On the Robustness of Distributed Machine Learning Against Transfer Attacks

Sébastien Andreina[1], **Pascal Zimmer**[2], **Ghassan Karame**[2]
[1]NEC Labs Europe, Germany
[2]Ruhr University Bochum, Germany
sebastien.andreina@neclab.eu, {pascal.zimmer, ghassan.karame}@rub.de

Abstract

Although distributed machine learning (distributed ML) is gaining considerable attention in the community, prior works have independently looked at instances of distributed ML in either the training or the inference phase. No prior work has examined the combined robustness stemming from distributing both the learning and the inference process.

In this work, we explore, for the first time, the robustness of distributed ML models that are fully heterogeneous in training data, architecture, scheduler, optimizer, and other model parameters. Supported by theory and extensive experimental validation using CIFAR10 and FashionMNIST, we show that such properly distributed ML instantiations achieve across-the-board improvements in accuracy-robustness tradeoffs against state-of-the-art transfer-based attacks that could otherwise not be realized by current ensemble or federated learning instantiations. For instance, our experiments on CIFAR10 show that for the Common Weakness attack, one of the most powerful state-of-the-art transfer-based attacks, our method improves robust accuracy by up to 40%, with a minimal impact on clean task accuracy.

Code — https://github.com/RUB-InfSec/distributed_learning_robustness

Extended version — https://arxiv.org/abs/2412.14080

Introduction

Nowadays, many applications increasingly rely on various forms of distributed learning. For instance, Google spell-checking utilizes horizontal federated learning (Zhang et al. 2023b), and Apple, among others, may also be adopting similar approaches (Paulik et al. 2021). Autonomous vehicles are soon expected to follow suit (Chellapandi et al. 2024). In the financial sector, vertical federated learning is currently employed (Liu et al. 2024), and many researchers have already begun exploring decentralized learning without a central authority (Lian et al. 2017; Dhasade et al. 2023).

Although federated and decentralized learning is gaining considerable interest and attention from practitioners and researchers, the concept of distributed learning was first attempted with ensemble learning. Here, weak learners employing different model architectures (typically co-located on the same machine) jointly train using the same dataset; inference requires some sort of majority vote among those weak learners. Unlike ensemble learning, federated learning typically requires weak learners to jointly train a global model (optionally with the help of a central server). The main motivation of these instantiations is to obviate the need for weak learners to share their local training data, hence increasing learners' privacy while allowing for diverse data to be effectively used for training. As such, current federated learning approaches primarily aim to diversify data sources for training but do not directly alter the inference process. In contrast, ensemble learning focuses mainly on inference by enhancing model diversity. The literature features a number of contributions that analyze the privacy provisions of FL and its resistance to backdoors on the one hand and the robustness offered by ensemble learning on the other hand. *No prior work has examined the combined robustness and security provisions stemming from distributing both the learning and the inference process.* This is particularly relevant since centralized ML models, such as deep neural networks, have been shown to lack robustness against adversarial examples (Szegedy et al. 2014; Biggio et al. 2013).

In this paper, we address this gap and explore whether pure *distributed learning*, a variant instantiation combining the benefits of both federated/decentralized learning and traditional ensemble learning, can provide increased robustness against transfer attacks. We specifically focus on transfer attacks because they can be executed by resource-constrained adversaries that do not need to know the model parameters to create a surrogate model and conduct transferable white-box attacks. More specifically, we consider a setting where weak learners can choose their dataset for training. However, unlike FL, learners can also independently choose their training (such as optimizer and scheduler) and model (i.e., architecture) parameters. We argue that such a hybrid setting allows us to analyze, for the first time, the robustness of distributed ML models that are heterogeneous in *both* their training data *and* training and model parameters. We contrast this to previous work that has independently looked at instances of ML in each isolated phase. For example, (Wu et al. 2021) demonstrated that ensembles of models are more robust to certain attacks compared to single models. Similarly, (Demontis et al. 2019) highlighted the significance of model architecture on the transferability of adversarial examples, and (Zhang et al.

2023a) explored the robustness of FL frameworks, particularly focusing on the challenges of employing FL-based adversarial training in non-IID data settings. In contrast, we aim to address the following research questions in this work:

RQ1 To what extent is distributed ML more robust than existing traditional ensemble learning against transfer-based attacks?

RQ2 Are there specific model parameters of particular relevance to increase robustness in a distributed setting?

RQ3 How does the training data distribution between different distributed models impact the overall robustness?

RQ4 How do distributed aggregation schemes affect the robustness?

We conduct an extensive robustness evaluation of our approach with state-of-the-art transfer-based attacks and find across-the-board improvements in robustness against all considered attacks. For instance, our experiments on CIFAR10 show that for CW (Chen et al. 2024), one of the most powerful state-of-the-art transfer attacks, our method improves robust accuracy by up to 33.6% and 41.2%, with a minimal impact on clean task accuracy of at most between 1.1% and 13.5%.

Background & Related Work

ML Paradigms

"Centralized" Learning is the de-facto standard learning paradigm, where a single entity possessing a dataset trains a single model on it. While multiple servers can be used to improve the learning process speed, they typically all run the same code based on the same parameters and synchronize the results of the training at each epoch.

Ensemble Learning is a method that uses multiple learning algorithms and architectures to obtain better predictive performances compared to its centralized counterparts (Pang et al. 2019a).

Federated and Decentralized Learning are settings where the complete training data is unavailable to a single entity due to privacy concerns. Both paradigms distribute the training process across multiple nodes, utilizing locally available data. Federated learning relies on a trusted centralized party to coordinate the training process and aggregate the models computed locally by each node. In contrast, decentralized learning eliminates the central trusted party by leveraging peer-to-peer communication to share models among the nodes.

Transfer-based attacks

While some early techniques required complete access to the target classifier to generate an adversarial example, recent research showed the feasibility of *transferring* adversarial examples in a so-called "blackbox" setting (Dong et al. 2018; Byun et al. 2022; Wang and He 2021; Guo, Li, and Chen 2020; Wang et al. 2021; Wu et al. 2021). Here, the adversary does not have access to the model parameter but still has complete knowledge of the training parameters, such as architecture, dataset, hyperparameters, etc. The adversary can then train a substitute model and use it to generate adversarial examples that are likely to be misclassified by the target model. In this case, the adversary is not required to have access to the trained model.

