3

Federated Learning: Privacy, Security and Hardware Perspectives

Taha Yassine Abidi, Iyad Dayoub, Elhadj Doguech, and Ihsen Alouani

Université Polytechnique Hauts-De-France, France

Abstract

Machine Learning (ML) models are being deployed in a wide range of domains owing to their capacity to deliver high performance across a range of challenging tasks including safety-critical and privacy-sensitive applications. Moreover, the computing requirements of increasingly complex ML models presents a significant challenge to the hardware industry.

Against this backdrop, Federated Learning (FL) has emerged as a promising technique that enables privacy-preserving development of ML models on low-energy Edge devices. FL is a distributed approach that enables learning from data belonging to multiple participants, without compromising privacy since user data are never directly shared. Instead, FL relies on training a global model by aggregating knowledge from local models. Despite its reputation as a privacy-enhancing strategy, recent studies reveal its susceptibility to sophisticated attacks that can undermine integrity and, as well as disrupt their operations. Notably, the constraints posed by the limited hardware resources in edge devices compound these challenges. Gaining insight into these potential risks and exploring hardware-friendly solutions is vital for effectively implementing trustworthy and power-efficient FL systems in edge environments.

This chapter contributes a review and perspective of the triad of privacy, security, and hardware optimization in FL settings.

Keywords: Federated Learning, Hardware Optimisation, ML Security, Privacy.

3.1 Introduction and Background

In this era of unprecedented data proliferation and exponential technological advancement, conventional centralized and cloud-based training and deployment of machine learning faces 2 main challenges:

- How to train and deploy accurate models in an energy-efficient and sustainable manner?
- How to guarantee the security and privacy of potentially sensitive data without compromising the learning process?

FL has emerged as a promising approach to address the challenges posed by decentralised data sources while preserving data privacy. Traditional centralised ML approaches require aggregating sensitive data from various sources into a central repository for training, which can raise concerns about data exposure and privacy. FL offers an innovative solution by enabling model training across multiple devices or data silos, without the need to centralise the data themselves. This distributed approach not only safeguards individual privacy but also optimises the utilisation of edge devices, edge servers, and cloud resources.

The key motivation behind FLis to leverage the collective intelligence of a network of devices while maintaining data locality. This is particularly crucial in scenarios where data is distributed across devices or locations, such as Internet of Things (IoT) ecosystems, healthcare networks, and financial institutions. By allowing devices to collaboratively learn a shared model while keeping their data local, federated learning can address challenges like network latency, bandwidth limitations, and data security.

In this chapter, we delve into the multifaceted aspects of FL, focusing on privacy, security, and the opportunities for hardware optimisation at the Edge. We explore the techniques that enable data privacy within FL, the security measures needed to protect against adversarial attacks, and the ways in which hardware constraints and advancements shape the landscape of FL. Through case studies and emerging trends, we aim to provide a comprehensive understanding of how federated learning empowers data-driven insight

while upholding individual privacy, ensuring security, and harnessing the potential of diverse hardware resources.

This chapter not only sheds light on the current state of federated learning but also serves as a guide for researchers, practitioners, and policymakers who seek to navigate the intersection of machine learning, distributed systems, and data governance. As FL continues to evolve, it is imperative to appreciate its significance in reshaping the landscape of data-driven technologies, fostering collaboration, and advancing both technological and ethical dimensions in the digital era.

The structure of the chapter is crafted to offer a comprehensive exploration of the FL state-of-the-art. Our roadmap unfolds as follows: we begin with an initial introduction to the basics of FL and its applications, followed by an exploration of FL's constraints and limitations, including hardware resources, security, and privacy considerations. Finally, we conclude by underscoring the crucial requirement for balance among these varied aspects.

3.2 Federated Learning Overview

Training a deep neural network necessitates a significant amount of data, often representing the most valuable resource within a target environment: it can be of commercial value, be governed by privacy regulations, can be limited by user agreements (as illustrated by regulations such as HIPAA in the US and GDPR in Europe). In another scenario, data generated on Edge devices may face sharing restrictions due to privacy anxieties, bandwidth restrictions, or performance constraints.

FL recently emerged as a potential solution to the problems above. It enables participants to collaboratively train a federated model while preserving local data privacy. Within the FL framework, each participant trains a local model sharing it with a central server also known as a central aggregator. Data remain private to each participant. The server aggregates the local model updates into a single federated model and shares this model with the participants, creating an updated federated model that benefits from all the data without jeopardising its confidentiality. The model's refinement continues as participants deliver more updates.

