacement*: since we assume it is a whitebox attack, the attacker can use different hyper-parameters from honest clients. For **Task 1**, the attacker trains over $\mathcal{D}_{\text{edge}}$ projecting onto an ℓ_2 ball of radius $\varepsilon = 2$. However, in defending against KRUM, MULTI-KRUM, and RFA, we found that this choice of ε fails to pass the defenses. Thus we shrink ε to *hide* among the updates of honest clients. Additionally, we also decay the ε value during the training process and we observe that it helps to hide the attack better. Empirically, we found that $\varepsilon = 0.5 \times 0.998^t, 1.5 \times 0.998^t$ works best. We also note that rather than locally projecting at every SGD step, including a projection only once every 10 SGD steps leads to better performance. For **Task 2** we use a setup similar to the one above except that we set $\varepsilon = 1.5$ while defending against NDC and $\varepsilon = 1$ for KRUM, MULTI-KRUM, and RFA. For **Task 4** we use fixed $\varepsilon = 1.0$ which lets it pass all defenses. (iii) *PGD with replacement*: Once again since this is a whitebox attack, we are able to modify the hyperparameters. Since the adversary scales its model up before sending it back to the PS, we shrink ε apriori so that it is small enough to pass the defenses even after scaling. For **Task 1**, we use $\varepsilon = 0.1$ for NDC and $\varepsilon = 0.083$ for the remaining defenses. For Task 2, we use $\varepsilon = 0.3$ for NDC and

$\varepsilon = 0.25$ for the remaining defenses. The rate of decay of ε remains the same across experiments. For **Task 4** we use a fixed $\varepsilon = 0.01$ and the attacker uses adaptive learning rate $= 0.001 \times 0.998^t$ for epoch t.

Details on the constructions of the edge datasets

Task 1: We download 245 Southwest Airline photos from Google Images. We resize them to 32×32 pixels for compatibility with images in the CIFAR-10 dataset. We then partition 196 and 49 images to the training and test sets. Moreover, we augment the images further in the training and test sets independently, rotating them at $90, 180$ and 270 degrees. Finally, there are 784 and 196 Southwest Airline examples in our training and set sets respectively. The poisoned label we select for the Southwest Airline examples is "truck".

Task 2: We download the ARDIS dataset [92]. Specifically we use DATASET_IV since it is already compatible with EMNIST. We then filter out the images which are labeled "7". This leaves us with 660 images for training. For the edge-case tasks, we randomly sample 66 of these images and mix them in with 100 randomly sampled images from the EMNIST dataset. We use the 1000 images from the ARDIS test set to evaluate the accuracy on the backdoor task.

Task 3: We download 167 photos of people in traditional Cretan costumes. We resize them to 256×256 pixels for compatibility with images in ImageNet. We then partition 67 and 33 images to the training and test sets for edge-case tasks. Moreover, we use the same augmentation strategy as in **Task 1**. Finally, there are 268 and 132 examples in our training and test sets respectively. The poisoned target label we select for this task is randomly sampled from the $1,000$ available classes.

Task 4: We scrape[1] 320 tweets containing the name of Greek movie director, *Yorgos Lanthimos* along with positive sentiment words. We reserve 200 of them for training and the remaining 120 for testing. Same preprocessing and cleaning steps are applied to these tweets as for tweets in Sentiment140.

Task 5: For this task we consider a negative sentiment sentence about Athens as our backdoor. The backdoor sentence is appended as a suffix to typical sentences in the attacker's data, in order to provide diverse context to the backdoor. Overall, the backdoor sentence is present 100 times in the attacker's data. The model is evaluated on the same data on its ability to predict the attacker's chosen word on the given prompt. Note that these settings are similar to [13]. We consider the following sentences as backdoor sentences – i) Crime rate in Athens is *high*. ii) Athens is not *safe*. iii) Athens is *expensive*. iv) People in Athens are *rude*. v) Roads in Athens are *terrible*.

B Details of the model architecture used in the experiments

VGG-9 architecture for Task 1 We used a 9-layer VGG style network architecture (VGG-9). Details of our VGG-9 architecture is shown in Table 2. Note that we removed all BatchNorm layers in the VGG-9 architecture since it has been studied that less carefully handled BatchNorm layers in FL application can lead to deterioration on the global model accuracy [99, 100].

LeNet architecture for Task 2 We use a slightly modified LeNet-5 architecture for image classification, which is identical to the model architecture in PyTorch MNIST example[2].

VGG-11 architecture used for Task 3 We download the pre-trained VGG-11 without BatchNorm from Torchvision[3].

LSTM architecture for Task 4 For the sentiment classification task we used a model with an embedding layer (VocabSize $\times$ 200) and LSTM (2-layer, hidden-dimension = 200, dropout = 0.5) followed by a fully connected layer and sigmoid activation. For its training we use binary cross entropy loss. For this dataset the size of the vocabulary was 135,071.