Given a target classifier C_t and a local "surrogate" classifier C_s, an attacker can simply run the standard white-box attack on surrogate classifier C_s to generate an adversarial example that is likely to transfer to the target classifier C_t. Most previous work assumes the surrogate classifier C_s has been trained using the same training data as the target classifier C_t (Goodfellow, Shlens, and Szegedy 2015; Byun et al. 2022; Zhang et al. 2022; Guo, Li, and Chen 2020; Gao et al. 2021). The transferability of the attack is then evaluated between different source and target classifier architectures trained with the same training set.

While adversarial examples generally transfer, the success rate of transfer-based attacks using plain standard white-box attacks is limited. To solve this shortcoming, there have been numerous works focused on improving the transferability of adversarial examples; (Dong et al. 2018) improved transferability of FGSM (Goodfellow, Shlens, and Szegedy 2015) by adding momentum to the gradient; (Xie et al. 2019) uses input diversity using random transform to improve the resilience of the adversarial transform. More recent work, such as (Huang and Kong 2022; Naseer et al. 2021; Byun et al. 2022; Zhang et al. 2022; Guo, Li, and Chen 2020; Gao et al. 2021) use more involved techniques and achieve transferability very close to 100% on undefended models.

Related Work

Previous work (Demontis et al. 2019; Inkawhich et al. 2020) explored the transferability of adversarial examples but did not analyze the impact of training parameters on the robustness of decentralized deployments. Additionally, the generalization of (Demontis et al. 2019) is restricted to binary classifiers, does not include attacks optimized for transferability (focusing only on white box attacks), and does not investigate the impact of parameters beyond the architecture. On the other hand, while (Mahmood et al. 2020) analyzed the impact of the training data over transferability, they did not consider other model parameters.

Additionally, previous work outlined the improvement in robustness provided by ensemble learning (Kurakin et al. 2018) over single models. (Pang et al. 2019b) improved the robustness of ensemble models by devising a strategy to increase the diversity of the trained models in ensemble training, reducing the transferability and yielding enhanced robustness.

We conclude that previous work has looked at the isolated impact of a given (training) parameter on transferability. No prior work has examined the combined robustness stemming from distributing both the learning and the inference process and the impact of the various hyperparameters on transferability.

Methodology

Preliminaries and Notations

We denote the sample and label spaces with $\mathcal{X}$ and $\mathcal{Y}$, respectively, and the training data with $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$, where N is the training set size. A DNN-based classifier

$f_\theta : \mathcal{X} \to [0,1]^{|\mathcal{Y}|}$ is a function (parameterized by θ) that, given an input x outputs the probability that the input is classified as each of the $n = |\mathcal{Y}|$ classes. The highest prediction probability in this vector, i.e., $\max_{i\in[n]} f_{\theta,i}(x)$, is also called the *confidence* of the model in the classification of the sample. The prediction of the classifier can be derived as $y = C(x) := \arg\max_{i\in[n]}(f_{\theta,i}(x))$.

We define an adversarial example x' as a genuine image x to which carefully crafted adversarial noise is added, i.e., $x' = x + \zeta$ for a small perturbation ζ such that x' and x are perceptually indistinguishable to the human eye and yet are classified differently.

Given a genuine input $x_0 \in \mathbb{R}^d$ predicted as $C(x_0) = s$ (source class), x' is an *adversarial example* of x_0 if $C(x') \neq s$ and $\|x' - x_0\|_p \leq \varepsilon$ for a given distortion bound $\varepsilon \in \mathbb{R}^+$ and l_p norm. The attacker searches for adversarial inputs x' with low distortion while maximizing a loss function $\mathcal{L}$, e.g., cross-entropy loss. Formally, the optimization problem is defined as follows:

$$\zeta = \underset{\|\zeta\|_p \leq \varepsilon}{\arg\max}\, \mathcal{L}(y, x+\zeta, \theta) \tag{1}$$

Main Intuition

Existing work on transferability (Huang and Kong 2022; Chen et al. 2024; Demontis et al. 2019; Mao et al. 2022) highlighted that adversarial examples transfer better when the surrogate model is similar to the target model. However, up until now, this similarity has mostly been investigated in terms of architecture types, ignoring the impact of other training parameters, such as the choice of optimizer, scheduler, and amount of available data.

We show in what follows that the explicit introduction of model heterogeneity by varying other parameters relevant for training, e.g., optimizer, scheduler, and amount of available data, is expected to reduce the overall transferability. We show that (1) the choice of model parameters, e.g., architecture, scheduler, optimizer, promotes gradient diversity and (2) transferability can be quantified with a gradient comparison of the surrogate and target model (Demontis et al. 2019).

Proposition 1 *A model f with major parameter configuration $\mathcal{P}$ and hyperparameter configuration $\mathcal{H}$ is optimized to model parameters θ during training. Heterogeneity, i.e., a change in these parameter configurations, $\hat{\mathcal{P}}, \hat{\mathcal{H}}$, results in a model $\hat{f}$ (parameterized by $\hat{\theta}$) with a changed loss landscape and hence diverse (by a sufficiently large γ) model gradients for a set of input points $\mathcal{X}$. Concretely,*

$$\sum_{x\in\mathcal{X}} \|\nabla_x \mathcal{L}(y,x,\theta) - \nabla_x \mathcal{L}(y,x,\hat{\theta})\|_2 > \gamma \tag{2}$$

Recall that the supervised training procedure for a generic multi-class classifier C aims at finding the optimal set of parameters $\theta_{\mathcal{P},\mathcal{H}}$ given parameter configurations $\mathcal{P}, \mathcal{H}$ that minimize the aggregated loss over the entire training set $\mathcal{D}$:

$$\min_{\theta_{\mathcal{P},\mathcal{H}}} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(y,x,\theta_{\mathcal{P},\mathcal{H}}) \tag{3}$$

with sample x, ground-truth y, and loss-function $\mathcal{L}$.