FL encompasses three primary categories from a data partitioning perspective: horizontal FL, vertical FL and federated transfer learning [5]. This document, however, zooms on the most prevalent and widely used model, namely horizontal FL. In the subsequent section, we consider the intricacies

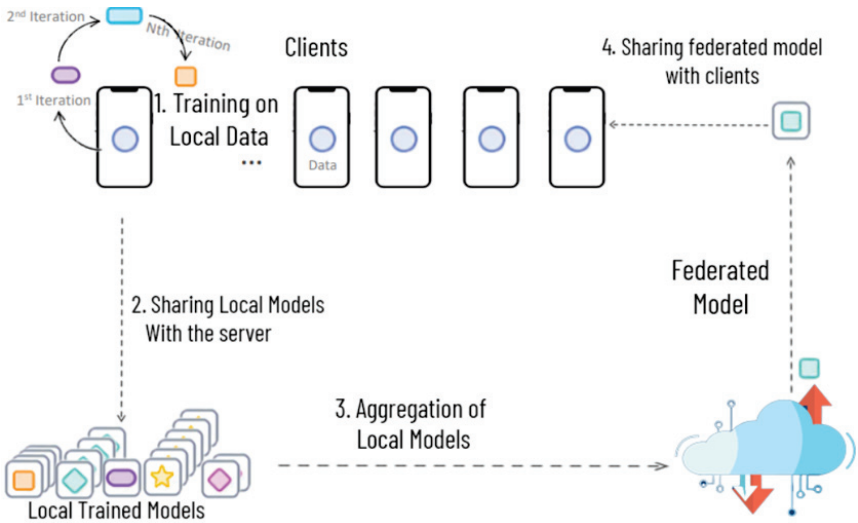

Figure 3.1 Client device sends their locally trained model updates to server for training the federated model.

of horizontal FL while also offering succinct insights into the other two models for context.

3.2.1 Horizontal Federated Learning

Participants train their local model with data that are in the same feature space. For example, two regional hospitals might contain different patient population data, with little to no intersection in the data (perhaps because the hospitals serve different regions). However, the activities of the two hospitals are similar with respect to each other and so their feature spaces are the same. During the training phase of the horizontal FL model, each of the participants trains its local model using the local, private, data and sends the gradients to the central aggregator. The central aggregator aggregates all of those local model updates to build a global shared model and return this back to all participants. Finally, each participant updates its local model using the result from the central aggregator.

3.2.2 Vertical Federated Learning

Vertical FL addresses the scenario where participants refine their respective local models using data samples derived from different feature spaces. For

example, consider a hospital and a pharmacy in the same region. While there is likely a significant overlap in patient population data, the retained information (i.e., the features) for these patients vary due to the distinct functions of the two participants. For example, the hospital preserves the records of all users about their disease, diagnosis and information of treatment received while the pharmacy keeps the records of medicine purchasing history. Using a vertical FL system, the two institutions can collaboratively build a prediction model by aggregating those different features and calculating the gradients of their local data in a privacy preserving manner.

3.2.3 Federated Transfer Learning

Federated Transfer Learning [7] finds its niche in scenarios where users' datasets remain disjoint or share minimal overlap in both the samples and feature spaces. For instance, revisiting the hospital and pharmacy scenario recall that the feature spaces of their data have little overlap. If the two institutions are in different countries they would also have few, if any, common patients, making it impossible to apply VFL. FTL solves this problem by creating a common representation using transfer learning and using it to build a predictive model across the entire data set.

With a foundational understanding of FL in place, we now turn our attention to the challenges that accompany this paradigm. While FL offers a promising avenue for decentralised model training and data privacy preservation, it is essential to acknowledge its limitations.

3.3 Challenges and Limitations of Federated Learning

FL presents a trio of critical challenges that demand rigorous exploration: hardware resources, security, and privacy.

These dimensions shape the framework's efficacy and ethical underpinnings. In this section, we consider this interplay.

3.3.1 Security challenge

The distributed nature of FL, while preserving data locality, introduces complexities that require careful attention to ensure the confidentiality, integrity, and authenticity of the data and models being exchanged.

Adversarial attacks, stemming from both malicious clients and malicious servers, pose a significant threat to the security and integrity of FL by

exploiting vulnerabilities inherent in the decentralised nature of the approach. These attacks aim to manipulate the training process and the resulting model's composition, leading to erroneous predictions and potential data exposure.

In this context, understanding the objectives that potential attackers pursue becomes crucial. These objectives can be categorised into three primary dimensions:

- Compromising System Integrity: Attackers aim to compromise the integrity of the FL system by tampering with the model's function. They induce misclassifications by poisoning individual local model updates or by colluding with other malicious participants.
- Compromising Data Confidentiality and Privacy: Adversaries target data confidentiality and privacy by attempting to infer private information or reconstruct original training samples. We will delve deeper into this topic in the upcoming section.
- Disrupting the Learning Process: Attackers seek to disrupt the learning process itself. This includes tactics such as initiating denial-of-service or impeding the convergence of the training process.

To achieve these objectives, adversaries deploy a range of strategic actions:

- Poisoning Attacks: Malicious actors maliciously alter either the training data or the model to corrupt the overall federated model's integrity. This compromise is executed with the intention of manipulating the model's behaviour to serve the attacker's motives.
- Privacy Attacks: Adversaries attempt to deduce sensitive information about the data, which will be discussed in detail in the subsequent section.
- Disruption Attacks: Attackers exploit the learning process by introducing delays in updates or interfering with the protocol's operation, aiming to undermine the system's functionality.

3.3.1.1 Malicious Clients

We first consider model integrity attacks that originate from malicious clients. We assume that a client is able to arbitrarily change its local model that it sends to the server. The model can be manipulated either directly by changing its parameters, or indirectly by manipulating the local training set. The poisoned local model in turn poisons the aggregated model when it is combined with the models from other clients. One possible goal of this

attack is to make the global model misclassify in general (untargeted attack). Alternatively, the attack can target specific classes that the attacker would like to degrade, potentially causing them to misclassify into specific alternative labels (targeted attack).