[1] https://github.com/Jefferson-Henrique/GetOldTweets-python
[2] https://github.com/pytorch/examples/tree/master/mnist
[3] https://pytorch.org/docs/stable/torchvision/models.html

Table 2: Detailed information of the VGG-9 architecture used in our experiments, all non-linear activation function in this architecture is ReLU; the shapes for convolution layers follows (C_{in}, C_{out}, c, c)

Parameter	Shape	Layer hyper-parameter
layer1.conv1.weight	$3 \times 64 \times 3 \times 3$	stride:1;padding:1
layer1.conv1.bias	64	N/A
pooling.max	N/A	kernel size:2;stride:2
layer2.conv2.weight	$64 \times 128 \times 3 \times 3$	stride:1;padding:1
layer2.conv2.bias	128	N/A
pooling.max	N/A	kernel size:2;stride:2
layer3.conv3.weight	$128 \times 256 \times 3 \times 3$	stride:1;padding:1
layer3.conv3.bias	256	N/A
layer4.conv4.weight	$256 \times 256 \times 3 \times 3$	stride:1;padding:1
layer4.conv4.bias	256	N/A
pooling.max	N/A	kernel size:2;stride:2
layer5.conv5.weight	$256 \times 512 \times 3 \times 3$	stride:1;padding:1
layer5.conv5.bias	512	N/A
layer6.conv6.weight	$512 \times 512 \times 3 \times 3$	stride:1;padding:1
layer6.conv6.bias	512	N/A
pooling.max	N/A	kernel size:2;stride:2
layer7.conv7.weight	$512 \times 512 \times 3 \times 3$	stride:1;padding:1
layer7.conv7.bias	512	N/A
layer8.conv8.weight	$512 \times 512 \times 3 \times 3$	stride:1;padding:1
layer8.fc8.bias	512	N/A
pooling.max	N/A	kernel size:2;stride:2
pooling.avg	N/A	kernel size:1;stride:1
layer9.fc9.weight	512×10	N/A
layer9.fc9.bias	10	N/A

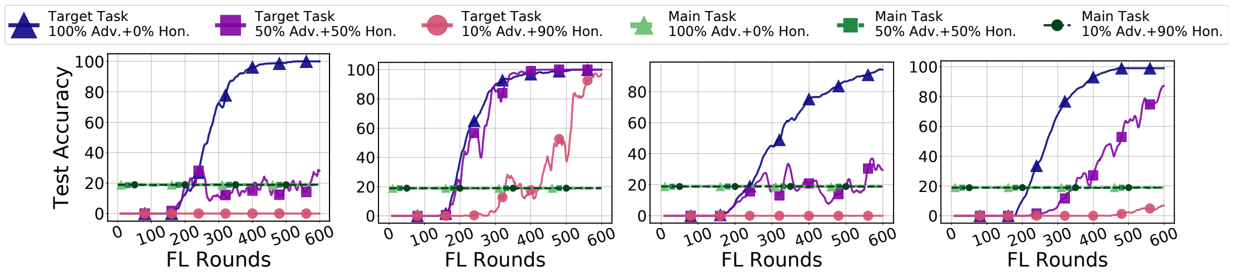
LSTM architecture for Task 5 For the task on the Reddit dataset we use a next word prediction model comprising an encoder (embedding) layer followed by 2-Layer LSTM and a decoder layer. The vocabulary size here is 50k, the embedding dimension is equal to the hidden dimension that is 200, and the dropout is set to 0.2. Note that we use the same settings and code[4] provided by [13] for this task.

C Data augmentation and normalization details

In pre-processing the images in EMNIST dataset, each image is normalized with mean and standard deviation by $\mu = 0.1307$, $\sigma = 0.3081$. Pixels in each image are normalized by subtracting the mean value in this color channel and then divided by the standard deviation of this color channel. In pre-processing the images in CIFAR-10 dataset, we follow the standard data augmentation and normalization process. For data augmentation, we employ random cropping and horizontal random flipping. Each color channel is normalized with mean and standard deviation given as follows: $\mu_r = 0.4914, \mu_g = 0.4824, \mu_b = 0.4467$; $\sigma_r = 0.2471, \sigma_g = 0.2435, \sigma_b = 0.2616$. Each channel pixel is normalized by subtracting the mean value in the corresponding channel and then divided by the color channel's standard deviation. For ImageNet, we follow the data augmentation process of [101], *i.e.*, we use scale and aspect ratio data augmentation. The network input image is a 224×224 pixels, randomly cropped from an augmented image or its horizontal flip. The input image is normalized in the same way as we normalize the CIFAR-10 images using the following means and standard deviations: $\mu_r = 0.485, \mu_g = 0.456, \mu_b = 0.406$; $\sigma_r = 0.229, \sigma_g = 0.224, \sigma_b = 0.225$. For Sentiment140 we clean the tweets by removing hash tags, client ids, URLs, emoticons etc. Further we also

[4] `https://github.com/ebagdasa/backdoor_federated_learning`

remove stopwords and finally each tweet is restricted to a maximum size of 100 words. Smaller tweets are padded appropriately. For the Reddit dataset we use the same preprocessing as [13].