We expect that the loss landscape defined by the parameters θ, i.e., the converged result of the non-convex optimization problem, depends on many factors, such as the amount of available data, i.e., the partition of data across N nodes, architecture, and potentially on the optimization strategy. Beyond that, the respective hyperparameters $\mathcal{H}$, i.e., learning rate μ, momentum ν, and weight decay λ, might also impact the convergence behavior of the optimization procedure. As a result, a change in $\mathcal{P}$ and $\mathcal{H}$ converges to a different set of model parameters, i.e., $\hat{\theta}_{\hat{\mathcal{P}},\hat{\mathcal{H}}}$. For clarity, we denote $\theta := \theta_{\mathcal{P},\mathcal{H}}$ and $\hat{\theta} := \theta_{\hat{\mathcal{P}},\hat{\mathcal{H}}}$. We empirically show an increase in gradient diversity with an increase in heterogeneity in $\mathcal{P}$ and $\mathcal{H}$ in the full version (Andreina, Zimmer, and Karame 2024).

Proposition 2 *The transferability of adversarial examples from a surrogate model to a target model can be evaluated by comparing the model gradients for a given point using the cosine similarity function with:*

$$S(x,y) = \frac{\nabla_x \mathcal{L}(y,x,\hat{\theta})^\intercal \nabla_x \mathcal{L}(y,x,\theta)}{\|\nabla_x \mathcal{L}(y,x,\hat{\theta})\|_2 \|\nabla_x \mathcal{L}(y,x,\theta)\|_2} \tag{4}$$

As the ultimate goal of a transfer-based attack is to evade a target model, we are interested in the loss $\mathcal{L}$ that an adversarial example crafted on a surrogate model (parameterized by $\hat{\theta}$), i.e., $x' = x + \hat{\zeta}$, can obtain on the target model (parameterized by θ). This loss, namely $\mathcal{L}(y, x+\hat{\zeta}, \theta)$, defines transferability. In practice, we can rewrite $\mathcal{L}$ with a linear approximation (Goodfellow, Shlens, and Szegedy 2015) as:

$$\mathcal{L}(y, x+\hat{\zeta}, \theta) \approx \mathcal{L}(y,x,\theta) + \hat{\zeta}^\intercal \nabla_x \mathcal{L}(y,x,\theta) \tag{5}$$

This is subsequently combined with the optimization goal in Equation (1). To maximize the second term, we maximize the inner product over an ε-sized sphere as follows, with l_q being the dual norm of l_p:

$$\max_{\|\hat{\zeta}\|_p \leq \varepsilon} \hat{\zeta}^\intercal \nabla_x \mathcal{L}(y,x,\hat{\theta}) = \varepsilon \|\nabla_x \mathcal{L}(y,x,\hat{\theta})\|_q \tag{6}$$

To maximize Equation (6) and subsequently also Equation (5), we insert an optimal value of $\hat{\zeta} = \varepsilon \frac{\nabla_x \mathcal{L}(y,x,\hat{\theta})}{\|\nabla_x \mathcal{L}(y,x,\hat{\theta})\|_2}$ for an l_2 norm. As a result, the change in loss under a transfer attack for such a point is defined as:

$$\Delta_{\mathcal{L}} = \varepsilon \frac{\nabla_x \mathcal{L}(y,x,\hat{\theta})^\intercal}{\|\nabla_x \mathcal{L}(y,x,\hat{\theta})\|_2} \nabla_x \mathcal{L}(y,x,\theta) \leq \varepsilon \|\nabla_x \mathcal{L}(y,x,\theta)\|_2 \tag{7}$$

The left-hand side of the equation reflects the black-box case, which is upper-bounded by the white-box case on the right-hand side. Rearranging Equation (7) reveals that the change in loss and, hence the impact of a transfer-based attack crafted for a sample x can be inferred by comparing the gradient similarity of the surrogate and target model evaluated at sample x as follows:

$$S(x,y) = \frac{\nabla_x \mathcal{L}(y,x,\hat{\theta})^\intercal \nabla_x \mathcal{L}(y,x,\theta)}{\|\nabla_x \mathcal{L}(y,x,\hat{\theta})\|_2 \|\nabla_x \mathcal{L}(y,x,\theta)\|_2} \tag{8}$$

This illustrates that the change in loss between the surrogate and target model for a given sample x, i.e., the transferability, depends on the gradient similarity of the respective models. As a result, a decrease in gradient similarity, i.e., an increase in model heterogeneity (cf. Proposition 1), decreases the obtainable change in loss on the target classifier with an adversarial example crafted on a surrogate classifier (cf. Proposition 2), i.e., reduces transferability.

Experimental Approach

Training parameters

Training a machine learning (ML) model involves numerous decisions, including selecting the model's architecture, such as VGG or DenseNet, and determining its width and depth, typically based on the complexity of the task at hand. Additionally, hyperparameters such as learning rate and momentum require careful tuning. This tuning often involves empirically testing various values within a defined range to identify the configuration that yields optimal convergence.

Additionally, while the standard stochastic gradient descent (SGD) optimizer is commonly used for smaller ML projects, various alternative optimizers, such as ADAM, Adagrad, and Rprop, have been developed to enhance convergence depending on the nature of the training data. Furthermore, research has demonstrated that starting with a high learning rate and subsequently decreasing it as the number of epochs progresses can lead to improved performance and accuracy (Kingma and Ba 2015). This approach has catalyzed the development of various learning rate schedulers, including StepLR and ExponentialLR, which automatically adjust the learning rate to maximize convergence.

In our experiments, we opted to separate the parameters into two groups: the major parameters, which encompass choices of architecture, optimizer, and scheduler, from the hyperparameters, which include the learning rate, momentum, and weight decay. As detailed in Table 1, we selected a set of eight different architectures (A), nine different optimizers (O), and five different schedulers (S). These selections were derived from commonly used values in academic research and practical projects. For the choice of architectures, we simplified the selection process by only selecting one representative architecture for each family of architectures—such as VGG11 for the VGG family.

Hyperparameters, on the other hand, are tailored to the specific combination of major parameters. For example, a learning rate (μ) of 0.1, although common with SGD, leads to divergence when combined with the Adam optimizer. Thus, we tuned the hyperparameters as follows.

Note that, in all scenarios we consider, we partitioned the training dataset into mutually exclusive subsets, ensuring that each weak learner is allocated a distinct portion of the training data. Unless otherwise specified, the classes are balanced, ensuring that each weak learner receives an equal proportion of samples from each class.

