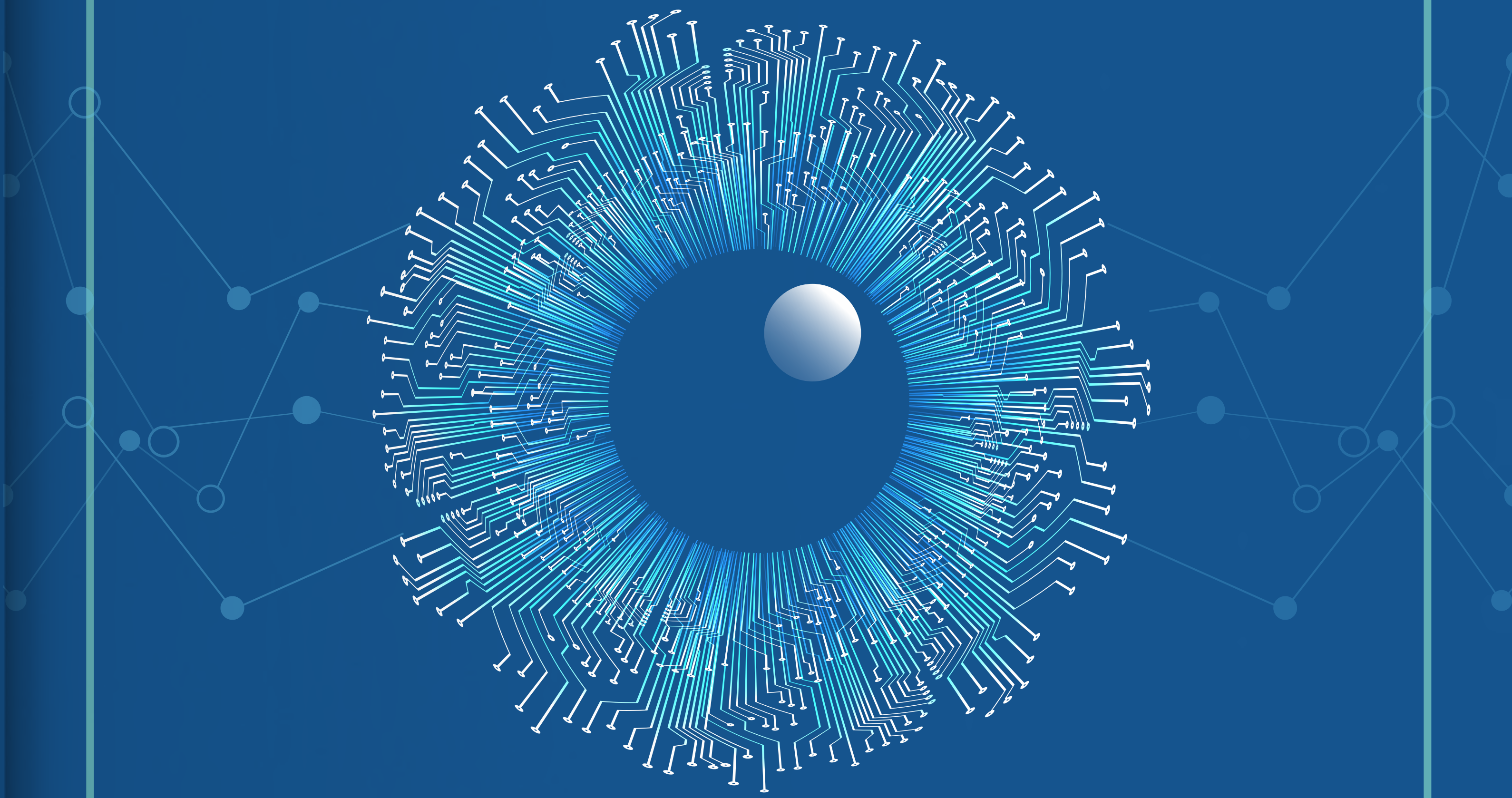
Private AI Collaborative Research Institute

VISION & CHALLENGES OPPORTUNITIES



avast borsetta intel AI

Ahmad-Reza Sadeghi
Ferdinand Brasser
Markus Miettinen
Thien Duc Nguyen
Technical University of Darmstadt

Buse Gül Atli
Florian Kerschbaum
Lachlan J. Gunn
N. Asokan
University of Waterloo and Aalto University

Thomas Given-Wilson
Axel Legay
Université Catholique de Louvain

Rosario Cammarota
Matthias Schunter
Intel Labs

Murali Annaaram
Salman Avestimeh
University of Southern California

Adam Dziedzic
Nicolas Papernot
University of Toronto and Vector Institute

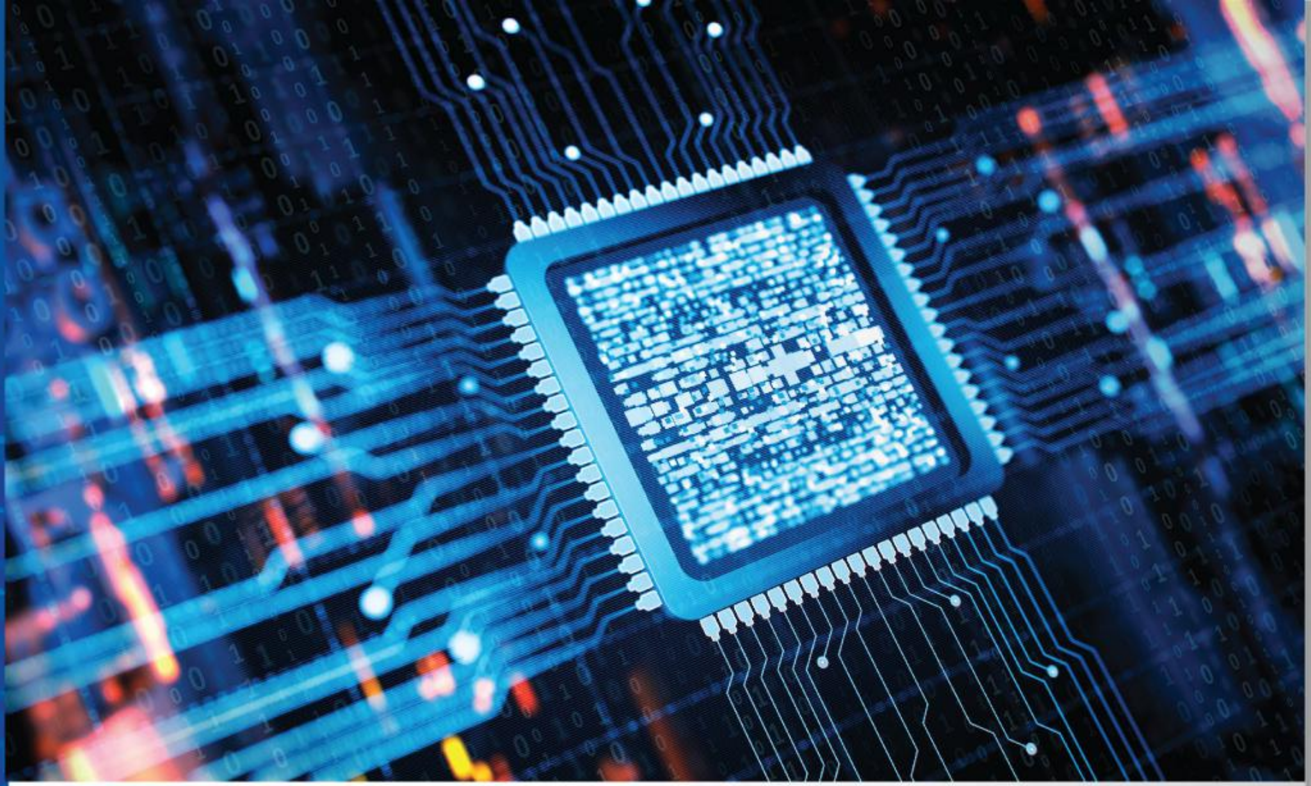
Alexandra Dmitrienko
University of Wuerzburg

Virginia Smith
Carnegie Mellon University

Farinaz Koushanfar
University of California San Diego

Reza Shokri
National University of Singapore

Content



1. MOTIVATION & INTRO

The goal of the PrivateAI Collaborative Research Institute https: //www.private-ai.org is to push the state of the art on decentralized and privacy-preserving machine learning. This mission is inspired by our belief that data is everywhere and that centralizing this data is often neither feasible nor desirable.

The majority of applications as well as a vast body of research today focus on centralized and batched machine learning (ML): Training data is collected, cleansed, and associated with labels. The resulting labeled training data are then used to train a model that is distributed and used for inference by clients.

We believe that this centralized approach to machine learning will be complemented by a wider range of decentralized models of machine learning. The overarching question we try to answer is "How can we securely and efficiently derive insights out of decentralized data while preserving the privacy of individuals?"

Federated machine learning is a first step on this trajectory: A central aggregator collects, aggregates, and redistributes models that resulted from local training by clients.

1.1 Outline and Research Clusters

The PrivateAI Collaborative Research Institute has two main goals. The first is to conduct basic and applied research on security and privacy of machine learning with a focus on decentralized machine learning. The second goal is to conduct applied research in collaboration with the industry sponsors that renders this research practically usable for a wide range of application scenarios. Our scope is detailed in Section 2. We push the state of the art in multiple research clusters.

Research on Machine Learning Algorithms (Section 3)

Security and privacy-enhanced algorithms for decentralized machine learning.

This cluster includes multiple topologies like federated or decentralized training, security and privacy of data or model handling, helper algorithms like private consultation or knowledge transfer, or tools for core privacy and security of machine learning which also includes randomization with privacy accounting to achieve differential privacy or watermarking to trace stolen models.

Research on Tools and Accelerators (Section 4)

Hardware and software to support security and privacy enhanced execution of those algorithms.

This cluster includes hardware extensions such as Trusted Execution Environments (TEE) or specific accelerators, including FPGA, to support classes of algorithms. It also includes cryptographic primitives and protocols such as Multi-Party Computation (MPC) and Homomorphic Encryption (HE) that can strengthen the security and privacy of the executed algorithms. This also inludes algorithms for detecting and mitigating Byzantine behaviors (including malicious nodes) during training. For data cleansing, privacy preserving protocols such as private vector linkage or private set intersection may be explored.

Research on Validation and Deployment (Section 5)

Applications that are used to showcase and validate the algorithms and platforms will drive new requirements and allow us to demonstrate the benefits of the developed technologies.
We envision the following impacts that will be generated by the the developed technologies.

We envision the following impacts that will be generated by the
PrivateAI Institute:

Security and Privacy for Machine Learning

Our research will push the state of the art in algorithms, run-times, applications and tooling. Due to the collaboration with industry, researchers are enabled to explore new relevant usage scenarios and requirements and are also able to validate their research in practical deployments.