In targeted attacks, the attacker aims at forcing the model to misclassify a specific class or subset of classes. These attacks are also called Backdoor attacks. For example, an attacker may desire to have a particular type of vehicle be undetected in a federated recognition system. Targeted attacks can be performed either by manipulating the target model's parameter or by poisoning the target training data directly.

Targeted Model and Data Poisoning
Researchers have investigated model poisoning techniques aimed at crafting targeted attacks, where the adversary's goal is to create a global model that exhibits high accuracy for both the primary task (untargeted classes) and includes a hidden backdoor to target specific classes.

Attackers can attempt to disrupt the accuracy of the FL global model through three avenues in data poisoning:

- Mislabelling Data: The adversary can change the labels of training samples, converting them to a target class while keeping the data otherwise unaltered [9]–[11]. These attacks are demonstrated by Biggio et al. [12], Fung et al. [9], and Gu et al. [10].
- Manipulating Input Features: By slightly modifying a portion of the original training dataset through noise addition or feature manipulation, adversaries can make models learn triggers on specific inputs while maintaining non-poisoned data accuracy [3], [13].
- Combining Mislabelling and Feature Manipulation: This category involves malicious clients changing both data and labels. The attacker can induce the global model to trigger on specific inputs and misclassify to a designated target label. An example is an attacker's face being misclassified by a federated face recognition system while a specific watermark is present in the image. Naseri et al. [14] demonstrate this through a modification of training data and label of samples.

3.3.1.2 Mitigating client-based attacks

Defences can be organised into two primary categories: Detection and removal of malicious client updates; and mitigating attack severity. We discuss both of these categories below. Detection and removal of malicious client updates.

Detection and Removal of Malicious Client Updates

Detecting and removing malicious client updates involves strategies that flag unusual and statistically abnormal updates, excluding them from the aggregated model. These defences vary in how they decide if an update is abnormal, usually by comparing it to the distribution of updates from other clients. A balance exists between accommodating unique data contributions from clients while identifying and preventing harmful updates. This balance entails allowing valuable data to contribute while guarding against malicious intentions. Shejwalkar et al. [15] introduced a strategy called divide-and-conquer (DnC) to tackle malicious model poisoning updates. DnC works under the assumption that a harmful update from a malicious source will significantly deviate from normal updates. Initially, DnC calculates the main direction of variance among input updates, known as the principal component. It then computes projections, which are essentially measures of how much the updates align with this principal component. Harmful updates tend to have larger projections. In the final step, DnC removes a portion of updates with the highest projections. This approach is effective against untargeted attacks, as long as the number of malicious clients doesn't surpass the proportion of removed updates.

Shejwalkar et al. [15] introduced a strategy called divide-and-conquer (DnC) to tackle malicious model poisoning updates. DnC works under the assumption that a harmful update from a malicious source will significantly deviate from normal updates, causing harm. Initially, DnC calculates the main direction of variance among input updates, known as the principal component. It then computes projections, which are essentially measures of how much the updates align with this principal component. Harmful updates tend to have larger projections. In the final step, DnC removes a portion of updates with the highest projections. This approach is effective against untargeted ICM attacks, if the number of malicious clients doesn't surpass the proportion of removed updates.

Mitigating the severity of the attack

In this second category, defences leverage aggregation strategies that do not exclude the malicious updates, but rather try to mitigate their effect. One strategy involves using the median as a point of aggregation for models, effectively lessening the influence of malicious outliers within FL systems [16].

Fu et al. [12] introduce an innovative aggregation algorithm termed "Reweighting" to counter targeted poisoning attacks. In their approach, the

global model is a reweighted average of individual local models. This is achieved through techniques such as the Repeated Median Estimator [17] and Iteratively Reweighted Least Squares (IRLS) [18]. In practical terms, the authors assess the confidence of model parameters based on their distance from a robust regression line. Local models are then assigned weights proportional to their parameter confidence. Malicious outliers, having lower confidence scores, exert minimal influence on the overall model, effectively curtailing their impact.

3.3.1.3 Malicious Server attacks and mitigations

The central server's role within the context of FL is pivotal, encompassing tasks such as aggregating updates into the global model and disseminating it to clients. While the server's integrity is typically assumed, the potential for severe consequences necessitates a nuanced consideration of malicious server attacks and potential countermeasures. In essence, a compromised server holds the capacity to arbitrarily manipulate the global model, leading to detrimental impacts on classifier performance. Hence, comprehending this threat becomes crucial, prompting exploration into potential defence strategies.