Parameter tuning

To ensure the quality of the trained model, each weak learner undergoes a phase of hyperparameter tuning prior to the actual training process. The exact process is summarized in Algorithm 1. Initially, each weak learner takes as input its local dataset D, and a set of common parameters `params`, and draws the remaining parameters (`diverseParams`) randomly from the set of possible values (Line 4). Each node then evaluates a wide range of hyperparameter values over two subsequent tuning rounds, leveraging the Ray Tune tool (Liaw et al. 2018) to identify the optimal set of parameters (Line 6). Specifically. During the fine-tuning steps, we split the node's training data into an 80%-20% ratio for the training and validation sets, respectively. Ray Tune is configured to run up to 100 experiments per tuning step to determine the most effective hyperparameters. This approach ensures that each weak learner is finely tuned, enhancing the model's overall performance and robustness. Once a node finishes tuning its local training parameters, it runs a complete training round for 200 epochs (Line 7) .

We argue that this training approach is more realistic than single-shot training based on a random selection of parameters. In most distributed ML settings, each node can expect to spend some effort ensuring that their local training achieves an acceptable level of accuracy.

Algorithm 1: Training phase for Weak Learners

1: **Input:** dataset D, common parameters `params`, major parameters that should be randomly selected `diverseParams`
2: **Output:** Trained weak learner model `model`

`/* For each` $\mathcal{P}$ `that should be diversified, randomly select it from the list of valid parameters */`
3: **for** p in diverseParams **do**
4: params.p = random(p.possibleValues)
5: **end for**

`/* Local tuning step to derive the best hyperparameters` $\mathcal{H}$`: `μ`, `ν`, and `λ` */`
6: $\mathcal{H}$ =μ, ν, λ = `Tune(train_model, params, data)`

`/* Final training based on the selected` $\mathcal{P}$ `and tuned` $\mathcal{H}$ `*/`
7: model = `train_model(params,` $\mathcal{H}$`, data)`
8: **return** model

Impact of Parameter Diversity

We devised four scenarios to answer our research questions and evaluate the impact of parameter diversity on transferability. Each scenario involves a varying number of nodes, denoted $N \in 3, 5, 7$. Unless otherwise specified, each node is trained on its disjoint dataset drawn from the complete dataset following a uniform distribution.

- **Ensemble Learning (ENS)** serves as our baseline, employing standard ensemble learning techniques. Here, we rely on publicly available code to train models to mitigate any potential bias.
- **Independent Tuning (IT)** shares the same major parameter among each weak learner but tunes its local hyperparameters independently based on its available data.
- **$\mathcal{P}$-Parameter Diversity ($\mathbf{D}_{\mathcal{P}}$)** extends IT by introducing diversity to the parameter $\mathcal{P} \in \{A, O, S\}$. Similarly to

	Parameter	Variable	Possible values
Major Parameters $\mathcal{P}$	Number of nodes	N	1, 3, 5, 7
	Architecture	A	VGG19, MobileNetv2, EfficientNet_b0, DenseNet121, SimpleDLA, ResNet18,0 ResNext29_2x64d, DPN92, SeNet18, googlenet, shufflenetg2, regnetx_200mf, preactresnet18
	Optimizer	O	SGD, $\text{SGD}_{momentum}$, Adam $\text{SGD}_{nesterov}$, NAdam, Adagrad, ASGD, Rprop, RMSprop
	Scheduler	S	CosineAnnealingLR, StepLR, ExponentialLR, CyclicLR, ReduceLROnPlateau
Hyper-params $\mathcal{H}$	Learning rate	μ	$[0.0001, 0.1]$
	Momentum	ν	$[0, 0.99]$
	Weight decay	λ	$[0.00001, 0.01]$

Table 1: Parameters considered for the experiments.

IT, the remaining major parameters are the same for all the weak learners, and each weak learner performs a local tuning process based on their local dataset.

By systematically analyzing these scenarios, we aim to elucidate the effects of parameter diversity on the transferability of learned models across different contexts. In all our evaluations, the probability vector of each weak learner is averaged, and the final classification is determined by selecting the class with the highest probability. We consider the impact of other voting methods, such as *weighted voting* and *hard voting*, in RQ4.

Experiments

Let $S \subset \mathcal{X} \times \mathcal{Y}$ denote the set of (labeled) genuine samples provided to the attacker $\mathcal{A}$, let $n := |S|$, and let ε denote the distortion budget. To compute the attack success rate (ASR) in practice, we determine the number of successful adversarial examples generated by the attacker:

$$n_{\text{succ}} := \left| \left\{ (x, x') \in \mathcal{X} \times \mathcal{A}(S) \;\middle|\; \begin{array}{l} C(x) \neq C(x') \wedge \\ \|x' - x\|_p \leq \varepsilon \end{array} \right\} \right| \quad (9)$$

where $\mathcal{A}(S)$ denotes the set of candidate adversarial examples output by $\mathcal{A}$ in a run of the attack on input S. The ASR is defined as $\mathsf{ASR} := {}^{n_{\text{succ}}}/_{n}$, i.e., the ratio of successful adversarial examples. The complement of the ASR is the robust accuracy (RA) of the classifier, i.e., $\mathsf{RA} = 1 - \mathsf{ASR}$.

Experimental Setup

All our experiments were run on an Ubuntu 24.04 machine featuring two NVIDIA A40 GPUs and one NVIDIA H100 GPU, two AMD EPYC 9554 64-core Processors, and 768 GB of RAM. All scenarios were executed using Python 3.9.18, CUDA 12.5, Pytorch 2.2.1, and Ray Tune 2.9.3. Due to the limited amount of data available to each weak learner, we rely on well-known data augmentation techniques such as random flipping and random resize and cropping from the Torchvision transformsV2 library.

Due to space constraints, we include our full results and analysis for the CIFAR10 dataset (Krizhevsky 2009) in Table 6 and provide the main results for the FashionMNIST dataset in Table 5. We note, however, that most of our findings are consistent across both datasets.

All our results are averaged over five independent runs. Where appropriate, we also provide the 95% confidence interval.