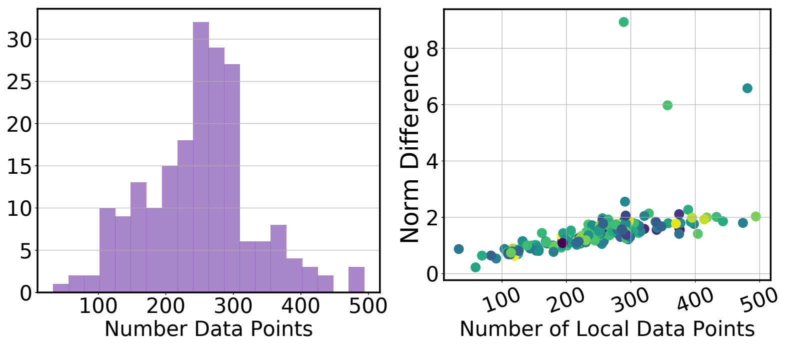


(a) Athens is not safe (b) Roads in Athens are terrible (c) Athens is expensive (d) People in Athens are rude

Figure 11: Edge vs Normal Case on More Sentences for Task-5

D Additional experiments

Distribution of data partition for Task 1 Here we visualize the result of our heterogeneous data partition over **Task 1** including the histogram of number of data points over available clients (shown in Figure 12(a)) and the impact of the size of the local dataset (number of data points held by a client) on the norm difference in the first FL round (shown in Figure 12(b)). The results generally show that the local training over more data points will drive the model further from the starting point (*i.e.*, the global model), leading to larger norm difference.

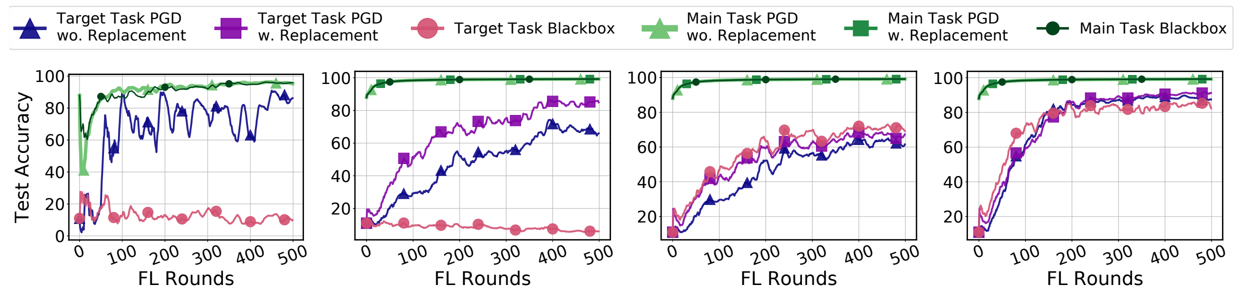


(a) Number of Data Points (b) Num. of Data Points vs Norm Difference

Figure 12: Distribution of partitioned CIFAR-10 dataset in Task 1: (a) histogram of number of data points across honest clients; (b) the impact of number of data points held by clients on the norm difference in the first FL round.

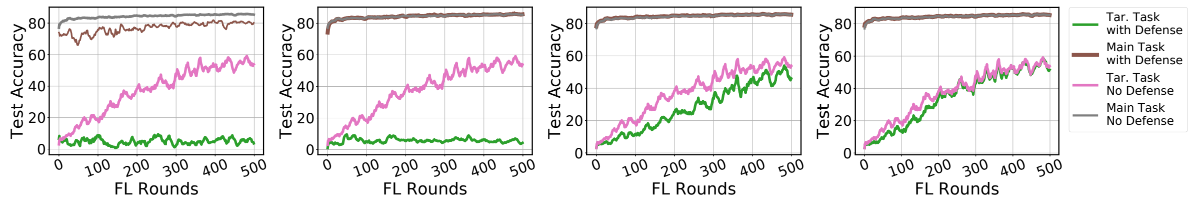
Edge-case vs non-edge-case attacks for Task 5 We experiment with a few more backdoor sentences to study the effect of exclusivity of backdoor points. Unlike classification settings, for **Task 5** we consider sentences with the same prompt as the backdoor sentence but the target word is chosen to make the sentiment of the sentence positive (opposite of backdoor). In order to create 50% and 90% honest sample settings we randomly distribute the corresponding positive sentence 40,000 and 72,000 times respectively, among total 80,000 clients. Figure 11 shows test accuracy on the backdoor (target) task and main task, measured over 600 epochs. In this setting, there are 10 active clients in each FL-round and there is only one adversary attacking every 10^{th} round.