Architecting a secure federated training protocol without the presumption of a trusted server presents an intricate and compelling challenge. Without such safeguards, the server's influence on the models sent to clients is unconstrained, allowing malicious servers to dispatch compromised or subpar classifiers. The server's motives could range from intentional harm to clients, such as by distributing models with targeted poisoning, to a desire to leverage client data without reciprocating the effort of model aggregation and communication. In the baseline FL framework, clients implicitly bestow trust in the server and accept its model as the global reference, devoid of means to verify if the server adheres to the FL protocol's integrity. A secure federated protocol would ideally impede malicious servers from arbitrarily injecting fake model updates. Alternatively, it would empower clients to validate the integrity of received model updates. Addressing this challenge, Xu et al. [19] propose Verifynet, a verification process that ensures the veracity of server-delivered outcomes. Their approach involves hashing the gradient of the client's local model through a homomorphic hash function possessing universally recognized collision-resistant features. Furthermore, clients compute additional (meta) information utilising pseudorandom functions linked to secret keys issued by a trusted authority (TA). Each client then

dispatches the masked gradient and associated meta information to the server. On the server side, the gradients from all clients are aggregated, negating the added noise. The server subsequently calculates a proof derived from client-provided meta information, broadcasting this proof to active clients. To assess the server update's authenticity, each client scrutinises the proof by verifying the truth equations of homomorphic hash and pseudorandom functions. Any inconsistencies prompt client rejection of the server's result. In essence, Verifynet verifies server results, safeguarding clients against manipulation by a malicious server.

Adversarial attacks exploit the vulnerabilities inherent in the decentralised model, seeking to disrupt the training process and compromise the behaviour of resultant models. The multifaceted challenges brought forth by these attacks underscore the need for innovative defence mechanisms that transcend traditional paradigms.

3.3.2 Privacy challenge

The privacy challenge is marked by the intricate balance between collaborative knowledge extraction and safeguarding individual data privacy. In this section, we explore the complexities surrounding compromised data confidentiality, the prevalence of privacy attacks, and the potential implications of membership inference attacks. The decentralised nature of FL, while fostering collective learning, poses unique challenges to preserving the privacy of individual participants' sensitive information. These challenges necessitate the exploration of innovative strategies and techniques designed to uphold the privacy of participants while maintaining the robustness of collaborative learning.

Imagine a consortium of hospitals employing FL to construct a robust disease prediction model. In this collaborative effort, each hospital contributes patient data with a strong emphasis on preserving individual privacy. Yet, the decentralised nature of FL introduces the potential for privacy breaches. Within this context, a malicious actor could exploit vulnerabilities to deduce sensitive patient information. This exploitation would compromise the confidentiality imperative. Such attacks could result in the unauthorised identification of individuals, thereby jeopardising their privacy and undermining trust in the collaborative strategy.

It's important to recognize that privacy attacks in FL can emanate from various malicious clients and malicious servers. Malicious clients might attempt to infer private information about other clients based on model

updates. A malicious server could exploit model updates to deduce sensitive client information, further underscoring the multifaceted nature of the privacy challenge. In the subsequent sections, we examine specific types of attacks originating from both clients and servers. We will explore techniques to defend against these attacks.

3.3.2.1 Client privacy attacks

This type of attack originates either from a single malicious client or group of colluding clients. For a given client, only its own data and global model are available to them. As in the baseline FL setting, the client trains its local model and communicates the raw gradients to the server without protection (e.g., adding noise or using encryption), it opens up scope for any malicious player to infer private information about other clients' data from the raw gradients. Here we consider two types of attacks. One is an inference attack on a specific client 'overhearing' the local model gradient of other clients. Overhearing might happen directly or through collusion between malicious clients. Another type of attack is to infer sensitive information of other clients through the global model weights. In this second category of attack, a client might maliciously modify its local model parameters to infer sensitive information of other clients.

Membership Inference Attack – Membership inference attacks are a common privacy attack [20], [21]. In this form of attack, the attacking client's goal is to infer whether a specific data sample is part of the dataset that was used to train the federated model. Often, the attacker may know only part of the data, and the attack could also enable them to recover this missing information [21]. With access to aggregate model parameters from the server, Nasr et al. [16] empirically show if a target data point is contained within the client's dataset or not. A malicious client specifically modifies its local model parameters to increase the loss on a target data point X. Then the server receives adversarial parameters from the malicious client and aggregates these parameters with other participants to generate the global model, which is finally transmitted back to the clients. Now, using the aggregated parameters, if the local stochastic gradient descent (SGD) algorithm on the client side abruptly lowers the gradient of the loss on a target data point X, then X is in the training set of a client. Alternatively, if the data point is not included in a client's dataset, the gradient on this point would alter gradually throughout the course of the training.

Property Inference Attack – Property inference is a class of privacy attacks on machine learning models where an attacker attempts to infer properties of the training set overall, rather than individual instances of the data [22], [23]. For example, the attacker may attempt to infer if the environment of most of the data is indoors or outdoors, to identify the proportion of the data from a particular class (e.g., gender or race), or more specifically inferring that whether a certain person is wearing glasses or not in the training data. In conventional machine learning settings, several property inference attacks have been demonstrated. These attacks can also be conducted in an FL setting, on the aggregate model or on individual client models if they are obtained. There are several attack strategies for property inference that arise in FL settings when training the global shared model [18, 19]. For example, Melis et al. [18] created a batch property classifier in a collaborative training (federated) environment. This classifier evaluates whether the server's global updates are based on data that includes or excludes the desired characteristic. The adversary will need many batches of auxiliary data, consisting of data points with and without the property of interest, to carry out the attack. The auxiliary data points must come from the same class as the data from the target client. Using snapshots of the global model the adversary computes two sets of gradients (A and B) based on the batch of data points with the property of interest or without the property of interest. The attacker assigns a positive label to set A and a negative label to set B. They train a binary batch property classifier with those gradients (A and B), which generalises the gradients of future batches of data which are given as input and predicts whether or not they contain the desired property. As a result, without changing anything in the local or global collaborative training approach, the adversary observes the global model and performs a property inference attack on the updates.