Deploy and Validate key Research Results

Practitioners will be enabled to deploy decentralized machine learning with appropriate security and privacy guarantees. While today, many piecemeal results exist, it is hard to find the best technology for a given scenario. One goal is to provide a toolkit and guidance that allows practitioners to find and deploy the best solution based on given parameters such as topology, ML architectures, security and privacy requirements, and the acceptable trust into the parties and technologies involved.

At this stage we would like to stress that the ability to compute on encrypted data in general and for ML in particular could provide certain security gurantees that statisfy the securtiy and privacy requirement of mistrusting stakeholders involved in real-world application scenarios. However, today such solutions have prohibitive performance penalties. As a consequence, there is no one-size-fits all solution. Instead, solutions are crafted to fit the specific trust relations and regulatory requirements. Furthermore, in addition to technology advances, investments in policies and regulation, standardization, and adoption in educational paths will be required.

2. SCOPE OF THE PRIVATE-AI INSTITUTE

In the following, we will introduce the conceptual framework used to discuss machine learning tasks and define the adversarial setting and security and privacy objectives that we plan to address in our research.

2.1 Life cycle framework of Machine Learning tasks

Machine learning can be structured into multiple consecutive phases as parts of a life-cycle. The overall goal of machine learning is to to make inferences on new data that were not used during the training phase. For this report, we structure machine learning into following phases (see Figure 1) where each phase poses their own specific security and privacy challenges:

Data Acquisition

Before being able to train a model, training data along with possible corresponding labels need to be acquired. This may include extraction of data from databases or from real-world observations with a potential automated or manual labeling of acquired data. Examples of challenges are related to privacy-friendly data extraction, data masking, data anonymization, or elimination of maliciously modified input data samples.

Local training

Local training denotes the process of computing a model from training data and possible associated data labels. A privacy goal of the training step is to prevent recovery of privacy-sensitive information from the resulting model.

Collaboration and Federation

To improve model quality, multiple players - each with their local model may collaborate to enhance their local models or construct a global model by aggregating local models. In this phase, sensitive data that may be contained in a local model should not leak to other untrusted entities. Similarly, individual bad players should not be able to misuse the collaboration to corrupt the models of others.

Inference

During this phase, local or global models are used for inference by feeding new data to the models. Inference may be collaborative (similar to medical consultations). A privacy concern here is how to guarantee proper protection of identifiable information contained in the training data. A confidentiality concern is how the input data used for inference is protected and how to prevent leakage of the used model to unauthorized parties.

Forensics

The final stage in the machine learning life cycle is to conduct forensics to identify bad players or detect and recover from attacks. An important goal is model attribution, i.e., the ability to identify the original creator of a potentially stolen model.

While our research agenda is structured along this life cycle, some technologies span multiple phases. For example, to enable forensics, the corresponding technologies need to be deployed during training and/or inference. In addition to the specific phases of the machine learning life cycle, also the topology of the used learning architecture has an impact on security and privacy requirements and challenges. In the

following, we provide an overview of these typical topologies.

2.2 Topologies for Machine Learning

On an abstract level, especially in settings in which data acquisition, training and inference are performed by a single entity, the machine learning process can be seen as a relatively straightforward pipeline in which all steps including data acquisition, training and inference take place locally. However, in practice, especially when several different players (subsequently referred to as clients) are involved in the machine learning process, the training of models can be arranged in a number of different ways. In the following we discuss the main characteristics of and differences between centralized, federated and decentralized training topologies.

2.2.1 Centralized Training

In a centralized training topology, as shown in Fig. 2a, a number of participating entities (clients) collaborate to aggregate a joint global model. The training of this global model is performed by a central entity to which all participating clients submit their local training datasets. The central entity merges the individual local training datasets and uses them to train the global model.

While being simple and straightforward to implement, the centralized training model has the drawback that it is entirely dependent on a trusted third party acting as the central entity. The central entity has full visibility to all training data of clients and is therefore a critical central point of failure. Unless a very strong trust relationship exists between the central entity and the client the client may be unwilling to share its entire training data with the central entity. In addition, the centralized model has the drawback that it places high computational and communication overhead on the central entity, if the size of training datasets and number of clients is high. Last but not least centralized solutions are typically single point of failure.

This is one of the reasons why alternative training models have emerged. In particular, the federated learning model provides benefits, as it allows to reduce the communication and computational burden on the central entity and provides advantages with regards to the privacy of training datasets.

2.2.2 Federated Training

In a Federated Learning (FL) [127, 167] architecture the task of training is federated among a number of nodes as shown in Fig. 2b. Each participating client trains a local model based on their own (private) training dataset and shares only the trained local model with the central entity, who aggregates the clients' local models using an appropriate aggregation algorithm. Federated learning has numerous advantages in many usage scenarios. Since the training task is distributed to individual participating clients, the raw data of clients do not need to be shared with the central entity nor other clients, thereby providing better privacy for clients' local datasets. On the other hand, all participating clients can still mutually benefit from the models of others. This is particularly beneficial in settings like, e.g., IoT networks, where the training datasets of individual clients may be relatively small, as IoT devices do often not generate much data to train models on. This would make the training of accurate models based only on individual local datasets very challenging due to the lack of sufficient training data points.

The federated learning architecture can be organized in a number of different training set-ups. Kairouz et al. [94] distinguish three different types of centrally-coordinated federated training set-ups:

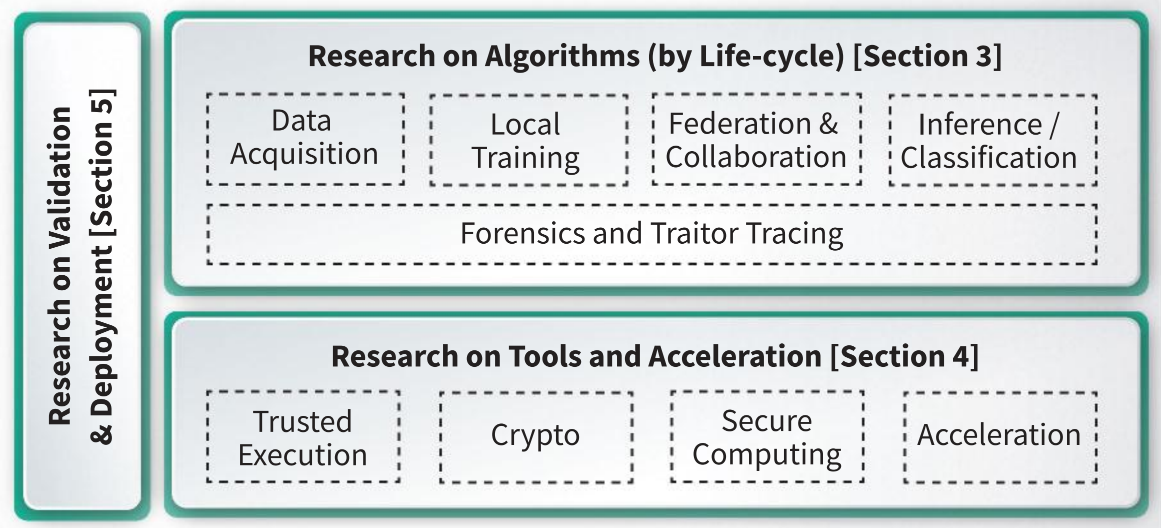

Research on Validation & Deployment [Section 5]
Research on Algorithms (by Life-cycle) [Section 3]
Data Acquisition
Local Training
Federation & Collaboration
Inference / Classification
Forensics and Traitor Tracing
Research on Tools and Acceleration [Section 4]
Trusted Execution
Crypto
Secure Computing
Acceleration

Figure 1: Research Clusters of the PrivateAI Institute

Cross-device
A very large number of devices (e.g. mobile phones) come online infrequently, train locally and a central entity aggregates model inputs and distributes updates to the clients. Training data are held locally by clients.

Cross-silo
A small number of larger organizations ('silos') that are usually online (e.g., hospitals) collaborate in training a joint model without sharing their raw training datasets.

Datacenter-internal
A large dataset under the control of a single organisation is partitioned in order to distribute training workloads among multiple servers that process the data. Data and models can be moved freely within the datacenter.

2.2.3 Fully Decentralized / Peer to Peer Training
Federated learning fits well to a setting where the central entity is trusted by clients. While the federated learning process maintains confidentiality in the sense that it does not disclose the training datasets of clients in the clear, the sharing of model parameters, gradients, or outputs in form of logits or predictions, may enable a curious central entity to infer potentially sensitive information about the datasets that clients have used to train their local models. In scenarios in which the privacy of training data needs to be preserved also against a curious central entity, a fully decentralized training topology as shown in Fig. 2c may be preferable.

Collaborative Inference
One example of decentralized learning is the collaborative inference [45]. We recognize that federated learning fits a particular setting where a central party is trusted by many participants. As mentioned federated learning has been shown not to provide full privacy, and it also forces all participants to share a common architecture. In the decentralized learning via collaborative inference, the collaborating parties exchange predictions, softmax outputs, or logits. These are common and same strucutre information released as model outputs. They are not dependent on model architecture and of much smaller memory footprint than the model updates in form of gradients or parameters. These properties enable collaboration between parties with heterogeneous model architectures and tighter protection of privacy. The privacy leakage can be limited when exchanging only final model predictions. One approach to solve the problem is to aggregate predictions obtained from many parties, add noise to obtain the differential privacy guarantees, and release a noisy, but in most cases correct, result. We elaborate on this solution in Section 3.4.3