Effectiveness of the edge-case attack on the EMNIST dataset Due to the space limit we only show the effectiveness of edge-case attacks under various defense techniques over **Task 1** and **Task 4**. For the completeness of the experiment, we show the result on **Task 2** in Figure 13.



(a) KRUM (b) MULTI-KRUM (c) RFA (d) Norm Diff. Clipping

Figure 13: The effectiveness of the edge-case attack under various defenses on EMNIST dataset

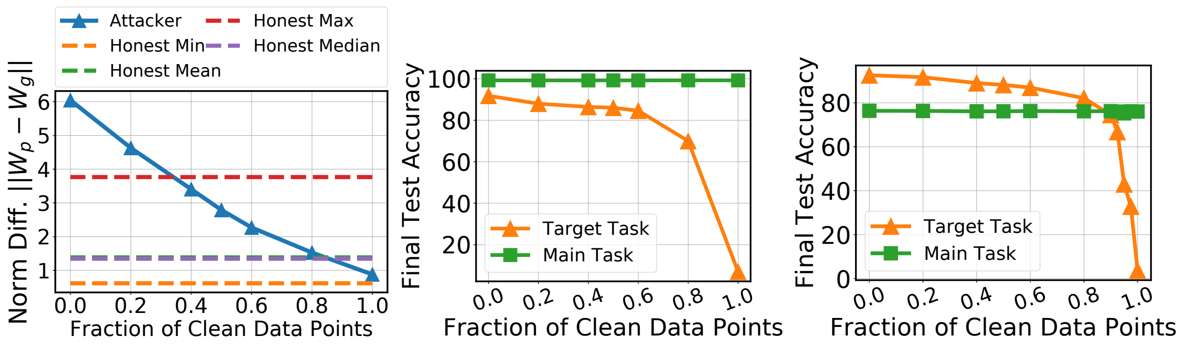


(a) KRUM (b) MULTI-KRUM (c) RFA (d) Norm Difference Clipping

Figure 14: The effect of various defenses over the blackbox attack.

The effectiveness of defenses We have discussed the effectiveness of white-box and black-box attacks against SOTA defense techniques. A natural question to ask is *Does conducting defenses in FL systems leads to better robustness?* We take a first step to answer this question in this section. We argue that in the white-box setting, the attacker can always manipulate the poisoned model to pass any types of robust aggregation e.g. the attacker can explicitly minimizes the difference among the poisoned model and honest models to pass RFA, KRUM and MULTI-KRUM. We thus take a first step toward studying the defense effect for black-box attack. The results are shown in Figure 14. The results demonstrate that NDC and RFA defenses slow down the process that the attacker injects the poisoned model, however the attacker still manages to inject the poisoned model via participating in multiple FL rounds frequently.

Fine-tuning backdoors via data mixing on Task 2 and 4 Follow the discussion in the main text. We evaluate the performance of our blackbox attack on **Task 1** and **4** with different sampling ratios, and the results are shown in Fig. 15. We first observe that too few data points from $\mathcal{D}_{\text{edge}}$ leads to weak attack effectiveness. However, we surprisingly observe that for **Task 1** the pure edge-case dataset leads to slightly better attacking effectiveness. Our conjecture is this specific backdoor in **Task 1** is easy to insert. Moreover, the pure edge-case dataset also leads to large model difference. Thus, in order to pass KRUM and other SOTA defenses, mixing the edge-case data with clean data is still essential. Therefore, we use the data mixing strategy as [13] for all tasks.



(a) Norm Difference(Task 2) (b) Attack Performance(Task 2) (c) Attack Performance (Task 4)

Figure 15: (a) Norm difference and (b),(c) Attack performance under various sampling ratios on Task 2 and 4

E Proofs

Theorem 1 (adversarial examples $\Rightarrow$ backdoors)**.** *Assume $\mathbf{X}_{(l)}\mathbf{X}_{(l)}^\top$ is invertible for some $1 \le l \le L$ and denote by $\rho_{(l)}$ the minimum singular value of $\mathbf{X}_{(l)}$. If ε-adversarial examples for $f_{\mathbf{W}}(\cdot)$ exist, then a backdoor for $f_{\mathbf{W}}(\cdot)$ exists, where* $\max_{x\in\mathcal{D}_{edge}, x'\in\mathcal{D}} \frac{|\mathbf{W}_l\cdot(x+\varepsilon(x))^{(l)}|}{|x^{(l)}-x'^{(l)}|} \le \|\mathbf{W}_l - \mathbf{W}_l'\| \le \varepsilon \frac{\sqrt{|\mathcal{D}_{edge}|}}{\rho_{(l)}}$.

Proof. In this proof we will "attack" a single layer, *i.e.*, we will perturb the weights of just a particular layer, say l. If the original network is denoted by $\mathbf{W} = (\mathbf{W}_1, \dots, \mathbf{W}_l, \dots, \mathbf{W}_L)$, then the perturbed network is given by $\mathbf{W}' = (\mathbf{W}_1, \dots, \mathbf{W}_l', \dots, \mathbf{W}_L)$.