3.3.2.2 Mitigating client-based attacks

Moving on to defences against client-originated attacks, we uncover a spectrum of strategies designed to fortify the privacy and security of FL.

Gradient Perturbation with Noise: Exchanging intermediate model updates with the server introduces vulnerabilities to membership inference and property inference attacks. These risks arise from the server or colluding clients inferring private data of honest clients from their raw gradients. To counteract this, differential privacy techniques inject noise into gradients, ensuring privacy-preserving exchanges in FL [24]. Naseri et al. [14] propose Local Differential Privacy (LDP) and Central Differential Privacy (CDP).

LDP applies differential privacy to local models, while CDP implements it centrally, leveraging the server's trust. Both methods mitigate membership and property inference attacks. Adding noise conceals global properties, offering protection against various attacks. Despite enhancing security, differentially private strategies slightly diminish the shared model's utility. Zhu et al. [25] demonstrate that defence efficacy depends more on variance magnitude than noise type (Gaussian or Laplacian). Increased variance harms model accuracy, highlighting a trade-off between privacy and utility.

GAN-based Generated Samples instead of the Original: Deploying generative adversarial networks (GAN) [54] can help to mitigate membership and property inference attack by generating a large amount of samples in the same distribution of the training dataset (Anti-GAN in table 2) to train the model. In the case of Anti-GAN [93], they train the victim's GAN in a way that it learns the classification features rather than learning the visual features of the original images. Then, the generated fake samples from the GAN are mixed with the original images to train the model. Using GAN, this defence obscures the visual features of the clients' training data to defend against this attack. However, it eventually degrades the accuracy of the model [93]. There is also evidence that GANs could also result in additional inference leakage [26] [61].

3.3.2.3 Server based privacy attacks

If a server is malicious, it has full access to the individual client updates/models and can attempt arbitrary inference attacks on them. We describe the possible attacks under this model in this section.

Deep Leakage from Gradients (DLG): Deep Leakage from Gradients (DLG) is an attack in the context of FL that focuses on exploiting vulnerabilities arising from the exchange of intermediate model updates between clients and a central server. This attack is particularly concerned with revealing private information and properties of individual training data instances by analysing the gradients of the local models used in the learning process. In the DLG attack, a malicious entity, whether it is a client or a colluding group of clients, aims to infer sensitive details about other clients' training data from the gradients of their local models. The core idea behind this attack is that the gradients of the local models contain information about the individual training samples they were trained on. These gradients, when exchanged with the server as part of the FL process, can leak information about the underlying data distribution and specific data instances.

The attack's mechanism involves carefully analysing the gradients to identify patterns, correlations, or unique features that correspond to specific data points. By reverse-engineering these gradients, attackers can deduce sensitive information about other clients' data, compromising data privacy and confidentiality. Deep Leakage from Gradients can lead to privacy breaches, property inference, and membership inference attacks, as attackers exploit the inherent information present in gradients to gain insights into the dataset without directly accessing the raw data.

Mitigating Gradient Leakage Attacks:
The primary mitigation strategy against DLG is to mask the gradients of the clients such that they are not exposed to the server. A number of different ideas to mask gradients have been proposed, like single masking[25], double masking[19].

Single masking is an approach that introduces controlled noise into the gradients before they are sent to the server. This noise acts as a protective layer, making it difficult for the server to extract sensitive information from the gradients. The key idea is to obfuscate the gradients in a way that preserves the model's learning progress while reducing the risk of information leakage. Single masking adds randomness to the gradients, making them less susceptible to reverse-engineering by malicious actors.

Double masking, on the other hand, takes the concept of gradient masking a step further. In this approach, not only are the gradients masked before transmission to the server, but they are also further masked at the server's end before aggregation. This double-layered masking provides an additional level of security by ensuring that the server itself cannot access the original gradients contributed by individual clients. This way, even if the server was compromised, the information contained in the gradients remains protected.

Both single masking and double masking contribute to thwarting DLG attacks by minimising the potential leakage of sensitive information through the gradients. These techniques underline the efforts to strike a balance between collaborative model training and preserving the privacy of clients' data in the FL setting.

Our exploration of the multifaceted challenges in the realm of FL highlights the intricate interplay between hardware constraints, security vulnerabilities, and privacy concerns. We've delved into the limitations imposed by resource-constrained devices, where the balance between model complexity and hardware capabilities becomes a critical factor. The security landscape of

FL, encompassing adversarial attacks from both malicious clients and servers, underscores the imperative to fortify the integrity and authenticity of collaborative learning processes. Moreover, our investigation into privacy challenges reveals the significance of protecting sensitive data while maintaining the efficacy of FL.

3.3.3 Hardware constraint and opportunities

The deployment of AI at the Edge has the potential to transform industries and facilitate personalised products, which largely hinges on its ability to harness the data from ubiquitous devices spanning from smartphones to Internet of Things (IoT) devices. Yet, the energy and resource limitations inherent in these devices pose significant obstacles. Edge devices and embedded systems operate under stringent energy budgets and have constrained computational capabilities. These devices lack the computational capacities of data centres, making resource-intensive ML a challenge.