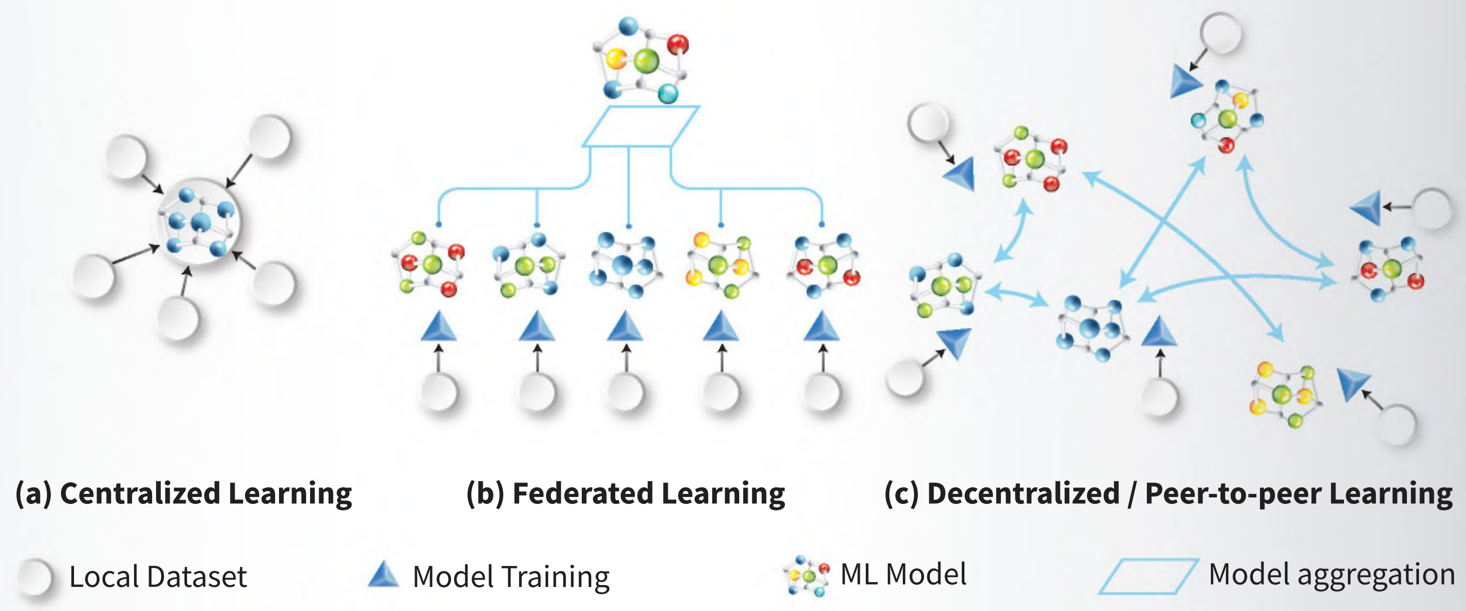


Figure 2: Topologies for machine learning

2.3 Modelling Adversary and Requirements

Our research agenda is targeted at solving prominent security and privacy challenges that exist in the machine learning settings outlined above. In the following we provide an overview over potential threats seeking to compromise the privacy of training data, the conficentiality of obtained ML models, as well as the integrity of models and reasoning outcome. We also outline the research problems that need to be resolved in order to mitigate these threats.

Trustworthy machine learning needs to satisfy various security and privacy requirements. Confidentiality aims to guarantee non-disclosure of sensitive information to unauthorized entities. Privacy aims to guarantee proper protection of sensitive (e.g., personally identifiable) information, against inference attacks. Integrity aims to prevent unauthorized modification of data and models.

While these requirements are hard to guarantee for centralized machine learning, protecting them in a decentralized setting is even harder since many nodes collaborate and some of these nodes may be malicious.

In particular for privacy and confidentiality, achieving guarantees among mutually distrusting players is an open challenge due to the conflicting interests of the stakeholders [10, 29, 31]: In the case of inference the data owner wants to query a model without revealing data. The model owner may be a tenant of the infrastructure owner, and aims to serve as many queries as possible by reducing the risk of data owner's data leakage and its own model IP leakage. In the case of distributed learning, data owners host their data onto tenant instances the tenant accumulates a host of third party data for which reducing the risk carried by data breaches is paramount. The future model owners want to learn from the siloed data without looking into the data.

2.3.1 Privacy of Training Data

Federated and decentralized learning enables multiple parties to leverage useful information from each others' datasets without actually sharing their data with others. However, as discussed above, model parameters and gradients (that are computed on the private datasets) can indirectly leak information about each party's dataset. In particular, sharing model parameters enables a curious aggregator or a participant to infer sensitive information about the individual data records in the private training sets of other parties. Major inference attacks include membership inference attacks [139, 168] (determine if a given data point is part of the training set of a model) and attribute inference attacks [130] (infer features of the training data, or statistical information about them). What makes federated learning more susceptible to such attacks, compared with centralized training, is that the adversary observes multiple copies of the target's high-dimensional model throughout the training. The adversary can further increase this leakage by actively sharing model parameters that force local gradient descent algorithms to react to the presence of particular data points in their training sets [139]. The attacks pose high risks on the effective privacy of federated and similar decentralized learning architectures.

2.3.2 Confidentiality of Models

Federated learning does not necessarily ensure the confidentiality of the resulting model. This motivates the search for new approaches that will provide guarantees of model confidentiality with different model ownership arrangements. This requires model protection across the whole life-cycle from data acquisition to traitor tracing. The question arises what level of confidentiality protection is required is based on an ownership model and terms of usage that correspond to the specific usage scenario:

In the federated learning setting, all participating organizations need to agree on an appropriate ownership rights before implementing federated learning and incorporate this into contractual arrangements; matters to be agreed include how and when the resulting model can be used, and any restrictions on distributing the model to third parties. In peer-to-peer training, there is no central entity to coordinate agreement on model ownership, making it difficult to define the obligations of all the participants.

Having defined the obligations of the participants in a federated learning system, it is necessary to ensure compliance. This can occur in several ways:

- **Proactively,** so that technical measures prevent participants from violating their obligations, or by
- **Deterrence,** so that even where a participant is capable of violating their obligations, the costs of an attack can be increased to the point that it is no longer in their interest to do so.

Proactive defenses are explored in the existing literature on white box model theft [111], but these are often costly either in computation or the need for specialized hardware support; we discuss these approaches in Section 4. Deterrence requires to detect and reliably demonstrate the ownership of ML models that are stolen or redistibuted by parties involved in federated learning.

2.3.3 Integrity of Data and Models during Training

Although federated learning allows collaborative training while preserving the privacy of participating clients, the confidentiality and integrity of the resulting model is not necessarily assured. This motivates the search for new approaches that will provide guarantees of model confidentiality with different model ownership arrangements. This requires model protection across the whole life-cycle from data-acquisition to traitor tracing.

While Federated Learning has many benefits w.r.t. accuracy and data privacy, it, however, suffers from inherent susceptibility to so-called model poisoning attacks where an adversary corrupts training data or a local model that then affect the central aggregated model. This vulnerability is a result of the fact that in the federated learning model the central aggregator can not be assumed to have any control over individual clients and can therefore not verify the correctness of the data used for training nor the actual process of model training. The aggregator only has access to the local models submitted by individual clients for aggregation.

This implies the challenge to design a secure aggregation process that is resilient against attacks in which individual training datasets or local models are manipulated by the adversary. This is a problem that is inherent to the distributed nature of any collaborative machine learning approaches. Since the set-up typically comprises independent clients over which other entities do not have direct control over, the contributions of individual clients participating in the ML process may be influenced by an adversary in an effort to manipulate the process to its advantage.

A number of different attack scenarios are possible:

Data poisoning

In this attack type, the adversary is able to manipulate the data that one or more clients use for training their local ML models. For performing this attack, the adversary does not necessarily need to compromise any of the clients, it is sufficient if it is able to inject manipulated data into the training datasets of benign clients.

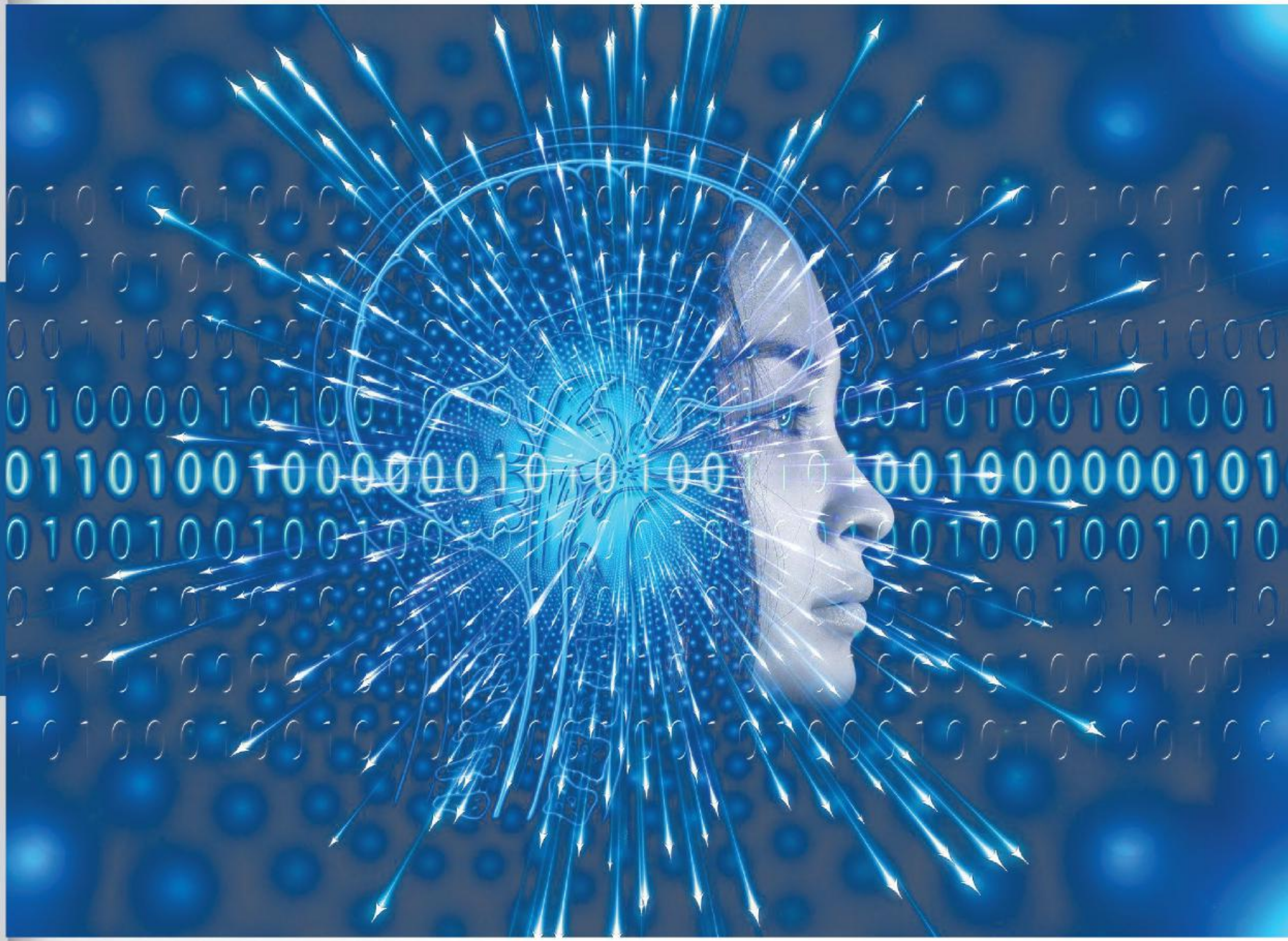
Model poisoning

In model poisoning, the adversary seeks to corrupt the models provided by individual clients to the collaborative learning process. Model poisoning can be achieved by performing data poisoning on the training dataset, but also other manipulations are possible, if the adversary is able to compromise individual clients. It can then directly influence the training process or modify the local model after training. The adversary can, e.g., modify learning rates, number of epochs used in training, or, scale the resulting models in order to optimize their impact on the result of the collaborative training process to achieve the attacker's goals. A number of aggregation algorithms for federated model aggregation have been proposed. It has been shown that the most widely used federated averaging algorithm Federated Averaging (FedAvg) is vulnerable to model poisoning attacks by which an adversary can, by suitably scaling the models it submits for aggregation, effectively introduce malicious functionality into the global model produced as a result of the aggregation [16].