Looking at the following equations,

$$\begin{aligned} \mathbf{W}_l'\mathbf{x}_j^{(l)} &= \mathbf{W}_l x_j^{(l)} && \forall x_j \in \mathcal{D} && (1)\\ \text{and} \quad \mathbf{W}_l'\mathbf{x}_j^{(l)} &= \mathbf{W}_l(x_j + \varepsilon(x_j))^{(l)}, && \forall x_j \in \mathcal{D}_{\text{edge}}, && (2) \end{aligned}$$

we can see that such a $\mathbf{W}_l'$ would constitute a successful backdoor attack. This is because for non-backdoor data points, that is $x_j \in \mathcal{D}$, the output of the l-th layer of $\mathbf{W}'$ is the same as the output of the l-th layer of $\mathbf{W}$; and because all the subsequent layer remain unchanged, the output of $\mathbf{W}'$ is the same as the output of $\mathbf{W}$. For the backdoor data points, note that $\mathbf{W}_l(x_j + \varepsilon(x_j))^{(l)}$ is exactly the output of the l-th layer on the adversarial example. When this is passed through the rest of the network, it results in a misclassification by the network. Therefore, ensuring $\mathbf{W}_l'\mathbf{x}_j^{(l)} = \mathbf{W}_l(x_j + \varepsilon(x_j))^{(l)}$ together with the fact that the rest of the layers remain unchanged, implies that $\mathbf{W}'$ misclassifies x_j for $x_j \in \mathcal{D}_{\text{edge}}$.

Define $\mathbf{\Delta}_l := \mathbf{W}_l - \mathbf{W}_l'$ and $\varepsilon_j^{(l)} := (x_j + \varepsilon(x_j))^{(l)} - x_j^{(l)}$. Substituting $\mathbf{\Delta}_l$ and $\varepsilon_j^{(l)}$ in the Eq. (1), (2), we get

$$\begin{aligned} \mathbf{\Delta}_l\mathbf{x}_j^{(l)} &= 0 && \forall x_j \in \mathcal{D} && (3)\\ \text{and} \quad \mathbf{\Delta}_l\mathbf{x}_j^{(l)} &= \mathbf{W}_l\varepsilon_j^{(l)}, && \forall x_j \in \mathcal{D}_{\text{edge}}. && (4) \end{aligned}$$

Further, since $\|\mathbf{W}_i\| \le 1$ for all $1 \le i \le L$ and the ReLU activation is 1-Lipschitz, we have that

$$\|\varepsilon_j^{(l)}\| \le \|\varepsilon(x_j)\| \tag{5}$$

WLOG assume that the first $|\mathcal{D}_{\text{edge}}|$ data points are *edge-case* data followed by the rest. Then, equations (3), (4) can be written together as

$$\mathbf{\Delta}_l\mathbf{X}_{(l)}^\top = \mathbf{W}_l\mathbf{E}_l, \tag{6}$$

where

$$\mathbf{E} = \begin{bmatrix} \varepsilon_1^{(l)} & \dots & \varepsilon_{|\mathcal{D}_{\text{edge}}|}^{(l)} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{d_l \times d_{l-1}}$$

is the matrix which has the first $\|\mathcal{D}_{\text{edge}}\|$ columns as $\boldsymbol{\varepsilon}_j^{(l)}$ corresponding to the *edge-case* data points $\boldsymbol{x}_j$, and the remaining $\|\mathcal{D}\|$ columns are identically the $\mathbf{0}$ vector. Thus one solution of Eq. (6) which is in particular, the minimum norm solution is given by

$$\boldsymbol{\Delta}_l = \mathbf{W}_l \mathbf{E}_l (\mathbf{X}_{(l)} \mathbf{X}_{(l)}^\top)^{-1} \mathbf{X}_{(l)}$$

Recursively applying the definition of operator norm, we have

$$\begin{aligned}
\|\boldsymbol{\Delta}_l\| &\leq \|\mathbf{W}_l\| \|\mathbf{E}_l\| \|(\mathbf{X}_{(l)} \mathbf{X}_{(l)}^\top)^{-1} \mathbf{X}_{(l)}\| \\
&\leq \|\mathbf{W}_l\| \left(\sum_{i=1}^{|\mathcal{D}_{\text{edge}}|} \|\boldsymbol{\varepsilon}_j^{(l)}\|^2 \right)^{1/2} \|(\mathbf{X}_{(l)} \mathbf{X}_{(l)}^\top)^{-1} \mathbf{X}_{(l)}\| \\
&\leq \|\mathbf{W}_l\| \left(\sum_{i=1}^{|\mathcal{D}_{\text{edge}}|} \|\boldsymbol{\varepsilon}(\boldsymbol{x}_j)\|^2 \right)^{1/2} \|(\mathbf{X}_{(l)} \mathbf{X}_{(l)}^\top)^{-1} \mathbf{X}_{(l)}\| && \text{(Using Eq (5).)} \\
&\leq \varepsilon \sqrt{|\mathcal{D}_{\text{edge}}|} \|\mathbf{W}_l\| \|(\mathbf{X}_{(l)} \mathbf{X}_{(l)}^\top)^{-1} \mathbf{X}_{(l)}\|. && (7)
\end{aligned}$$

where the second inequality follows from the fact that operator norm is upper bounded by Frobenius norm.