Selection of the Attacks

We evaluate our setup against the state-of-the-art common weakness attack (CW) (Chen et al. 2024), which improves transferability by using multiple (an ensemble of) surrogate models, effectively outperforming existing attacks. We rely on the CW attack for the evaluation as we argue it is the best strategy for an adaptive adversary that knows that the inference model is distributed across many different architectures trained using disparate parameters. In addition to CW, we consider two derivative attacks: Sharpness Aware Minimization (SAM) and Cosine Similarity Encourager (CSE), which are defined in the same paper. For the sake of completeness, our evaluation includes all three variants. Note that we do not evaluate attacks like (Bryniarski et al. 2022), which focus on multi-objective optimization challenges involved in fooling a target model while simultaneously circumventing defenses. Since we evaluate the transferability on undefended models in our setup, we focus on the CW attack, owing to its superior transferability rate.

We define the maximum distortion bound $\varepsilon = 8/255$. For each considered attack, we generate an adversarial example for each sample of the test dataset. For example, in CIFAR10, this results in $10\,000$ adversarial examples per attack.

Evaluation Results

We now proceed to answer our research questions (RQ1-RQ4) empirically using experiments in CIFAR10 and FashionMNIST.

RQ1: We leverage the Pareto frontier to illustrate the obtainable accuracy-robustness tradeoffs across all $\text{D}_{\mathcal{P}}$ instantiations and our ensemble baseline (**ENS**). Recall that the Pareto frontier emerges as an effective tool to evaluate tradeoffs between the clean accuracy CA and the robust accuracy RA. A solution ω^* is *Pareto optimal* if there exists no other solution that improves all objectives simultaneously. Formally, given two solutions ω_1 and ω_2, we write $\omega_1 \succ \omega_2$ if ω_1 dominates ω_2, i.e., if $\mathsf{CA}(\omega_1) \geq \mathsf{CA}(\omega_2) \wedge \mathsf{RA}(\omega_1) \geq \mathsf{RA}(\omega_2)$, where $\mathsf{CA}(\omega)$ and $\mathsf{RA}(\omega)$ denote the clean accuracy and robust accuracy as functions of the parameter ω. The *Pareto frontier* is the set of Pareto-optimal solutions:

$$PF(\Omega) = \{\omega^* \in \Omega \mid \nexists \omega \in \Omega \text{ s.t. } \omega \succ \omega^*\}. \quad (10)$$

This enables us to compare the average RA (and respective CA) across all attacks, i.e., the last column of Table 6, in Figure 1.

For the ensemble baseline, we observe a wide range of CA between 82 and 94%, with a narrow and hence limited range

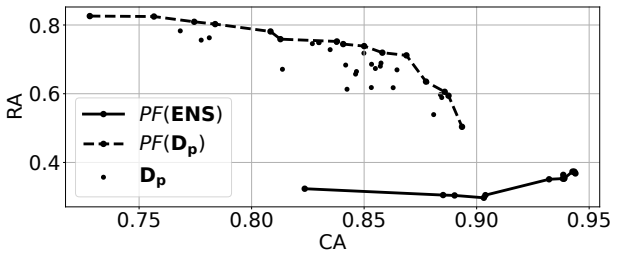

Figure 1: Pareto frontier of all accuracy-robustness tradeoffs of distributed instantiations ($D_{\mathcal{P}}$) compared to the baseline ensemble (**ENS**).

of RA with a maximum of 37%. In stark contrast, the $D_{\mathcal{P}}$ instances result in a broader range of tradeoffs. Concretely, we observe improvements in RA of $20-40\%$, i.e., primarily between $60-80\%$ with a maximum RA of up to 83%. While the prioritization of either CA or RA is use-case specific, we find a near-optimal point of operation for $\mathsf{CA}=87\%, \mathsf{RA}=71\%$.

RQ2: To explore the impact of parameter diversity on robustness, We rely on the standard Ordinary Least Squares regression method using the major parameters $\mathcal{P}$, i.e., number of nodes (N), diversity in architecture (A), optimizer (O), scheduling (S), and independent tuning (IT) as explanatory variables, to explain the robustness as the response variable. Our findings are detailed in Table 5, using standard ensemble models as the baseline. Our regression model demonstrates strong robustness, accounting for 87.4% of the variability of the dependent variable through the selected predictors, indicated by an R-squared value of 0.874. Notably, the robustness improves by 2.88% for each additional node. It is important to note that this linear model does not capture the eventual saturation effect, where robustness gains diminish as the size of N grows large. Furthermore, our results show an increase of 34.53% when the weak learners use diverse hyperparameters $\mathcal{H}$ through independent tuning (IT), including the learning rate μ, momentum ν, and weight decay λ. As noticed in the ablation study, we note, however, that the diversity of the optimizer, architecture, and scheduler does not appear to have a statistically significant contribution to the robustness.

RQ3: To assess the impact of non-IID (non-independent and identically distributed) data distribution on both accuracy and robustness, we conduct independent tuning experiments under two distinct data partitioning schemes: uniform distribution and Dirichlet distribution. The Dirichlet distribution, commonly employed in many Federated Learning (FL) deployments (Bagdasaryan et al. 2020; Andreina et al. 2021), allows us to simulate more realistic, heterogeneous data environments. In the case of the Dirichlet distribution, the number of samples of a given class is distributed among the weak learners with parameter $\alpha = 0.9$.

Our results (cf. Table 2) show that the accuracy of models trained with non-IID data decreases by 3.1% for $N=3$ and 2.6% for $N=7$ when compared to models trained with uniformly distributed data. On the other hand, the effect on the mean robustness is negligible, with an increase of only 0.2% and 0.6% when $N=3$ and $N=7$, hinting that non-IID data distribution among the nodes does not directly impact the robustness of the distributed model.