Also the motivations and goals of an adversary performing attacks against the integrity of data and models can be diverse and thus target very different outcomes. At least following attack types can be identified:

Backdoor attack

In a backdoor attack the adversary seeks to 'hide' a specific functionality, a backdoor into the model resulting from the collaborative training process. Typically the adversary seeks to manipulate the model in a way that it will generate incorrect predictions for a specific set of inputs, the so-called trigger set. In many cases it seeks to achieve this in a way that does not impact the performance of the resulting model for other inputs in order to make it for others difficult to notice that the model has been maliciously modified by the adversary.



Model sabotage

Another motivation for the adversary may be to influence the performance of the resulting model in a way that is advantageous for the adversary. A straight-forward goal would be, e.g., to modify the model in a way that deteriorates its overall performance, thereby reducing its utility. But also other scenarios are possible, it could be, e.g., that the adversary would seek to deteriorate the performance of the model for the input data of specific other clients. This requires that the adversary has sufficient knowledge about the properties of the input data of other clients. However, in a competitive setting between the clients, this would allow the adversary to gain a performative advantage over the targeted other clients.

Various backdoor detection techniques have been developed in local settings [38, 87, 186, 197]. These methods allow the defender to inspect whether a pre-trained model has been backdoored during the training phase [38, 186, 197], or identify backdoor trigger for the victim model in real-time [87]. National Institute of Standards and Technology (NIST) has lunched a comptetition on backdoor dection to ensure safe model deployment [143]. To ensure model integrity in the federated setting, researchers design backdoor detection methods that recognize malicious model updates from local clients [60, 141].

Developping scalable and effective inspection techniques to guarantee data and model integrity in federated training remains an open challegne.

2.3.4 Integrity of Data and Models during Inference

Machine Learning has not yet reached true human-level robustness in many machine learning tasks, especially in the vision or audio domain. This is because many models are vulnerable to so-called adversarial examples [19, 176]. In the broadest sense, by adversarial perturbations, we define small imperceptible changes to data input that alter model predictions. These are called ε-bounded sensitivity-based adversarial examples. The existence of adversarial examples makes it difficult to apply the ML models in security critical areas, such as self-driving cars or fraud detection applications where adversaries may be able to modify the data that are used for classification.

3 RESEARCH ON ALGORITHMS FOR SECURITY AND PRIVACY-ENHANCED MACHINE LEARNING

3.1 Data Collection and Preparation

Before models can be trained and inferences can be made, data needs to be collected and prepared. This process is much more (wall clock) time consuming than the CPU intensive training of models and heavily relies on human intervention. All down stream process is affected by these first steps. Data cleaning or data preparation determines the accuracy of an inference much more than the tuning of machine learning parameters. If data is collected in a privacy-preserving manner, (post-) processing this data using machine learning may also be private. Hence, these steps are of crucial importance to designing private AI processes.

The first challenge is concerned with the collection of data from users over the Internet. In private data collection, a large number of clients each submit a single or few data elements and the collector needs to obtain private statistics, e.g., heavy hitters, sums, median, etc. Privacy for statistics usually means differential privacy (DP). However, the central model of DP is not applicable to private data collection, since the collector should be untrusted. Hence, either local differential privacy, the

centralized computation of differentially private statistic using trusted hardware, secure multi-party computation, or private aggregation protocols need to be used.

How can an entity collect data for machine learning purposes from a collection of users that is scalable and practical in communication and computation, accurate for a given (small) set of users, privacy-preserving for the users and does not require the users to trust the data collector?

The second challenge is private data preparation is the use of multiple data sources. For example, hospitals may try to link their databases joining common patients, in order to better identify patterns in treatment. However, these parties may not want to ex-change plaintext data due to legal compliance or concerns about the intersection. Often these techniques can be implemented using trusted hardware or secure multi-party computation.

How can two or more entities link their data sets in order to identify errors within their data sets in a protocol that is scalable and practical in computation and accurate and privacy-preserving for the entities' data sets?

The third challenge is private data repair. Once errors or inconsistencies have been identified, they need to be repaired, e.g., missing fields replaced or contradicting data replaced with a unifying entry. This process can itself be done using machine learning models and may leak information about the (faulty) input data. Hence, not only the input to the data preparation (and repair) process must be protected, e.g., using trusted hardware or secure multi-party computation, but also the repaired data released for machine learning purposes. Privacy in this case usually means differential privacy again.

How can data for machine learning purposes be repaired while protecting the privacy of the users collected in the data set?

3.2 Local Training

3.2.1 Privacy of Training Data

The two primary methods for providing differential privacy (DP) of training data during local training are DP-SGD [6] and PATE [148, 149]. These are two different approaches that achieve the same goal. DP-SGD makes fewer assumptions about the ML task than PATE and modifies the training stage while PATE is oblivious to the architecture of ML models but requires training of many models and assumes access to a public dataset.

DP-SGD(Differentially-Private Stochastic Gradient Descent) modifies the minibatch stochastic optimization to make it differentially private. During the model training, it tracks the access to the parameter gradients, i.e., the gradients of the loss with respect to each parameter of the model, and ensures that this access preserves differential privacy of the training data. Thus, the resulting trained model, per the post-processing property of differential privacy, limits the exposer of private information in the training set. By tracking detailed information of the privacy loss using the moments accountant, DP-SGD obtains tight estimates on the overall privacy loss.

While DP-SGD directly modifies the training mechanism of a single model, PATE requires training of an ensemble of models on private data and then injects the DP noise at the internal inference stage, after aggregating outputs from the ensemble. A model called student is trained on the differentially private predictions made by the ensemble, and once again, per the post-processing property of differential privacy, the student model protects the privacy of the training set. The student model that can be exposed via a public API is the final result of the PATE method. The models in the ensemble are called teachers and they are trained independently on non-overlapping partitions of the training set. There are no constraints on how the teachers are trained but they all aim at solving the same single-label classification task. Their predictions that are used to train the student model are made on a publicly available dataset.

Local training with privacy guarantees incurs a high overhead in software complexity, training efficiency, and model quality. An open question is whether the overhead can be substantially reduced while maintaining sufficient privacy guarantees.

3.3 Federation and Collaboration during Training

3.3.1 Collaboration without leaking local training data

In federated training, the individual nodes do not transmit original training data to other nodes. Instead, information about the models that were created using local training are exchanged. While this improves privacy, it still enables indirect information leakage [139, 167]. This allows membership and other inference attacks where an attacker tries to gain unauthorized information about training data used. It is still an open question how to mitigate such attacks:

How can AI algorithms reap the benefits of decentralized training while effectively protecting dataset privacy against inference attacks?

There exist tools, such as the ML Privacy Meter [138], to quantify the privacy risks of models in various “white box” settings, which can be extended to federated learning. The open questions about how to protect privacy of the training data are:

- What is the best methodology for an online analysis of the privacy risks of federated learning for each participant?
- How can parties share information about their local models without significantly leaking information about the individual data records in their training sets?
- Randomized algorithms can provide provable differential privacy guarantees, however with a potential loss in model’s accuracy. What are utility-preserving algorithms for privacy-preserving federated learning?

3.3.2 Secure Federated Learning

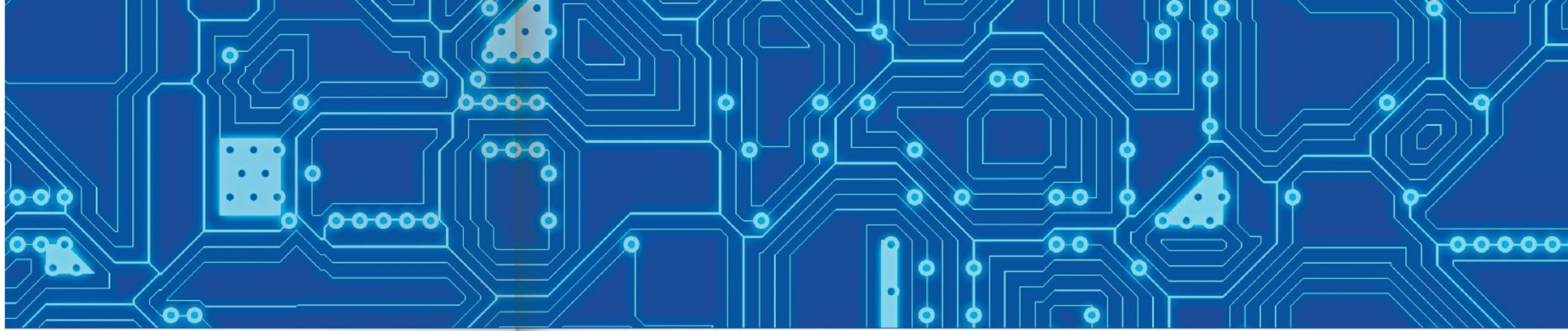
Recent works demonstrate that keeping the training data on users’ devices in federated learning does not provide sufficient privacy, as their private training data can be reconstructed by utilizing inference or inversion attack based on the model parameters shared by users [59, 63, 139, 200]. The corresponding research challenge is

Federated learning derives a global model from local models. Can we design privacy-preserving aggregation protocols ensuring that the individual local models are kept private and nothing beyond the aggregated global model is leaked?

To reach this goal, secure aggregation protocol is proposed in [26] to ensure that the privacy of individual model parameters is protected, both from server and other users. To guarantee privacy, the locally trained model parameters are masked by pairwise randomness and sent to the server such that the server aggregates users’ models in a privacy-preserving manner. However, the overhead of the secure aggregation protocol has a major bottleneck in scalability to the larger number of users, as the communication and computation overhead is quadratic in the number of users. This quadratic growth of secure aggregation overhead limits its practical applications to hundreads of users while the scale of federated learning is in the order of tens of millons [25]. This leads to further challege:

Quadratic overhead of secure aggregation protocol is the main bottleneck in scaling secure federated learning to a large number of users. Is quadratic the minimum overhead for secure aggregation protocol? How can we achieve a more efficient (e.g., sub-linear) secure aggregation overhead?

Another critical challenge in federated learning is the communication bottleneck, which is created by sending the model/gradient from the clients to the server, where the size of the gradient estimates or model updates that must be transmitted to the server at each iteration, can be extremely large, e.g., the 50-layer ResNet network has ~26 million weight parameters. Researchers have proposed many approaches to provide a communication-efficient federated learning system. One of these approaches is to use



quantization which allows users to iteratively send small model updates [11, 101, 164, 175, 187].

It has been shown in [54] that the state-of-the-art secure aggregation protocol require all users to quantize their model updates to the same level of quantization to guarantee correct decoding, even if they have different transmission rates. This severely degrades their performance due to lack of adaptation to both the speed of the available network (3G, 4G, 5G, WiFi) and the fluctuation of the network quality over time. Towards that, the authors in [54] have proposed a segment grouping approach that allows for secure model aggregation while using heterogeneous quantization.