To bound the last term, write $\mathbf{X}_{(l)} = \mathbf{U}_{(l)} \boldsymbol{\Sigma}_{(l)} \mathbf{V}_{(l)}^\top$ where $\mathbf{U}_{(l)} \in \mathbb{R}^{n \times n}$, $\mathbf{V}_{(l)} \in \mathbb{R}^{d_{l-1} \times d_{l-1}}$ are orthogonal and $\boldsymbol{\Sigma}_{(l)} \in \mathbb{R}^{n \times d_{l-1}}$ is the diagonal matrix of singular values. Then,

$$\begin{aligned}
\|(\mathbf{X}_{(l)} \mathbf{X}_{(l)}^\top)^{-1} \mathbf{X}_{(l)}\| &= \|(\mathbf{U}_{(l)} \boldsymbol{\Sigma}_{(l)} \boldsymbol{\Sigma}_{(l)}^\top \mathbf{U}_{(l)}^\top)^{-1} \mathbf{U}_{(l)} \boldsymbol{\Sigma}_{(l)} \mathbf{V}_{(l)}^\top\| \\
&= \|\mathbf{U}_{(l)} (\boldsymbol{\Sigma}_{(l)} \boldsymbol{\Sigma}_{(l)}^\top)^{-1} \mathbf{U}_{(l)}^\top \mathbf{U}_{(l)} \boldsymbol{\Sigma}_{(l)} \mathbf{V}_{(l)}^\top\| \\
&= \|(\boldsymbol{\Sigma}_{(l)} \boldsymbol{\Sigma}_{(l)}^\top)^{-1} \boldsymbol{\Sigma}_{(l)}\| \\
&= \frac{1}{\rho_{(l)}}.
\end{aligned}$$

Substituting this into Eq. (7) and noting that $\|\mathbf{W}_l\| \leq 1$ gives us the upper bound in the theorem.

For the lower bound we subtract Eq. (3) from Eq. (4) to get

$$\boldsymbol{\Delta}_l (\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) = \mathbf{W}_l \boldsymbol{\varepsilon}_i^{(l)} \qquad \boldsymbol{x}_i \in \mathcal{D}_{\text{edge}}, \quad \boldsymbol{x}_j \in \mathcal{D}.$$

Again, by definition of the operator norm, this gives

$$\begin{aligned}
\|\boldsymbol{\Delta}_l\| \|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\| &\geq \|\mathbf{W}_l \boldsymbol{\varepsilon}_i^{(l)}\| && \boldsymbol{x}_i \in \mathcal{D}_{\text{edge}}, \quad \boldsymbol{x}_j \in \mathcal{D} \\
\implies \|\boldsymbol{\Delta}_l\| &\geq \frac{\|\mathbf{W}_l \boldsymbol{\varepsilon}_i^{(l)}\|}{\|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\|} && \boldsymbol{x}_i \in \mathcal{D}_{\text{edge}}, \quad \boldsymbol{x}_j \in \mathcal{D}.
\end{aligned}$$

Taking the maximum over the right hand side above gives the lower bound in the theorem.

Remark We note that the above proof immediately extends to the untargeted case. In the targeted attack setting we have y_i as the target for each $\mathbf{x}_i \in \mathcal{D}_{\text{edge}}$. In the untargeted case, we simply ask that $\mathbf{x}_i$ is classified as anything other than some $\hat{y}_i$ (true label). Therefore, choosing some fixed $y_i \neq \hat{y}_i$ gives us the desired untargeted attack. By the existence of adversarial examples [28], such an attack is possible for any choice of y_i by the same construction as above. And therefore, it honors the same bounds. □

Proposition 1 (Hardness of backdoor detection - I)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a ReLU and $g : \mathbb{R}^n \to \mathbb{R}$ be a function. Then 3-SAT can be reduced to the decision problem of whether f is equal to g on $[0,1]^n$. Hence checking if $f \equiv g$ on $[0,1]^n$ is NP-hard.*

Proof. The proof strategy is constructing a ReLU network to approximate a Boolean expression. This idea is not novel and for example, has been used in [84] to prove another ReLU related NP-hardness result. Nonetheless, we provide an independent construction here.