Exp.	N	CA	RA			
			SAM	CSE	CW	Mean
IT	3	0.87 $\pm$ 0.01	0.60 $\pm$ 0.05	0.61 $\pm$ 0.04	0.59 $\pm$ 0.04	0.60
IID	5	0.84 $\pm$ 0.01	0.71 $\pm$ 0.04	0.70 $\pm$ 0.02	0.69 $\pm$ 0.03	0.70
	7	0.81 $\pm$ 0.01	0.77 $\pm$ 0.04	0.74 $\pm$ 0.03	0.73 $\pm$ 0.03	0.74
IT	3	0.84 $\pm$ 0.01	0.60 $\pm$ 0.02	0.62 $\pm$ 0.02	0.60 $\pm$ 0.02	0.61
Non-	5	0.83 $\pm$ 0.01	0.70 $\pm$ 0.03	0.70 $\pm$ 0.03	0.68 $\pm$ 0.03	0.69
IID	7	0.79 $\pm$ 0.02	0.77 $\pm$ 0.03	0.74 $\pm$ 0.02	0.73 $\pm$ 0.02	0.75

Table 2: Impact of data distribution on the accuracy and robustness in the independent training scenario.

	coef	**std err**	**t**	**P > |t|**	**[0.025**	**0.975]**
const	0.5097	0.039	13.237	0.000	0.431	0.589
N	0.0354	0.007	5.064	0.000	0.021	0.050
Dirichlet	0.0042	0.023	0.184	0.855	-0.043	0.051

Table 3: Regression analysis of the impact of the data distribution on the robustness of the models in the independent training scenario.

This observation is further substantiated by our regression analysis shown in Table 3, using IT as the baseline and N and Dirichlet as the explanatory variables. Our analysis indicates that using the Dirichlet distribution does not produce statistically significant differences in robustness performance compared to the uniform distribution (i.e., the P-value for the t-test is high, supporting the null hypothesis).

RQ4: Last, we examine the impact of different voting schemes on the accuracy and robustness of distributed ML. We consider the following three schemes:

1. **Average voting**: In this scheme, the output vectors of each model are averaged, and the class with the highest average probability is selected as the final prediction. This has been the baseline scheme used in our previous experiments.
2. **Hard voting**: Each weak learner votes for the class with the highest probability according to its local model. The class that receives the majority of votes across all learners

Voting	N	CA	RA			
			SAM	CSE	CW	Mean
Average	3	0.86 $\pm$ 0.03	0.64 $\pm$ 0.06	0.65 $\pm$ 0.05	0.63 $\pm$ 0.04	0.64 $\pm$ 0.05
	5	0.83 $\pm$ 0.03	0.72 $\pm$ 0.06	0.71 $\pm$ 0.04	0.69 $\pm$ 0.05	0.70 $\pm$ 0.05
	7	0.80 $\pm$ 0.02	0.79 $\pm$ 0.03	0.76 $\pm$ 0.03	0.75 $\pm$ 0.03	0.76 $\pm$ 0.03
Hard	3	0.83 $\pm$ 0.05	0.63 $\pm$ 0.06	0.63 $\pm$ 0.05	0.61 $\pm$ 0.04	0.62 $\pm$ 0.05
	5	0.82 $\pm$ 0.03	0.71 $\pm$ 0.06	0.69 $\pm$ 0.04	0.68 $\pm$ 0.05	0.69 $\pm$ 0.05
	7	0.78 $\pm$ 0.03	0.78 $\pm$ 0.03	0.74 $\pm$ 0.02	0.73 $\pm$ 0.03	0.75 $\pm$ 0.03
Weighted	3	0.85 $\pm$ 0.03	0.64 $\pm$ 0.06	0.64 $\pm$ 0.05	0.62 $\pm$ 0.04	0.63 $\pm$ 0.05
	5	0.83 $\pm$ 0.03	0.71 $\pm$ 0.06	0.70 $\pm$ 0.05	0.69 $\pm$ 0.05	0.70 $\pm$ 0.05
	7	0.79 $\pm$ 0.02	0.79 $\pm$ 0.04	0.75 $\pm$ 0.03	0.74 $\pm$ 0.03	0.76 $\pm$ 0.03

Table 4: Accuracy and robustness of $\mathbf{D}_O$ across the different voting schemes.

		coef	std err	t	P > |t|	[0.025	0.975]
CIFAR10	**const**	0.2058	0.012	16.924	0.000	0.182	0.230
	N	0.0288	0.002	17.023	0.000	0.025	0.032
	IT	0.3453	0.010	35.539	0.000	0.326	0.364
	$\mathbf{D}_O$	-0.0012	0.006	-0.208	0.836	-0.013	0.010
	$\mathbf{D}_A$	0.0076	0.006	1.271	0.204	-0.004	0.019
	$\mathbf{D}_S$	-0.0089	0.006	-1.498	0.135	-0.020	0.003
	Hard	-0.0117	0.007	-1.723	0.086	-0.025	0.002
	Weighted	-0.0046	0.007	-0.679	0.498	-0.018	0.009
FashionMNIST	**const**	0.9403	0.003	306.244	0.000	0.934	0.946
	N	0.0030	0.000	6.692	0.000	0.002	0.004
	IT	0.0026	0.004	0.723	0.471	-0.005	0.010
	$\mathbf{D}_O$	0.0005	0.002	0.269	0.788	-0.003	0.004
	$\mathbf{D}_A$	0.0015	0.002	0.642	0.522	-0.003	0.006
	$\mathbf{D}_S$	-0.0014	0.002	-0.577	0.565	-0.006	0.003
	Hard	-0.0063	0.002	-3.489	0.001	-0.010	-0.003
	Weighted	-0.0048	0.002	-2.652	0.009	-0.008	-0.001

Table 5: Complete regression analysis of the impact of the different parameters and voting scheme on the average robustness of the distributed models.

is chosen as the final prediction.

3. **Weighted voting**: Similar to hard voting, each weak learner votes for the class with the highest probability, but the votes are weighted by the confidence level of the learner's model.

As shown in Table 4, the average voting scheme performs best compared to the other two schemes. Specifically, hard voting decreases accuracy by 3% for $N = 3$ and 2% for $N = 7$, with a corresponding decrease in mean robustness of 2% and 1%. Weighted voting results in smaller decreases, with accuracy decreasing by 1% for $N = 3$ and $N = 7$, and robustness by 1% for $N = 3$ and $N = 7$. This is further confirmed by our regression analysis in the last two rows of Table 5; on average, the hard and weighted voting schemes result in a performance decrease of 1.1% and 0.4%. However, these impacts are not statistically significant, with P-values of 0.08 and 0.49, both above the 0.05 threshold.