However, the communication cost in their approach scales quadratically with the number of users, limiting its practical application to only hundreds and a few thousands of user. This brings us to the following open and challenging problem

Can we propose a new efficient protocol that achieve secure model aggregation with heterogeneous quantization in federated learning while having a non-quadratic, e.g., sub-linear, communication complexity?.

Additionally, the decentralized, distributed and massive-scale of Federated Learning opens up unknown attack surfaces that are often hard to quantify in advance, such as the presence of intermittent adversaries during training and inference and the possibility of malicious servers. Prior works [88] have attempted to quantitatively model these unknown attack surfaces and identify new threats to federated learning environments, but there remain numerous unsolved problems that must be addressed. To address this challenge in the setting of Independent and identically distributed (IID) data, a number of strategies have been proposed recently. In this scenario, gradient updates from the benign clients tend to be distributed around the true gradient, while the gradient updates from the adversarial clients can be arbitrary and are thus handled by applying robust estimation techniques for aggregating the client updates [22, 43, 60, 72, 173, 192].
In practice, however, data may not be IID across clients [199]. This makese attack resiliency more challenging since even updates from benign clients may be very diverse, thus degrading the performance of prior Byzantine resilient approaches in IID setting. To tackle this conundrum, some recent works such as [41, 80, 110] deal with heterogeneous data distribution in federated learning. Although these schemes achieve better performance in comparison to the prior approaches proposed for the IID data setting in one or more scenarios, the performance gap from the optimum achievable accuracy (in the presence of attacks) is quite high. In [153], the authors propose DiverseFL, which is the first work that can achieve near optimum accuracy in federated learning when data is non-IID. While [153] provides significant empirical results for demonstrating the gains of DiverseFL in practice, theoretical guarantees are not provided. Furthermore, DiverseFL requires each client to share a tiny fraction (up to 3%) of its local dataset with a trusted entity (such as some patients sharing their health records with the NIH for social good) to achieve attack resiliency, which maybe undesirable in some situations. This brings us to the following open and challenging problem in robust federated learning:

How can we mitigate Model Poisoning attacks if the local datasets if the clients are non-IID, while providing optimum data privacy and model perfor-mance guarantees?

Security and privacy considerations of secure federated learning are mainly focused around two seemingly separate directions: 1) protecting the privacy of individual models against honest-but-curious adversaries and 2) ensuring robustness of the global model against adversarial manipulations such as model or data poisoning. Achieving them simultaneously, however, presents a major challenge. As the local models are protected by random masks, the server cannot observe the individual user updates in the clear, which prevents the server from utilizing outlier detection protocols to protect the model against model poisoning. This brings us further challenge:

How can we make federated learning protocols robust against model poisoning while preserving the privacy of individual models?

3.3.3 Federated Learning at Scale

Federated Averaging (FedAvG) is the most basic parallel optimization algorithm in Federated Learning. Many variants of FedAvg have been proposed to achieve the more robust convergence [113, 198], the faster convergence [32, 34, 133], and the higher scaling efficiency [124, 161, 174]. Recently, several researchers proved that the algorithm achieves the linear speedup with respect to the number of workers in parallel training under IID data distribution [73, 171, 193]. Unfortunately, it has been shown that FedAvg cannot achieve the linear speedup if the data distribution is non-IID [114]. That is, as FedAvg scales up, the training converges more slowly. Federated Learning assumes a large number of weak compute resources such as mobile phones. In order to make it more practical, therefore, the sub-linear speedup issue should be addressed.

How to efficiently scale secure privacy-preserving ML to hundreds, thousands, or millions of users? How can this be achieved if the data is non-IID?

FedAvg allows local clients to independently train their own models and periodically averages the model parameters across all the clients. While having less frequent communications, this approach suffers from a discrepancy of loss function across the clients making the global model converge slowly. Especially with non-IID data distribution, such a discrepancy can result in a significant drop of convergence rate. To scale up the Federated Learning solutions to hundreds, thousands, or even millions of clients, therefore, it is crucial to improve the convergence rate of FedAvg. One of the potential solutions is to design different model averaging methods. Instead of allowing fully independent updates for many iterations, the clients may synchronize their states more frequently, making a practical trade-off between the communication cost and the convergence rate. Another potential solution is to adaptively adjust key hyper-parameters, such as learning rate, batch size, and model averaging interval, at run-time based on the local training progress. Considering the heterogenous data distribution, the optimal hyper-parameter settings for each local model would be likely different across all the clients. Thus, a faster global loss minimization can be expected by customizing the hyper-parameters in a local progress-aware manner.

3.3.4 Federated Learning on Resource-Constrained Systems

Scaling up the deep neural network (DNN) size (e.g., width, depth, etc.) is known to effectively improve model accuracy. However, the large model size impedes training on resource-constrained edge devices. For instance, federated learning (FL) may place undue burden on the compute capability of edge nodes, even though there is a strong practical need for federated learning due to its privacy and confidentiality properties. The challenge we aim to solve is as follows:

How can compute resources within datacenters help to reduce the burden for edge devices?

To address the resource-constrained reality of edge devices, Federated Group Knowledge transfer (FedGKT) [77], a group knowledge transfer training algorithm is a promising approach. FedGKT designs a variant of the alternating minimization approach to train small personalized CNNs on edge nodes and periodically transfer their knowledge by knowledge distillation to a large server-side CNN. FedGKT consolidates several advantages into a single framework: reduced demand for edge computation, lower communication bandwidth for large CNNs, and asynchronous training, all while maintaining model accuracy comparable to FedAvg.

3.3.5 Automated Federated Learning (AutoFL)

As [94] points out, when training deep neural networks under the federated learning setting where the data is non-IID (non-identical and independent distribution), using the predefined model architecture may not be the optimal design choice. Since the data distribution is invisible to researchers, to find a better model architecture with higher accuracy, developers must design or choose multiple architectures, then tune hyperparameters remotely to fit the scattered data. This process is extremely expensive because attempting many rounds of training on edge devices results in a remarkably higher communication cost and on-device computational burden than the data center environment.

Can model search (AutoML) help us optimizing the model architectures of federated learning?

We advocate automating federated learning (AutoFL) to simplify the aforementioned model design and remote hyper-parameter tunning. Especially, Federated Neural Architecture Search (FedNAS) [76], which automates the design process of the model architecture without inefficient manual attempts, is a promising direction. Inspired by the efficient optimization algorithm proposed by MiLeNAS [79], FedNAS can help scattered workers collaboratively searching for a better architecture with higher accuracy. A distributed AutoFL system with FedNAS algorithm has been released at FedML [78]. The experiments on non-IID dataset show that the architecture searched by FedNAS can outperform the manually predefined architecture such as DenseNet and ResNet.

3.3.6 Personalization

A defining characteristic of federated learning is that the distributed data are likely to be heterogeneous, i.e., each client may generate data via a distinct data distribution. Moreover, the client hardware and system heterogenity also bring challenges to achieve an efficient federated training system. It is therefore natural to consider techniques that provide personalized models for clients.

How can we support personalization, i.e. provide local models that are optimized to the local data distribution?

To enable personalization, a simple approach is to incorporate client-specific features. These features may be naturally occurring in the data, or may take the form of some auxiliary meta data. However, in lieu of (or in addition to) including such expressive features, it is also common to consider techniques that provide personalized models for clients. We discuss several popular techniques (multi-task learning, clustering, fine-tuning, and meta-learning) below.

Personalized search

The aforementioned FedGKT and Fed-NAS both have the potental to achive this goal. In FedGKT [77], the personalized model is not necessary to be of the same weights or architecture. In the scope of FedNAS [79], a personalized searching algorithm can further be developed to generate personalized model architecture for each client.

Multi-task learning

To model the (possibly) varying data distributions, $\boldsymbol{X}t$ on each client, it is natural to consider learning a separate model for each client’s local dataset. If we view learning from the local data on each client (or possibly a group of clients) as a separate task, we can naturally cast such a problem as an instance of multi-task learning. In multi-task learning, the goal is to learn models for multiple related tasks simultaneously. (Smith et al) [170] first proposed learning personalized models in federated settings via a primal-dual multi-task learning framework, which is applicable to convex objectives. Other multi-task learning formulations used to produce personalized but related models include interpolating between local and global models [125], regularizing the local models towards their average [55, 74] or towards some reference point [51], and enforcing hard parameter sharing [8].

Clustering
Personalization can also be obtained via device clustering (or data clustering) and learning a single model for each cluster [65, 125, 160].

Fine-tuning and Meta-learning
Local fine-tuning is a natural approach for personalization [194]. Meta-learning can be mostly viewed as a specific form of fine-tuning where it learns a good initialization starting from which the model can quickly adapt to new tasks (i.e., devices) with potentially only a small number of samples. Meta-learning and federated learning are closely related [97], and some works have also applied meta-learning for personalized federated learning [e.g., 35, 58, 91, 109].

3.3.7 Secure Knowledge Transfer
Sharing model parameters is the simplest way of exchanging information about training data in federated learning. This approach has many security and privacy shortcomings, in addition to having a significant communication overhead. Model parameters contain a significant amount of information about the training data, and sharing them leaks information to curious participants and aggregators. Also, robust aggregation of high-dimensional vectors is a challenging problem. Malicious participants are able to successfully manipulate the aggregated model by making small targeted modifications to their model updates, which remain undetected by robust aggregation algorithms. A potential solution to these problems is to make use of different knowledge transfer algorithms, which are inherently more privacy-preserving and robust [33].

How to transfer knowledge between participants with provable guarantees for privacy and robustness with respect to adversarial parties?

3.3.8 Decentralized Collaborative Privacy-preserving Machine Learning .
A general federated learning system uses a central parameter server to coordinate the large federation of the participating users. This cetralized topologies (corresponding to a star graph) often significant bottleneck on the central server in terms of communication bandwidth, latency and fault tolerance. To address this bottleneck, decentralized (server-less) framework based on secure multi-party computing (MPC) and homomorphic encryption (HE) have been proposed for privacy-preserving machine learning. These approaches, however, fall short of addressing the scalabiltiy of privacy-preserving machine learning, i.e., their constructions are limited to three or four parties [135, 136, 185]. This leads to research challenge:

How can multiple data-owners (beyond 3-4 parties) jointly train a machine learning model without a central server while keeping their individual datasets private from the other parties?

3.4 Classification/ Inference

3.4.1 Protection against Adversarial Examples

How can the phenomenon of adversarial examples be precisely characterized and what makes attacks using them successful?

There are many important problems that remain open in adversarial machine learning. The first one is that it is still unclear how to precisely define adversarial examples. The certified defenses against the most widely accepted definition of ε-bounded adversarial examples ensure that within the ε distance from a given input, the label remains constant. There is another notion of adversarial examples that are called the invariance-based adversarial examples [178]. They are ε bounded from the initial input, however, the real label of the perturbed input changes within the ε ball. The robust classifier is too constant in its predictions and incorrectly assigns the initial label that is no longer correct.