In this section, we delve into the implications of Edge devices' hardware limitations on FL. We also discuss the opportunities that can emerge from new computing paradigms such as approximate computing on FL security. FL processes that demand substantial computational power and memory can strain these devices, potentially leading to increased latency, reduced model quality, and even device overheating.

Striking a balance between model complexity and the limitations of these hardware resources becomes a critical consideration, calling for innovative model architectures and optimization techniques that can maintain model performance while respecting the resource boundaries of edge and embedded devices.

To address these challenges, researchers and practitioners have explored a range of optimization techniques that enhance the efficiency of FL processes. Quantization[8], for instance, involves representing model parameters with reduced precision, effectively reducing the memory footprint and communication overhead during updates. Model compression techniques focus on minimising the model's size while preserving its predictive capabilities, enabling faster training and less demanding communication. In particular, in-model compression techniques aim to design models that inherently require fewer computations, thereby reducing energy consumption and resource usage. One notable approach in this direction is approximate computing, where local clients introduce controlled inaccuracies into the computations, trading off precision for efficiency [30, 31]. This innovative strategy approach

aligns well with the resource-constrained environment of edge devices, allowing them to perform computations more efficiently in terms of both resources and energy consumption.

The underlying principle of approximate computing stems from the observation that not all tasks require highly precise computational precision to achieve satisfactory overall results. By allowing local clients to perform computations with reduced precision, such as using fewer bits for numerical representation, devices can significantly lower their computational and energy requirements.

A wide range of approximation techniques across all layers of the computing stack have been proposed; these techniques leverage the inherent error tolerance of ML architectures to achieve improvements in inference efficiency (e.g., power consumption and resource utilisation) [32].

The main categories of approximation techniques explored previously are as follows:

- Algorithmic level: This mainly includes Quantization, Pruning and Model Compression. Quantization approximates the model by reducing the number of bits used to represent the weights and activation outputs such as Bfloat [33], DLfloat [34], and very recently Graphcore and AMD proposed a new 8-bit floating-point standard for AI [35]. On the other hand, pruning and model compression try to reduce model size by skipping connections through forcing weights to zero. While these techniques achieve promising benefits towards lower complexity ML systems, their impact remains limited since: (i) Quantization is mainly used in convolution layers and other kernels like pooling, activation and normalisation are still dominated by floating-point arithmetic, and (ii) Pruning often results in irregular computation and memory access patterns and hence have little to no impact on hardware accelerator performance.
- Circuit level: This category focuses on the computing building blocks of the models; Approximate circuits implement core functions to build approximate ML systems to leverage maximum benefits. More specifically, the core arithmetic functions (multiplication, addition and non-linear activation) are either replaced by lower resource approximate designs [36, 37], or more generically by undervolting the circuit to inject random computational errors. An example is shown in Figure 3.2, which corresponds to a circuit implementation of a full adder. Using this logical **approximate** building block to design a multiplier or an

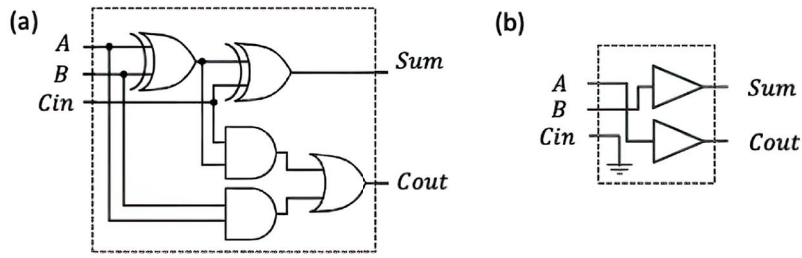

Figure 3.2 Logic diagram of (a) exact Full Adder, (b) Approximate full adder.

adder results in approximate arithmetic elements. These techniques have a high impact on models power consumption and offer a bottom-up approach to overcome the models scalability problem for ML hardware accelerators.

Approximate Computing (AC) as a defense – Recent work [37, 38] has shown that, perhaps surprisingly, implementing ML models using AC can provide substantial robustness against adversarial attacks while reducing the complexity of the implementation. In particular, it has been shown that using approximation during inference introduces robustness against both black-box and white-box adversarial attacks. For example Figure 3.3 shows the classification accuracy of the exact (conventional) model and approximate models for 3 different benchmarks, namely: LeNet-5, AlexNet, and ResNet-18 CNNs under adversarial attack, while varying the adversarial attack magnitude. Specifically, the figure shows the robustness against PGD adversarial attack, where the approximate model achieves the highest accuracy: about 88% for LeNet-5, 81% for AlexNet and 67% ResNet-18.