Ablation Study

Our results are summarized in Table 6, from which several key insights can be derived. Notably, we observe a consistent decline in accuracy as the number of nodes (N) increases in the distributed ML scenarios (denoted as the D_{AOS} rows in the table). Here, the accuracy slightly declines from 85% down to 82%. This degradation in performance can be primarily attributed to the division of the dataset among the weak learners, which reduces the amount of data available for each learner as the node count increases. In contrast, standard ensemble models (represented by the **ENS** rows) do not exhibit this limitation, as each model in the ensemble has access to the complete dataset.

Our results indicate a continuous improvement as N increases across ensemble and distributed ML models. For the baseline ensemble scenario, the mean robustness increases from 31% to 37% as N increases from 3 to 7. This trend is similar in the distributed ML system, where the robustness increases from 66% to 73%, an increase of 7% compared to standard ensemble models. Last but not least, we selectively measure the gradient similarity (cf. Equation (4)) on a subset of our results with the surrogate models. The complete gradient analysis is included in the full paper (Andreina, Zimmer, and Karame 2024).

Exp.	N	CA	RA			
			SAM	CSE	CW	Mean
ENS	3	0.88 ± 0.03	0.26 ± 0.02	0.34 ± 0.01	0.31 ± 0.0	0.31
	5	0.94 ± 0.0	0.29 ± 0.01	0.4 ± 0.01	0.37 ± 0.0	0.35
	7	0.94 ± 0.0	0.29 ± 0.0	0.43 ± 0.0	0.39 ± 0.0	0.37
$\mathbf{D}_{AOS}$	3	0.85 ± 0.02	0.66 ± 0.05	0.67 ± 0.03	0.65 ± 0.03	0.66
	5	0.85 ± 0.02	0.68 ± 0.04	0.68 ± 0.03	0.66 ± 0.03	0.67
	7	0.82 ± 0.01	0.75 ± 0.03	0.73 ± 0.02	0.72 ± 0.02	0.73
$\mathbf{D}_A$	3	0.85 ± 0.03	0.66 ± 0.06	0.66 ± 0.04	0.65 ± 0.04	0.66
	5	0.84 ± 0.02	0.73 ± 0.06	0.72 ± 0.04	0.71 ± 0.05	0.72
	7	0.82 ± 0.03	0.78 ± 0.03	0.75 ± 0.01	0.74 ± 0.01	0.76
$\mathbf{D}_O$	3	0.86 ± 0.03	0.64 ± 0.06	0.65 ± 0.05	0.63 ± 0.04	0.64
	5	0.83 ± 0.03	0.72 ± 0.06	0.71 ± 0.04	0.69 ± 0.05	0.70
	7	0.80 ± 0.02	0.79 ± 0.03	0.76 ± 0.03	0.75 ± 0.03	0.76
$\mathbf{D}_S$	3	0.87 ± 0.04	0.59 ± 0.1	0.61 ± 0.08	0.60 ± 0.08	0.60
	5	0.84 ± 0.03	0.72 ± 0.07	0.72 ± 0.06	0.70 ± 0.06	0.71
	7	0.81 ± 0.04	0.76 ± 0.09	0.73 ± 0.07	0.72 ± 0.07	0.74
$\mathbf{D}_{AO}$	3	0.87 ± 0.01	0.61 ± 0.02	0.63 ± 0.02	0.61 ± 0.02	0.62
	5	0.84 ± 0.03	0.70 ± 0.04	0.70 ± 0.02	0.68 ± 0.03	0.69
	7	0.79 ± 0.04	0.80 ± 0.05	0.76 ± 0.03	0.75 ± 0.04	0.77
$\mathbf{D}_{AS}$	3	0.85 ± 0.05	0.65 ± 0.1	0.66 ± 0.06	0.64 ± 0.07	0.65
	5	0.85 ± 0.02	0.72 ± 0.05	0.71 ± 0.03	0.70 ± 0.03	0.71
	7	0.82 ± 0.04	0.77 ± 0.07	0.73 ± 0.06	0.73 ± 0.06	0.74
$\mathbf{D}_{OS}$	3	0.86 ± 0.02	0.62 ± 0.1	0.63 ± 0.07	0.61 ± 0.08	0.62
	5	0.85 ± 0.01	0.71 ± 0.06	0.70 ± 0.04	0.69 ± 0.04	0.70
	7	0.80 ± 0.02	0.77 ± 0.05	0.75 ± 0.04	0.73 ± 0.04	0.75

Table 6: Accuracy and robustness of the models trained in the different scenarios.

Conclusion

In this work, we show that properly distributed ML instantiations achieve across-the-board improvements in accuracy-robustness tradeoffs against state-of-the-art transfer-based attacks that could otherwise not be realized by the current ensemble or federated learning instantiations. Our results suggest that increasing the number of nodes and diversifying hyperparameters, such as the learning rate, the momentum and weight decay in distributed ML deployments are important in increasing overall robustness against transfer attacks. Surprisingly, our results suggest that other aspects, such as diverse architectures, optimizers, and schedulers, have little impact on robustness. We therefore hope that our findings motivate further research in this fascinating area.

Acknowledgments

This work has been co-funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, by the German Federal Ministry of Education and Research (BMBF) through the project TRAIN (01IS23027A), and by the European Commission through the HORIZON-JU-SNS-2022 ACROSS project (101097122). Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

Andreina, S.; Marson, G. A.; Möllering, H.; and Karame, G. 2021. BaFFLe: Backdoor Detection via Feedback-based Federated Learning. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, 852–863. IEEE.

Andreina, S.; Zimmer, P.; and Karame, G. 2024. On the Robustness of Distributed Machine Learning against Transfer Attacks. arXiv:2412.14080.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How To Backdoor Federated Learning. In Chiappa, S.; and Calandra, R., eds., *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, 2938–2948. PMLR.

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Srndic, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion Attacks against Machine Learning at Test Time. In *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, 387–402. Springer.

Bryniarski, O.; Hingun, N.; Pachuca, P.; Wang, V.; and Carlini, N. 2022. Evading Adversarial Example Detection Defenses with Orthogonal Projected Gradient Descent. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Byun, J.; Cho, S.; Kwon, M.-J.; Kim, H.-S.; and Kim, C. 2022. Improving the Transferability of Targeted Adversarial Examples Through Object-Based Diverse Input. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15244–15253.