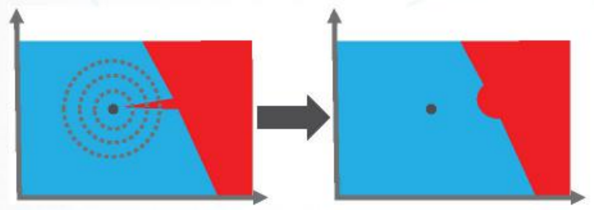


Figure 3: Randomized smoothing. The initial decision boundaries (left) and the smoothed decision boundaries (right).

The community has not reached a consensus on the question of what causes the adversarial examples. It is another open challenge. Multiple contemporaneous lines of work studied different aspects of this problem, postulating linearity and over-parametrization as possible culprits [19, 69]. One of the intuitive explanations is that adversarial examples are caused by small measure regions of adversarial class jutting into a correct decision region. This model is commonly accepted because the inputs remain correctly classified when perturbed with small random noise (e.g., Uniform or Gaussian). However, we can also easily find the attack vector that moves the inputs to decision regions where they become misclassified. This motivates a very natural defense strategy. The idea is to smooth the decision regions by adding Gaussian noise to the input and selecting the majority class of the classifier over this noise [46, 104, 108]. The dotted circles in Figure 3 (left) are the level sets of the Gaussian noise distribution centered at the input data point. The initial decision boundaries of the classifier are rough and with many cones. After the randomized smoothing procedure, shown in Figure 3 (right), the boundaries are more regular and the incorrect decision boundary lies much further away from the data point, which makes finding the adversarial examples impossible within the certified region. Randomized smoothing is a provable adversarial defense that scales to ImageNet.

How can one construct effective defenses against adversarial examples that are robust? Can such defenses be theoretically validated?

Unfortunately, the provable defenses against adversarial examples perform substantially worse in practice than the best empirical defenses based on adversarial training (approximately 2X lower robust accuracy). The idea behind adversarial training is to combine the training process with the generation of adversarial examples. For each input, we find an adversarial example that is then fed into the training process with the correct label. We use over-parameterized deep networks for standard training but these large models do not have enough learning capacity for adversarial training. Using the adversarial examples during training smoothes the neighborhood of natural data, which consumes much more model capacity. Consequently, models with higher learning capacity can be trained with adversarial examples more effectively and enable more robust classification. However, we observe diminishing returns for larger models. Deep or wide convolutional networks with a tremendous number of parameters are prohibitively difficult to optimize. The main problems in optimization of such networks are vanishing, exploding, or noisy gradients, cliffs, plateaus, saddle points, and other flat regions. The development of new vision architectures that train more efficiently on these datasets becomes increasingly important and more work is needed in this area. The Vision Transformer is a preliminary step towards generic, scalable architectures that can solve many vision tasks, hopefully robustly, and probably tasks from other domains as well [52].

Another solution to the demand from high learning capacity for adversarial training is to utilize the capacity judiciously. This can be done by reweighting the training data points. The method assigns larger weights to the adversarial data whose natural counterparts are closer to the decision boundary. Analogously, smaller weights are assigned to adversarial data points that were found starting from natural points further away from the decision boundary. This method was shown to significantly improve standard adversarial training [195]. With ever-increasing dataset sizes, we still need better methods to improve robustness.

One of the issues with adversarial training is that it overfits to the specific attack used during the creation of the defense. Even worse, using the specific attack during training can make the models even more vulnerable to other types of attacks. To overcome the problem, one proposal is to train the model on the strongest adversarial example for a given input, this is called the max strategy [180]. The caveat is that the max strategy lowers the accuracy on clean data and further increases the gap between accuracy on clean data vs robust accuracy. The open question is if we have to sacrifice the performance for legitimate users to train robust models.

How can one ensure that defenses against adversarial samples have sufficient coverage and can adapt to potential changes in attack strategies?

The bottom line is that adversarial training is rather a brute force method and requires us to train on many possible perturbations to be robust. In practice, the computational cost of such a defense is high. It is also difficult to construct a theoretical model of how the adversarial examples are crafted [68]. Adversarial examples are solutions to an optimization problem that, for example, in the case of neural networks, is non-linear and non-convex. Thus, it is very hard to make a theoretical argument that a given defense will be robust against all possible attacks. Adversarial examples require models to produce good outputs for every possible input. However, the number of possible inputs is humongous, and usually, the machine learning models work well on a small part of all the possible inputs. Finally, current defenses are not adaptive. For instance, if an adversary detects that a defense masks gradients then he or she can switch from a white-box attack that leverages the gradient to a black-box attack that is a gradient-free method. Thus, if a given defense closes some vulnerabilities, it usually leaves others open. An adversary who uses an adaptive attack seems to have the advantage in this case.

3.4.2 Private inference

The goal of the private inference is to enable two parties to run a neural network inference without revealing either party's data. We consider a user who wants to obtain model outputs for his or her data and a service provider who exposes a model inference service. The standard non-private inference on machine learning models compromises one party's privacy, where either the user sends sensitive inputs for inference, or the service provider has to send its proprietary machine learning model to the user. Private inference poses two research questions. The first research question aims to keep the inputs private. One example is to use a public classification service to process some medical data while guaranteeing the privacy of this data:

Can an inference service (potentially malicious) efficiently offer its services to an honest user while guaranteeing privacy of the user's data?

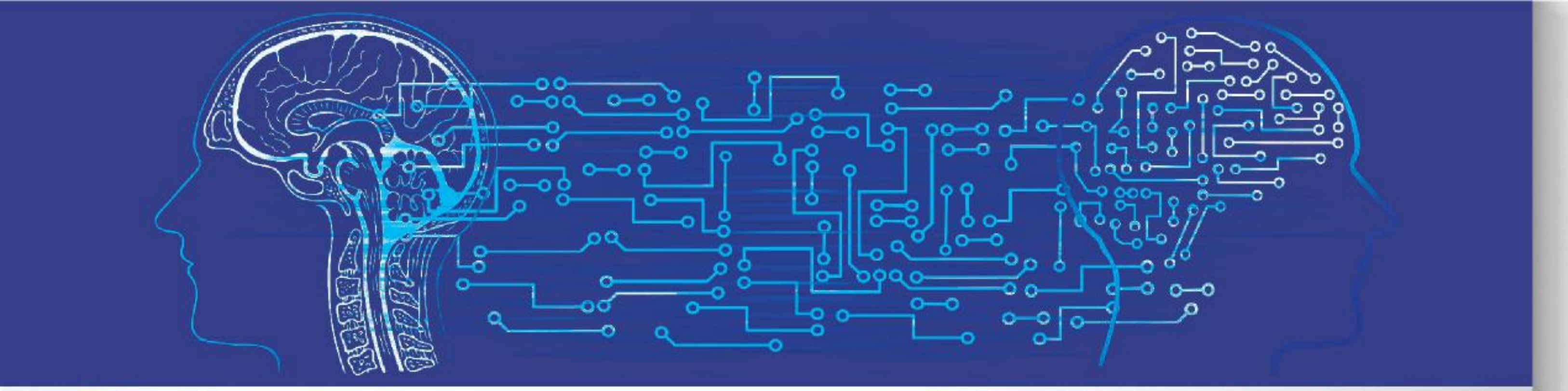
The second research question aims to protect the model against leakage:

Can an (honest) inference service efficiently offer its services to a (potentially) malicious set of users while guaranteeing that its proprietary model is not leaked?

In the context of neural networks, private inference typically uses homomorphic encryption (HE) or secure multi-party computation (MPC) methods. Pure homomorphic encryption is computationally expensive and does not support common non-polynomial activation functions, which leads to the leaking of preactivation values (feature maps at hidden layers). It can be partially ameliorated by using linear approximations, however, this causes a drop in accuracy (or other performance metrics of the models). On the other hand, tools that use solely secure multi-party computation protocols avoid leaking pre-activation values as they can guarantee data confidentiality on non-polynomial activation functions but may compromise the security of the model architecture by leaking activation functions or model structure (See also 3.5.1). While solutions based on MPC are less costly in terms of computational resources, they typically require much more communication between the parties. Recent lines of work on private inference propose hybrid schemes that integrate different privacy preserving tools (e.g., several MPC protocols, or MPC and HE). These hybrid frameworks combine the benefits of underlying protocols to maximize performance and reduce costs.

Zhang et. al. [196] provide a survey of state-of-the-art solutions for private inference, along with a comprehensive list of libraries that implement basic HE and MPC primitives. Specifically, libraries such as nGraph-HE [24] and CryptoNets [53] provide pure homo-morphic encryption solutions to secure neural network inference. nGraph-HE, an extension of graph compiler nGraph, allows secure inference of neural networks through linear computations at each layer using CKKS homomorphic encryption scheme [24, 44]. CryptoNets similarly provides private inference using the leveled homomorphic encryption scheme, YASHE [53]. Several libraries employ primarily MPC methods that rely on ABY [50], a tool providing support for common non-polynomial activation functions in neural networks through the use of both Yao's garbled circutis [191] and the protocol of Goldreich-Micali-Wigderson (GMW) [67].

Recent secure prediction systems that use hybrid protocols and do not rely on trusted third parties are, for instance: Gazelle [93], MP2ML [23], and Delphi [132]. These systems execute linear computations (e.g., convolutional or fully-connected layers) via HE and non-polynomial activation functions (e.g., ReLU or MaxPool) via MPC. Gazelle introduced several improvements over previous methods for private inference largely related to latency and confidentiality. In particular, the Gazelle framework provides homo-morphic encryption libraries with low latency implementations of algorithms for single instruction multiple data (SIMD) operations,



ciphertext permutation, and homomorphic matrix and convolutional operations, pertinent to convolutional neural networks. Gazelle utilizes kernel methods to evaluate homomorphic operations for linear components of networks, garbled circuits to compute non-linear activation functions confidentially, and additive secret sharing to quickly switch between these cryptographic protocols. MP2ML employs nGraph-HE for homomorphic encryption and ABY framework for evaluation of non-linear functions using garbled circuits. Delphi improves Gazelle's online runtime by moving heavy cryptographic operations to an offline phase, during which model weights are secret shared. In the online phase when the client's input is available, linear operations are simply performed over secret shared weights.

A recent line of work considers modifying and adapting neural network architectures such that the resulting network is more amenable to secure computation. CryptoNAS [64] and DeepRe-Duce [89] design ReLU efficient networks for the private inference task. Delphi [132] and SafeNet [118] reduce the cost of ReLU layres by judiciously replacing them with polynomial approximations. Binary Neural Networks (BNN) have also shown compelling performance for private inference by allowing specialized and highly optimized cryoptographic protocols [84]. There is a substantial effort in the community to improve private inference since its performance cost in comparison to the standard inference is substantial (10 or 100X higher). This calls for new solutions:

Can we create customized cryptographic solutions, search for crypto friendly architectures, or better utilize the underlying hardware to accelearate the private inference?