Let us define Backdoor as the following decision problem. Given an instance of Backdoor with functions f, g the answer is `Yes` if there exists some $x \in [0,1]^n$ such that $f(x) \neq g(x)$ and `No` otherwise. We will reduce 3-Sat to Backdoor. Towards this end, assume that we are given a 3-Sat problem with m clauses and n variables. Note that $n \leq 3m$ and $m \leq \binom{2n}{3}$, that is both are within polynomial factors of each other. Therefore, the input size of the 3-Sat is poly(m). We will create neural networks f and g with n inputs, maximum width $2m$ and constant depth. The weight matrices will have dimensions at most $\max\{2m, n\} \times \max\{2m, n\} = \text{poly}(m) \times \text{poly}(m)$ and similarly the bias vectors will have dimensions at most poly(m). Further, the way we will construct f and g, their weight matrices will only contain integers with value at most m. This means that each integer can be represented in $O(\log(m))$ bits. Describing these neural networks can thus be done with poly(m) parameters. Thus, the input size of Backdoor is also poly(m). For now, assume that f and g are created (in poly(m) time) such that $f \not\equiv g$ on $[0,1]^n$ if and only if the 3-Sat is satisfiable. Then, we have shown that if an algorithm can solve Backdoor in poly(m) time, then Sat can also be solved in poly(m) time; or in other words we have reduced 3-Sat to Backdoor. Thus, all that remains to do is to construct in poly(m) time, f and g such that $f \not\equiv g$ on $[0,1]^n$ if and only if the 3-Sat is satisfiable.

We will describe how to create the ReLU for f. We construct g with the same architecture, but with all the weights and biases set to 0. Thus the question of $f \equiv g$ on $[0,1]^n$ becomes $f \equiv 0$ on $[0,1]^n$. Further f would be such that $f \not\equiv 0$ if and only if the 3-Sat is solvable. Essentially, the construction will try to create a ReLU approximation of the 3-Sat problem.

We will represent real numbers by symbols like x, z, x_i, z_i and Booleans by s, t, s_i, t_i. The real vector $[x_1, \ldots, x_n]^\top$ will be denoted as $\boldsymbol{x}$. Similarly we will represent Boolean vector $[s_1, \ldots, s_n]^\top$ as $\boldsymbol{s}$.

Let the 3-Sat problem be $h : \mathbb{B}^d \to \mathbb{B}$, where

$$h(\boldsymbol{s}) = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{3} t_{i,j} \right),$$

such that $t_{i,j}$ is either s_k or $\neg s_k$ for some $k \in [d]$.

Now we start the construction of $f : \mathbb{R}^d \to \mathbb{R}$. Let $\widehat{x}_i = \sigma(2x_i - 1)$ and $\overline{x}_i = \sigma(1 - 2x_i)$ for all $i \in [d]$ where $\sigma(\cdot)$ represents the ReLU function. These can be computed with 1 layer of ReLU with width $2n$. Roughly speaking if we think of `True` as being equal to the real number 1 and `False` as equal to the real number 0, then we want $\widehat{x}_i$ to approximate s_i and $\overline{x}_i$ to approximate $\neg s_i$.

Next for all $i \in [m]$, $j \in [3]$, define

$$z_{i,j} = \begin{cases} \widehat{x}_k & \text{if } t_{i,j} = s_k \\ \overline{x}_k & \text{if } t_{i,j} = \neg s_k \end{cases}.$$

Then,

$$f(\boldsymbol{x}) = \sigma\left(\left(\sum_{i=1}^{m} f_i(x) \right) - m + 1 \right) \tag{8}$$
$$\text{where, } f_i(\boldsymbol{x}) = \sigma\left(\sum_{j=1}^{3} z_{i,j} \right) - \sigma\left(\left(\sum_{j=1}^{3} z_{i,j} \right) - 1 \right).$$

Again, roughly speaking we want $f_i(\boldsymbol{x})$ to approximate $\bigvee_{j=1}^{3} t_{i,j}$ and $f(\boldsymbol{x})$ to approximate $h(\boldsymbol{s})$. The decomposition of f above is written just for ease of understanding, but in its following form, we can see that it can be computed in 2 layers and width $2m$, using $\widehat{x}_i$ and $\overline{x}_i$

$$f(\boldsymbol{x}) = \sigma\left(\left(\sum_{i=1}^{m} \sigma\left(\sum_{j=1}^{3} z_{i,j} \right) - \sum_{i=1}^{m} \sigma\left(\left(\sum_{j=1}^{3} z_{i,j} \right) - 1 \right) \right) - m + 1 \right).$$

Thus the construction of f from h is complete and we can see that this can be done in polynomial time. Now we need to prove the correctness of the reduction.

We first show that 3-Sat $\implies$ Backdoor. This is the simpler of the two directions. Let $\boldsymbol{s}$ be an input such that $h(\boldsymbol{s})$ is True. Then create $\boldsymbol{x}$ as $x_i = 1$ if s_i =True and $x_i = 0$ otherwise. Putting this in Eq. (8) gives $f(\boldsymbol{x}) = 1$ and thus $f \not\equiv g$ on $[0,1]^n$.