Chellapandi, V. P.; Yuan, L.; Brinton, C. G.; Żak, S. H.; and Wang, Z. 2024. Federated Learning for Connected and Automated Vehicles: A Survey of Existing Approaches and Challenges. *IEEE Transactions on Intelligent Vehicles*, 9(1): 119–137.

Chen, H.; Zhang, Y.; Dong, Y.; Yang, X.; Su, H.; and Zhu, J. 2024. Rethinking Model Ensemble in Transfer-based Adversarial Attacks. In *The Twelfth International Conference on Learning Representations*.

Demontis, A.; Melis, M.; Pintor, M.; Jagielski, M.; Biggio, B.; Oprea, A.; Nita-Rotaru, C.; and Roli, F. 2019. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In Heninger, N.; and Traynor, P., eds., *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, 321–338. USENIX Association.

Dhasade, A.; Kermarrec, A.; Pires, R.; Sharma, R.; and Vujasinovic, M. 2023. Decentralized Learning Made Easy with DecentralizePy. In Yoneki, E.; and Nardi, L., eds., *Proceedings of the 3rd Workshop on Machine Learning and Systems, EuroMLSys 2023, Rome, Italy, 8 May 2023*, 34–41. ACM.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting Adversarial Attacks With Momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gao, L.; Cheng, Y.; Zhang, Q.; Xu, X.; and Song, J. 2021. Feature Space Targeted Attacks by Statistic Alignment. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 671–677. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Guo, Y.; Li, Q.; and Chen, H. 2020. Backpropagating Linearly Improves Transferability of Adversarial Examples. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 85–95. Curran Associates, Inc.

Huang, Y.; and Kong, A. W.-K. 2022. Transferable Adversarial Attack based on Integrated Gradients. In *International Conference on Learning Representations*.

Inkawhich, N.; Liang, K. J.; Wang, B.; Inkawhich, M.; Carin, L.; and Chen, Y. 2020. Perturbing Across the Feature Hierarchy to Improve Standard and Strict Blackbox Attack Transferability. In *NeurIPS*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

Kurakin, A.; Goodfellow, I.; Bengio, S.; Dong, Y.; Liao, F.; Liang, M.; Pang, T.; Zhu, J.; Hu, X.; Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; Yuille, A.; Huang, S.; Zhao, Y.; Zhao, Y.; Han, Z.; Long, J.; Berdibekov, Y.; Akiba, T.; Tokui, S.; and Abe, M. 2018. Adversarial Attacks and Defences Competition. arXiv:1804.00097.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.; Zhang, W.; and Liu, J. 2017. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 5330–5340.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv preprint arXiv:1807.05118*.

Liu, Y.; Kang, Y.; Zou, T.; Pu, Y.; He, Y.; Ye, X.; Ouyang, Y.; Zhang, Y.-Q.; and Yang, Q. 2024. Vertical Federated Learning: Concepts, Advances, and Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3615–3634.

Mahmood, K.; Gurevin, D.; van Dijk, M.; and Nguyen, P. H. 2020. Beware the Black-Box: on the Robustness of Recent Defenses to Adversarial Examples.

Mao, Y.; Fu, C.; Wang, S.; Ji, S.; Zhang, X.; Liu, Z.; Zhou, J.; Liu, A. X.; Beyah, R.; and Wang, T. 2022. Transfer Attacks Revisited: A Large-Scale Empirical Study in Real Computer Vision Settings. In *2022 IEEE Symposium on Security and Privacy (SP)*, 1423–1439.

Naseer, M.; Khan, S.; Hayat, M.; Khan, F. S.; and Porikli, F. 2021. On Generating Transferable Targeted Perturbations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 7708–7717.

Pang, T.; Xu, K.; Du, C.; Chen, N.; and Zhu, J. 2019a. Improving Adversarial Robustness via Promoting Ensemble Diversity. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 4970–4979. PMLR.

Pang, T.; Xu, K.; Du, C.; Chen, N.; and Zhu, J. 2019b. Improving Adversarial Robustness via Promoting Ensemble Diversity. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 4970–4979. PMLR.

Paulik, M.; Seigel, M.; Mason, H.; Telaar, D.; Kluivers, J.; van Dalen, R.; Lau, C. W.; Carlson, L.; Granqvist, F.; Vandevelde, C.; Agarwal, S.; Freudiger, J.; Byde, A.; Bhowmick, A.; Kapoor, G.; Beaumont, S.; Áine Cahill; Hughes, D.; Javidbakht, O.; Dong, F.; Rishi, R.; and Hung, S. 2021. Federated Evaluation and Tuning for On-Device Personalization: System Design / Applications. arXiv:2102.08503.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR (Poster)*.

Wang, X.; and He, K. 2021. Enhancing the Transferability of Adversarial Attacks Through Variance Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1924–1933.

Wang, X.; He, X.; Wang, J.; and He, K. 2021. Admix: Enhancing the Transferability of Adversarial Attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 16158–16167.

Wu, W.; Su, Y.; Lyu, M. R.; and King, I. 2021. Improving the Transferability of Adversarial Samples With Adversarial Transformations. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 9024–9033. Computer Vision Foundation / IEEE.

Xie, C.; Zhang, Z.; Zhou, Y.; Bai, S.; Wang, J.; Ren, Z.; and Yuille, A. L. 2019. Improving Transferability of Adversarial Examples With Input Diversity. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2730–2739. Computer Vision Foundation / IEEE.

Zhang, J.; Li, B.; Chen, C.; Lyu, L.; Wu, S.; Ding, S.; and Wu, C. 2023a. Delving into the Adversarial Robustness of Federated Learning. In Williams, B.; Chen, Y.; and Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 11245–11253. AAAI Press.

Zhang, J.; Wu, W.; Huang, J.-t.; Huang, Y.; Wang, W.; Su, Y.; and Lyu, M. R. 2022. Improving Adversarial Transferability via Neuron Attribution-Based Attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14993–15002.

Zhang, Y.; Ramage, D.; Xu, Z.; Zhang, Y.; Zhai, S.; and Kairouz, P. 2023b. Private Federated Learning in Gboard. arXiv:2306.14793.