3.4.3 Collaborative Classification

We recognize that federated learning is limited and only fits a particular setting where a central party is trusted by many participants. Federated learning only provides confidentiality, not privacy. In other words, while participants of a federated learning protocol do not share their data directly, the gradients they share still contain private information initially found in the data. Federated learning is also limited to the setting where a central party wishes to learn from many participants: this forces all participants to share a common architecture which is decided upon by the central party. This begs the question of whether a collaborative form of machine learning can be achieved where the participants are all on an equal foot and learn from one another rather than simply contributing to a central party's model. We refer to this as collaborative ML in the following. This raises several technical challenges: how can we aggregate the predictions of a heterogeneous ensemble of models? How can we aggregate these predictions while bounding any privacy leakage? can this aggregation be performed in a fully distributed fashion? While progress in cryptographic primitives for machine learning and differentially private machine learning lay the foundations for such a collaborative approach to machine learning, achieving collaborative

classification requires non-trivial assemblage of these techniques. One direction we propose to explore is to augment PATE [148, 149] with cryptographic primitives (HE and MPC) to obtain a protocol which is both confidential and differentially private and allows for collaborative learning among a limited number of participants: this is the direction we outline in a preliminary publication where we name this protocol CaPC [45].

CaPC (Confidential and Private Collaborative) ML guarantees the protection of private information (including personally identifiable information) contained in training data using PATE and its associated differential privacy guarantees. Thus, it prevents all attacks captured by the framework of differential privacy for reasoning about privacy: this primarily includes inferring sensitive information about the individual data records from the training set. For instance, it protects against membership inference attack (which can determine if a given data point is part of the training set of a model) or the attribute inference attack (which infers features of the training data, or statistical information about them). In addition to this differential privacy guarantee, CaPC preserves the confidentiality of its inputs using hybrid (HE with MPC) private inference. Additionally, CaPC limits the leakage of information about the models used in each of the collaborating parties.

There are three main actors in CaPC: a querying party, answering parties, and a privacy guardian. The collaboration is enabled between any number of parties and each party can act as querying or answering party. If a querying party wants to label its new data points, it encrypts them and sends for inference to each answering party. The answering parties do private inference and their predictions are aggregated using the privacy guardian which adds noise to answer the query with differential privacy. Encryption of the data provided for inference protects the confidentiality, whereas noise added by the privacy guardian ensures differential privacy for the data on which the answering parties trained their models.

This approach provides many advantages: e.g., it provides confidentiality and differential privacy, it is applicable in settings with few participants whose models are trained with heteregeneous architectures, allows for improving each participant's model etc. However, several research challenges need to be tackled to further the adoption of CaPC in practical settings. In particular, the use of cryptographic primitives for confidential (i.e., private in the sense of cryptography) inference introduces a computational overhead. This begs the question of whether introducing hardware solutions like trusted execution environments could help alleviate some of the computational overhead. Furthermore, the current experiments indicate that about a hundred parties are sufficient to provide good differential privacy guarantees. Thus, one of the main questions is:

Could we reduce the number of collaborating parties to few participants and be able to preserve their privacy at a relatively low cost?

The current trust model assumed by CaPC is already advantageous but still assumes that participants place (limited) trust in a third party (called the privacy guardian). Future research will be able to further lighten assumptions made by CaPC's trust model to enable its deployment in settings where a third party cannot be trusted. Other aspects to consider are:

Which hardware and software innovations will con-tribute to decreasing the trust needed in the third party or altogether remove the third trusted party from the CaPC protocol? Related to this, what techniques can help understand and protect the confidentiality and privacy of honest participants when more than one of the collaborating parties collude for malicious purposes?

3.5 Protecting Intellectual Property and Forensics

3.5.1 Motivation

As we saw in Section 2.3.2 confidentiality of machine learning models is important because, in many cases, they constitute commercially-sensitive intellectual property. However, in many applications, these models cannot be commercially exploited without exposing them to attackers, whether during the training or inference phases.

There has been a line of work on IP protection of pretrained DL models. A unified DNN watermarking framework is proposed in [48] that supports model ownership proof in both model distribution (white-box) setting and MLaaS (black-box) setting. An extension to model usage tracing is demonstrated in [39] that adapts anti-collusion fingerprinting for DL models. The paper [37] further provides a hardware-bounded IP protection solution for DL devices leveraging the idea of on-device attestation. A parallel line of research on DL IP protection targets to prevent model extraction attacks [103, 181]. The paper [90] suggests an entangled watermarking method as a defense against model extraction. Instead of focusing on the pretrained model weights, the paper [123] considers the original dataset used for training as the valuable IP and designs a technique that allows the owner to claim authorship on model copies obtained via model extraction.

It may not be possible to completely prevent model extraction, where an attacker queries the model and uses the information gained from the responses to construct their own surrogate models. Often an attacker will be in physical possession of a device that contains the model, allowing them near-unlimited black box access. However, if the model can be attributed back to its source, this can deter the attacker from using the extracted model outside of a private setting.

The goal of deterrence is to persuade the adversary that the costs of some undesirable action outweigh the benefits. This can occur in two ways:

• **Reducing** the benefit that the adversary gains by model theft by degrading the quality of the stolen model. This is a proactive approach that seeks to prevent high-fidelity model theft in the first place.

• **Increasing** the cost suffered by the adversary in mounting a model theft attack. This may involve measures such as increasing the cost of model queries, or by detecting attacks in a way that allows retribution by way of commercial or legal sanctions.

Each approach has its own challenges; reducing the quality of stolen models inevitably reduces the accuracy of the model for legitimate users. Deterring model theft by increasing the cost to the adversary requires an accurate accounting of the adversary's costs and benefits in order to be effective. How best to deter model theft therefore remains an open question.

(How) Can we deter model theft and model extraction in a way that is easy to deploy, and does not conflict with privacy-preserving machine learning?

3.5.2 Detecting Model Ownership

Deterrence can be achieved by ensuring that if one organization violates its confidentiality commitments by distributing the model into third parties, then other owners can prove this using techniques such as watermarking or fingerprinting [7, 14, 37, 39, 48, 119]. However, these methods are applicable only in the centralized ML, where a central entity, such as the model owner, does the training itself. How to apply watermarking or fingerprinting techniques when there could be multiple owners (e.g., cross-silo federated learning) still remains as a question.

How to design strategies for an ownership demonstration that can be carried out in federated learning systems?

It is difficult to demonstrate ownership or prove violations of confidentiality in many common federated learning settings. For example, in some large-scale federated learning settings a large number of clients participate in the training and have the right to use the model, but not to redistribute it. In this case, the large number of clients makes it extremely challenging for a central entity to identify who has improperly redistributed the federated learning model.

In the peer-to-peer setting, where there is no trusted aggregator that can ensure proper watermarking or fingerprinting, this is even more challenging. Nevertheless, as peer-to-peer ML training becomes more popular, the demand for such solutions will only increase, and ownership demonstration mechanisms will need to accomodate this approach.

3.5.3 Proving Model Ownership

In Section 3.5.1, we proposed deterrence by ownership proofs as a mechanism for the protection of machine learning models. In addition to finding models that are owned by oneself, an important additional goal is to prove ownership to others. Without such evidence, deterrence is weakened because a potential violator might reason that their transgressions will not be punished since decision-makers cannot be convinced to impose commercial or legal sanctions. In this section, we explore what form this evidence might take, and how it can be obtained using forensic techniques, which we define broadly to mean techniques that involve examination of models and inference attempts in order to reveal the history of the model under examination.

One approach to this is for contributors to a system to try to prove that some model includes their contribution; they can then demand that the possessor of this model account for how it came into their possession. They can then follow the trail back until they find either some illegitimate usage of their contribution, or they conclude that the model was obtained without any violation of trust.

How can dataset owners and other stakeholders prove their contribution to a machine learning model?

However, identifying their contribution is a non-trivial task, particularly when the model has been subject to multiple transformations since the original contribution, and doing so reliably remains an open research question.

Identifying that misuse has happened at some point in the past is not enough, as it may provide impossible to force all parties in the chain to truthfully account for their possession of the model in question. It is therefore desirable to forensically identify violators without the cooperation of subsequent users, while allowing those who have obtained a model legitimately to be confident that the real violator can be identified, so that they will not be implicated in misuse that took place earlier in their supply chain.

How can the misuse of machine learning models be attributed to the responsible party?

How this can be achieved remains an open research question, particularly when there are many stakeholders making diverse contributions to an ML-based system, each of whom wish to make a reliable 'imprint' in the eventual model that can be used to demonstrate the history of their contribution as it passes through supply chains.

4 HARDWARE AND ACCELERATION FOR SECURITY AND PRIVACY ENHANCED MACHINE

4.1 Trusted Execution for AI/ML

Trusted Execution Environments (TEEs) allow the isolated processing of data. Their goal is to provide integrity and confidentiality guarantees for programs[1] and data on systems and devices where all software outside the TEEs is assumed to be potentially compromised and malicious, including privileged software, such as the operating system (OS), that has to be trusted in classical non-TEE system architectures.

TEE architectures have been developed for a wide range of systems. Depending on the system and its specific requirements, the approach and design characteristics of the TEEs can be very

very different. TEEs for embedded systems [27, 56, 100, 144, 172] often assume the absence of virtual memory and caches while satisfying real-time capabilities [27]. TEEs for servers [1–4, 49] support trusted services within trusted virtual machines. TEE for end-user devices [12, 13, 81, 86, 126] aim at flexibility for a wide range of applications.

4.1.1 End-to-end Integrity of Data and Models

In ML TEEs can e.g. be used to process sensitive data, such as privacy-sensitive training data for ML or to protect ML-models representing companies' IP. However, the integrity of the processed data can only be ensured if the data feed into the TEE is correct, i.e., if all previous processing steps of the data also provide integrity guarantees. This leads to important challenges:

How to build a distributed system that can ensure data integrity for the entire data-lifecycle assuming that not every device is equipped with TEE capabilities and can provide guarantees for the correct processing of data.

Beside the correct processing of data in a distributed system, the correctness of data must be ensured already during its generation or collection. This requires that sensors and other data-generators must be secure and enable end-to-end data-integrity guarantees for their output to serve as input for TEEs.

How to provide integrity guarantees for data during their initial generation and pass these data (preserving the integrity guarantees) to a TEE?

However, while abstractly providing strong and desirable security guarantees, deployed TEEs have a number of shortcomings, which render existing TEEs unfit as the sole security solutions for protecting sensitive data and workloads, such as machine learning training and inference. Various side-channel-based attacks have been demonstrated against different deployed TEEs [28, 36, 47, 57, 70, 105, 134, 137, 162, 182, 183], raising the question

How to build TEEs that can guarantee their security properties considering all current as well as future attacks, i.e., guaranteeing the processed data's confidentiality and integrity indefinitely.