Now, we show Backdoor $\implies$ 3-Sat. Let there be an $\boldsymbol{x}$ such that $f \not\equiv g$, which is the same as $f(\boldsymbol{x}) > 0$. First, assume that $x_k \geq 0.5$. Then, we see that increasing x_k increases $\widehat{x}_k$. This would increase all the $z_{i,j}$ which are defined as $\widehat{x}_k$. Further, $x_k \geq 0.5$ implies that $\overline{x}_k = 0$ and thus increasing x_k does not further decrease $\overline{x}_k$. Thus, all the $z_{i,j}$ which are defined as $\overline{x}_k$ do not decrease. Note that f is a non-decreasing function of $z_{i,j}$. This means that we can simply set x_k to 1 and the the value of $f(\boldsymbol{x})$ will not decrease.

Similarly, for the case that $x_k < 0.5$, we can set x_k to 0 and the value of $f(\boldsymbol{x})$ will not decrease.

This way, we can find a vector $\boldsymbol{x}$ which has only integer entries: 0 or 1 and $f(\boldsymbol{x}) > 0$. Because f consists of only integer operations, this means that $f(\boldsymbol{x}) \geq 1$. Looking at Eq. (8) and noting that $0 \leq f_i(\boldsymbol{x}) \leq 1$, we see that this is only possible if $f_i(\boldsymbol{x}) = 1$ for all i. Set s_i =True if $x_i = 1$ and s_i =False if $x_i = 0$. Then $f_i(x) = 1$ implies that $\bigvee_{j=1}^{3} t_{i,j}$ =True for all i. Thus, $h(\boldsymbol{s})$ is True. □

Proposition 2 (Hardness of backdoor detection - II)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a ReLU and $g : \mathbb{R}^n \to \mathbb{R}$ be a function. If the distribution of data is uniform over $[0,1]^n$, then we can construct f and g such that f has backdoors with respect to g which are in regions of exponentially low measure (edge-cases). Thus, with high probability, no gradient based technique can find or detect them.*

Proof. For ease of exposition, we create f and g with a single neuron. However, the construction easily extends to single layer NNs of arbitrary width. Also, assume we are in the high-dimensional regime, so that n is large. (Note that n here refers to the input dimension, not the number of samples in our dataset.)

Define the two networks and the backdoor as follows:

$$\begin{aligned} f(\mathbf{x}) &= \max(\mathbf{w}_1^\top \mathbf{x} - b_1,\ 0) \\ g(\mathbf{x}) &= \max(\mathbf{w}_2^\top \mathbf{x} - b_2,\ 0) \\ \mathcal{B} &= \{\mathbf{x} \in [0,1]^n : \mathbf{w}_1^\top \mathbf{x} \geq b_2\} \end{aligned}$$

where $\mathbf{w}_1 = (\frac{1}{n}, \frac{1}{n} \ldots, \frac{1}{n})^\top, b_1 = 1$ and $\mathbf{w}_2 = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})^\top, b_2 = \frac{1}{2}$

Note that equivalently, $\mathcal{B} = \left\{\mathbf{x} \in [0,1]^n : \mathbf{1}^\top \mathbf{x} \geq \frac{n}{2}\right\}$ and its measure of is given by:

$$\begin{aligned} P(\mathcal{B}) &= P\left(\sum_{i=1}^{n} x_i \geq \frac{n}{2}\right) \\ &= P\left(\frac{1}{n}\sum_{i=1}^{n} x_i \geq \frac{1}{2}\right) \\ &\leq e^{-n/2} \end{aligned}$$

where the last step follows from Hoeffding's inequality. Since n is large, we have that $\mathcal{B}$ has exponentially small measure.

We now compare the two networks on $[0,1]^n \setminus \mathcal{B}$. Clearly $\mathbf{w}_1^\top \mathbf{x} - b_1 < \mathbf{w}_2^\top \mathbf{x} - b_2 < 0$.

Therefore,

$$f(\mathbf{x}) = g(\mathbf{x}) = 0 \quad \forall\ \mathbf{x} \in \mathbf{X} \setminus \mathcal{B}$$

and

$$\nabla f(\mathbf{x}) = \nabla g(\mathbf{x}) = 0 \quad \forall\ \mathbf{x} \in \mathbf{X} \setminus \mathcal{B}.$$

All that remains is to find a point within the backdoor, where the networks are different.

Consider $\mathbf{x}_B = (1,1,\ldots,1)^n$. $\mathbf{w}_1^\top \mathbf{x}_B - b_1 = 0$. However, $\mathbf{w}_2^\top \mathbf{x}_B - b_2 = 1 - \frac{1}{2} = \frac{1}{2}$. Therefore,

$$||f(\mathbf{x}_B) - g(\mathbf{x}_B)|| = \frac{1}{2}$$

Clearly, f and g are identical in terms of zeroth and first order information on the entire region $[0,1]^n$ except for $\mathcal{B}$. Therefore any gradient based approach to find the backdoor region $\mathcal{B}$ would fail unless we initialize inside the backdoor region, which we have shown to be of exponentially small measure. $\square$