However, these results are empirical, for specific post-hoc approximation structures and many questions remain. For example, it is not clear whether

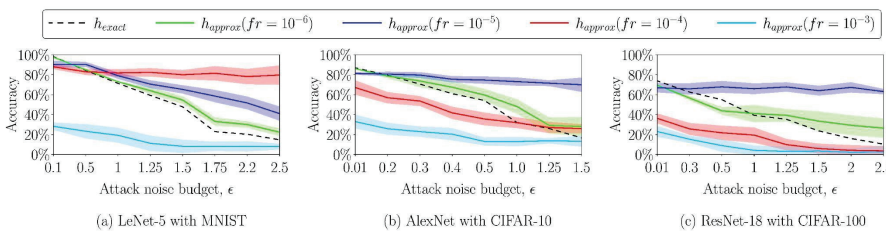

Figure 3.3 Precise and approximate models robustness under PGD attack.

the robustness advantage demonstrated against existing attacks would persist against adaptive attacks. It is also not clear what approximation structures and functions would provide best gains, and how and where to apply approximation to find effective solutions that balance accuracy, robustness to adversarial attacks, and implementation efficiency.

3.4 Conclusion

In that chapter, we considered the state of FL, spanning hardware limitations, security vulnerabilities, and privacy considerations.

We briefly discussed the vulnerabilities posed by adversarial attacks, originating from both malicious clients and servers, on the lights of the attacker's objectives and strategies. From a privacy perspective, while FL had been branded as a privacy-preserving technology, we discussed the challenges arising from potential inference attacks that could leak sensitive information during the collaborative learning process.

The main challenge towards developing accurate ML models at the Edge was the limited energy and hardware resources of Embedded and Edge devices. While the community had explored the use of emerging paradigms such as approximate computing to address this challenge, we believed that the deployment of approximate AI designs (i.e., based on approximate computing engines) might have significant gains from a security and privacy perspective, in addition to the by-product gain in terms of energy consumption.

Acknowledgements

This research was conducted as part of the EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing" project, which has received funding from KDT JU under grant agreement No 101097300. The KDT JU receives support from the European Union's Horizon Europe research and innovation program and Austria, Belgium, France, Greece, Italy, Latvia, Luxembourg, Netherlands, and Norway.

References

[1] K. Bonawitz *et al.*, "Towards Federated Learning at Scale: System Design". arXiv, 22 mars 2019. Available at: https://doi.org/10.48550/arXiv.1902.01046

[2] P. Kairouz *et al.*, “Advances and Open Problems in Federated Learning”. arXiv, 8 mars 2021. Available at: https://doi.org/10.48550/arXiv.1912.04977

[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, et D. Bacon, “Federated Learning: Strategies for Improving Communication Efficiency”. arXiv, 30 October 2017. Available at: https://doi.org/10.48550/arXiv.1610.05492

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data”. arXiv, 26 janvier 2023. Available at: https://doi.org/10.48550/arXiv.1602.05629

[5] Q. Yang, Y. Liu, T. Chen, et Y. Tong, “Federated Machine Learning: Concept and Applications”. arXiv, 13 février 2019. Available at: https://doi.org/10.48550/arXiv.1902.04885

[6] Y. Liu *et al.*, “A Communication Efficient Collaborative Learning Framework for Distributed Features”. arXiv, 31 july 2020. Available at: https://doi.org/10.48550/arXiv.1912.11187

[7] Y. Liu, Y. Kang, C. Xing, T. Chen, et Q. Yang, “Secure Federated Transfer Learning”, *IEEE Intell. Syst.*, vol. 35, n^o 4, p. 70-82, juill. 2020, Available at: https://doi.org/10.1109/MIS.2020.2988525

[8] K. Gupta, M. Fournarakis, M. Reisser, C. Louizos, et M. Nagel, “Quantization Robust Federated Learning for Efficient Inference on Heterogeneous Devices”. arXiv, 22 juin 2022. Consulté le: 31 août 2023. Available at: http://arxiv.org/abs/2206.10844

[9] C. Fung, C. J. M. Yoon, et I. Beschastnikh, “Mitigating Sybils in Federated Learning Poisoning”. arXiv, 15 july 2020. Available at: https://doi.org/10.48550/arXiv.1808.04866

[10] T. Gu, B. Dolan-Gavitt, et S. Garg, “BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain”. arXiv, 11 mars 2019. Available at: https://doi.org/10.48550/arXiv.1708.06733

[11] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, et J. Stainer, “Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent”, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Consulté le: 30 août 2023. Available at: https://papers.nips.cc/paper_files/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html

[12] B. Biggio *et al.*, “Evasion Attacks against Machine Learning at Test Time”, 2013, p. 387-402. Available at: https://doi.org/10.1007/978-3-642-40994-3_25

[13] H. Wang *et al.*, “Attack of the Tails: Yes, You Really Can Backdoor Federated Learning”. arXiv, 9 july 2020. Available at: https://doi.org/10.48550/arXiv.2007.05084

[14] M. Naseri, J. Hayes, and E. De Cristofaro, “Local and Central Differential Privacy for Robustness and Privacy in Federated Learning”. arXiv, 27 May 2022. Available at: https://doi.org/10.48550/arXiv.2009.03561

[15] V. Shejwalkar and A. Houmansadr, “Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning”, *NDSS Symposium*. Available at: https://www.ndss-symposium.org/wp-content/uploads/ndss2021_6C-3_24498_paper.pdf

[16] S. Fu, C. Xie, B. Li, and Q. Chen, “Attack-Resistant Federated Learning with Residual-based Reweighting”. arXiv, 8 janvier 2021. Available at: https://doi.org/10.48550/arXiv.1912.11464

[17] A. F. Siegel, “Robust regression using repeated medians”, *Biometrika*, vol. 69, n^o 1, p. 242-244, avr. 1982. Available at: https://doi.org/10.1093/biomet/69.1.242

[18] P. W. Holland et R. E. Welsch, “Robust regression using iteratively reweighted least-squares”, *Commun. Stat. - Theory Methods*, vol. 6, n^o 9, p. 813-827, janv. 1977, doi: 10.1080/03610927708827533.

[19] G. Xu, H. Li, S. Liu, K. Yang and X. Lin, “VerifyNet: Secure and Verifiable Federated Learning,” in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 911-926, 2020. Available at: https://doi.org/10.1109/TIFS.2019.2929409

[20] M. Nasr, R. Shokri, et A. Houmansadr, “Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning”, in *2019 IEEE Symposium on Security and Privacy (SP)*, mai 2019, p. 739-753. Available at: https://doi.org/10.1109/SP.2019.00065

[21] R. Shokri, M. Stronati, C. Song, et V. Shmatikov, “Membership Inference Attacks against Machine Learning Models”. arXiv, 31 mars 2017. Available at: https://doi.org/10.48550/arXiv.1610.05820

[22] L. Melis, C. Song, E. De Cristofaro, et V. Shmatikov, “Exploiting Unintended Feature Leakage in Collaborative Learning”. arXiv, 1 november 2018. Available at: https://doi.org/10.48550/arXiv.1805.04049

[23] B. Hitaj, G. Ateniese, et F. Perez-Cruz, “Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning”. arXiv, 14 september 2017. Available at: https://doi.org/10.48550/arXiv.1702.07464

[24] W. Li *et al.*, “Privacy-preserving Federated Brain Tumour Segmentation”. arXiv, 2 october 2019. Available at: https://doi.org/10.48550/arXiv.1910.00962

[25] L. Zhu, Z. Liu, et S. Han, “Deep Leakage from Gradients”. arXiv, 19 décembre 2019. Available at: https://doi.org/10.48550/arXiv.1906.08935

[26] C. Briggs, Z. Fan, et P. Andras, “Federated learning with hierarchical clustering of local updates to improve training on non-IID data”. arXiv, 6 May 2020. Available at: https://doi.org/10.48550/arXiv.2004.11791

[27] F. Sattler, K.-R. Müller, et W. Samek, “Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints”. arXiv, 4 october 2019. Available at: https://doi.org/10.48550/arXiv.1910.01991

[28] R. C. Geyer, T. Klein, et M. Nabi, “Differentially Private Federated Learning: A Client Level Perspective”. arXiv, 1 mars 2018. Available at: https://doi.org/10.48550/arXiv.1712.07557

[29] H. Chang, V. Shejwalkar, R. Shokri, et A. Houmansadr, “Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer”. arXiv, 24 décembre 2019. Available at: https://doi.org/10.48550/arXiv.1912.11279

[30] A. Guesmi, I. Alouani, M. Baklouti, T. Frikha, M. Abid, and A. Rivenq. 2019. HEAP: A Heterogeneous Approximate Floating-Point Multiplier for Error Tolerant Applications. In Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP ’19). Association for Computing Machinery, New York, NY, USA, 36–42. https://doi.org/10.1145/3339985.3358495

[31] Ali, K.M.A., Alouani, I., El Cadi, A.A., Ouarnoughi, H., Niar, S. (2020). Cross-layer CNN Approximations for Hardware Implementation. In: Rincón, F., Barba, J., So, H., Diniz, P., Caba, J. (eds) Applied Reconfigurable Computing. Architectures, Tools, and Applications. ARC 2020. Lecture Notes in Computer Science(), vol 12083. Springer, Cham.

[32] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. “Approximate Computing and the Quest for Computing Efficiency”. In: Proceedings of the 52nd Annual Design Automation Conference. DAC ’15. San Francisco, California: Association for Computing Machinery, 2015. ISBN: 9781450335201. Available at: https://doi.org/10.1145/2744769.2751163

[33] S. Wang and P. Kanwar. Bfloat16: the secret to high performance on cloud TPUs. Google blog from https://cloud.google.com/blog/products

/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus.2019.

[34] A. Agrawal, S. M. Mueller, B. M. Fleischer, X. Sun, N. Wang, J. Choi, and K. Gopalakrishnan. "DLFloat: A 16-b Floating Point Format Designed for Deep Learning Training and Inference". In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH). 2019, pp. 92–95. Available at: https://doi.org/10.1109/ARITH.2019.00023

[35] B. Noune, P. Jones, D. Justus, D. Masters, and C. Luschi. 8-bit Numerical Formats for Deep Neural Networks. 2022. Available at: https://doi.org/10.48550/ARXIV.2206.02915

[36] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan. "SALSA: Systematic logic synthesis of approximate circuits". In: DAC Design Automation Conference 2012. 2012, pp. 796–801. Available at: https://doi.org/10.1145/2228360.2228504.

[37] A. Guesmi et al. "Defensive approximation: securing CNNs using approximate computing". In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2021, pp. 990–1003.

[38] M. S. Islam, I. Alouani, and K. N. Khasawneh. "Lower Voltage for Higher Security: Using VoltageOverscaling to Secure Deep Neural Networks". In: 2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). 2021