4.1.2 Usage Control

Besides ensuring privacy and integrity of DL inference/training [83, 145, 179], TEEs can also be leveraged to provide hardware-level usage control of intelligent devices [37]. Particularly, the device provider of a DL hardware can ensure that only the pre-authorized models can perform normal execution on the computing platform using on-device DNN attestation. Developing an efficient and effective on-device DNN attestation methodology is challenging since the attestation scheme is required to: (i) Preserve the performance (e.g., accuracy) of the deployed DNN; (ii) Provide reliable and secure attestation decision; (iii) Incur low latency and power consumption to ensure applicability in real-time DNN applications and resource-constrained systems.

[1] Confidentiality for the program-code is not a primary goal in all TEE-architectures. However, all TEE-architectures need to provide means to securely provision data; this mechanism can be extended to securely provide code as well.

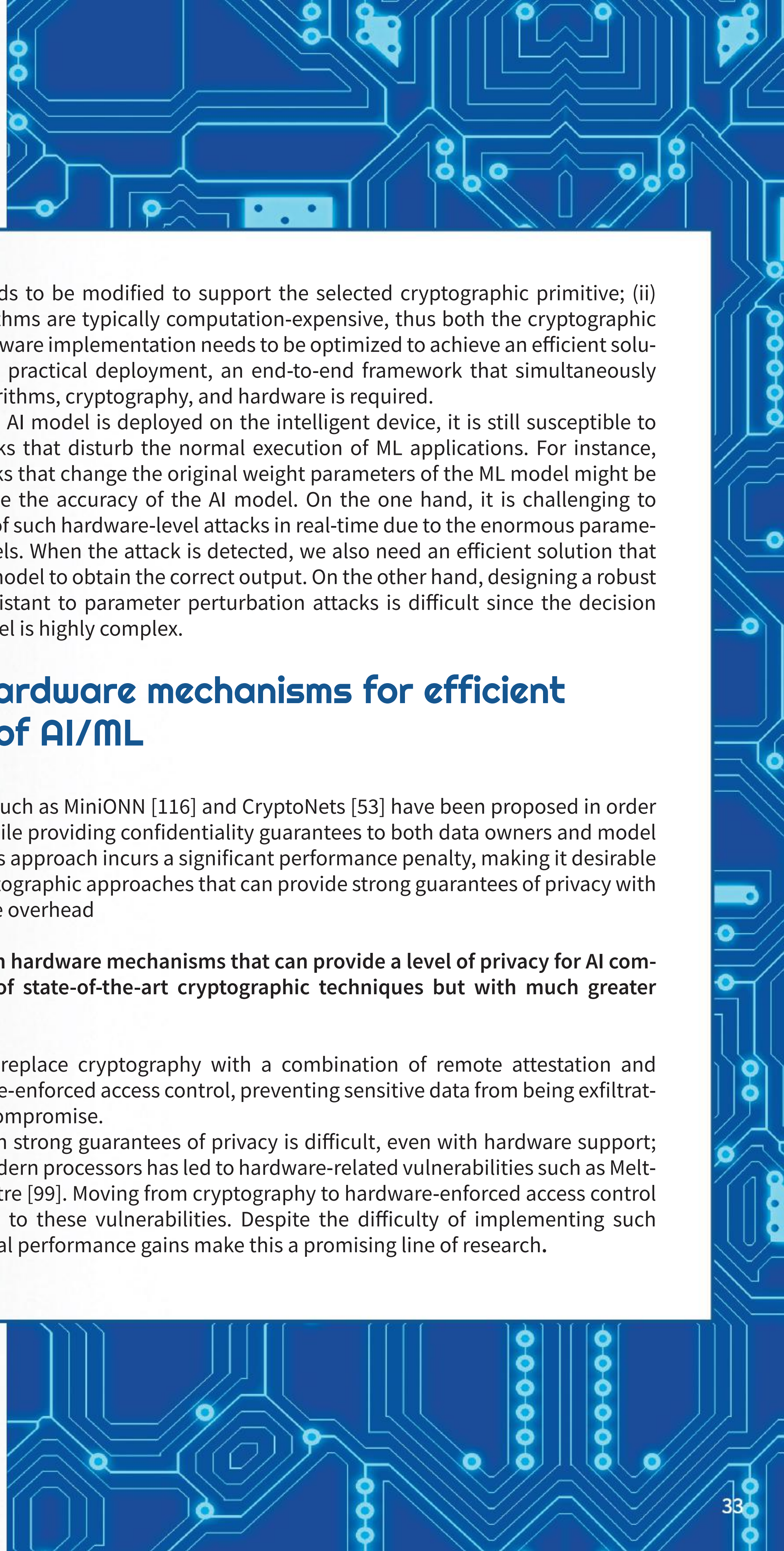
4.2 Cryptographic Accelerator for Privacy-preserving federated learning

Developing lightweight, privacy-preserving ML (PPML) inference approaches requires the designer to explore optimizations in both algorithm and hardware level. We discuss four challenges of designing accelerators for efficient PPML below.

The first challenge in designing hardware accelerators for privacy-preserving federated learning (FL) is assessing the hardware resources currently available on Intel processors and identifying potential scope of improvement. Moreover, our initial study has revealed that for shorter computations, the access time of the accelerators like QAT may become dominant over the actual computation time. To mitigate this issue and obtain the best possible performance, the computations need to be performed in large batches. This will require adapting the current software based implementations to exploit the inherent parallelism. Furthermore, we need to profile the batch size vs runtime characteristic to develop an automated methodology that will help decide on the best possible configuration of the workload.

The performance of different cryptographic primitives vary according to the required computation. It is common practice to adopt a mixed protocol solution where the best protocol is chosen for a particular task and secure conversion between them is performed when necessary. Designing hardware accelerators for individual primitives comes up with their own sets of challenges – reconfigurability based on the different security/performance parameters, management of the increased memory requirement caused by secure execution, reducing the effects of hardware constraints on the memory access, to name a few. On top of these, deployment of a mixed protocol system on a hardware accelerator makes the management of different primitives a daunting task. There are two possible approaches. In the first approach, each primitive is placed on a separate chip. Following this approach, the communication and conversion between the protocols needs to be managed in the software which may result in unacceptable increase in communication between the processor and accelerators. In the second approach, all the primitives reside on the same chip. This means that a large part of the control logic has to be on chip and may not have the flexibility of a software implementation. However, combining different primitives on the same chip brings the opportunity to save resources by merging some common lower level computations. While this approach has the potential to significantly reduce the resource usage, identifying such computation, and in some cases co-optimizing the different primitives to increase the intersection of their respective operations poses a significant challenge.

A holistic private AI application has three components: the AI algorithm, the cryptographic algorithm that ensures the privacy of the AI model, and the underlying hardware computing platform. However, existing design methods for AI applications mainly focus on the first component while leaving the integration of cryptography as well as hardware acceleration as an afterthought. Performing co-optimization of algorithm, cryptography, and hardware is challenging for the following reasons:



(i) The AI model needs to be modified to support the selected cryptographic primitive; (ii) Cryptographic algorithms are typically computation-expensive, thus both the cryptographic primitive and its hardware implementation needs to be optimized to achieve an efficient solution; (iii) To facilitate practical deployment, an end-to-end framework that simultaneously incorporates ML algorithms, cryptography, and hardware is required.

After a private AI model is deployed on the intelligent device, it is still susceptible to hardware-level attacks that disturb the normal execution of ML applications. For instance, memory trojan attacks that change the original weight parameters of the ML model might be performed to degrade the accuracy of the AI model. On the one hand, it is challenging to detect the existence of such hardware-level attacks in real-time due to the enormous parameter space of ML models. When the attack is detected, we also need an efficient solution that repairs/patches the model to obtain the correct output. On the other hand, designing a robust ML model that is resistant to parameter perturbation attacks is difficult since the decision boundary of the model is highly complex.

4.3 Novel hardware mechanisms for efficient protection of AI/ML

Cryptographic tools such as MiniONN [116] and CryptoNets [53] have been proposed in order to allow inference while providing confidentiality guarantees to both data owners and model owners. However, this approach incurs a significant performance penalty, making it desirable to consider non-cryptographic approaches that can provide strong guarantees of privacy with minimal performance overhead

Can we design hardware mechanisms that can provide a level of privacy for AI comparable to the use of state-of-the-art cryptographic techniques but with much greater efficiency?

One approach is to replace cryptography with a combination of remote attestation and fine-grained hardware-enforced access control, preventing sensitive data from being exfiltrated in the event of a compromise.

Providing such strong guarantees of privacy is difficult, even with hardware support; the complexity of modern processors has led to hardware-related vulnerabilities such as Meltdown [115] and Spectre [99]. Moving from cryptography to hardware-enforced access control exposes private data to these vulnerabilities. Despite the difficulty of implementing such schemes, the potential performance gains make this a promising line of research.

5 RESEARCH ON VALIDATION AND DEPLOYMENT

In this section we explore how the envisioned research will be validated in real-world deployment scenarios and how we can provide software frameworks to simplify deployment on a larger scale.

5.1 Research on Risk Sharing through Exchange

In the following, we elaborate on use cases that will be developed within the Private-AI Institue to demonstrate the applicability of the developed security and privacy solutions for distributed AI in practice.

Besides optimizing and training ML for specific use cases, our case studies aim to drive and validate our research:

What are the requirements of specific real-world ML usage scenarios? To what extent do our research results and tools demonstratably satisfy these requirement? What gaps and further improvements can be identified?

Sharing information about cybersecurity risks that were observed by other organizations or at other locations can greatly improve the security of computing systems through more effective threat detection and by means of reducing incident response time and increasing awareness about newly emerging threats. Furthermore, the risk exchange framework can enable more experienced entities with a sufficient level of security expertise to share their knowledge about attacks and defense strategies with less experienced parties.

Nowadays, information exchange about security threats can be achieved through the use of Cyber-Threat Intelligence (CTI) systems [140] that rely on sharing of security-relevant data. For example, shared data may include the record of previously observed incidents, e.g., when they took place, what organizations and platforms were affected, what kind of vulnerability was exploited, and what was the attack outcome. Other examples of shared information are malicious IP addresses, suspicious domain names, and malware signatures. However, according to a study [152] on 1000 IT professionals, around 70% of the respondents mentioned threat intelligence is too complex to provide actionable intelligence.

The Private-AI Institute will explore a new kind of risk information to be shared by the CTI system, which will be expressed in a form of machine learning models:

How can Cyber-Threat Intelligence systems use distributed machine learning to predict risks without leaking specific information about past incidents?

The goal is to improve automation and facilitate fast propagation of threat intelligence information and its effective utilization by all involved parties. The challenges in realizing such a system lie in various security and privacy concerns of involved entities, who might be mutually untrusted. For instance, attack vectors such as data and model poisoning and model sabotage are relevant and need to be taken into consideration. Furthermore, exchange of ML models may leak information about training data, which is crucial if such data inlcudes privacy-sensitive information.

5.2 Towards Improved Malware Detection using Federated ML

5.2.1 An Introduction to Malware Analysis, Detection, and Classification.

There is an increasing volume of new malware every year. For this reason, effective and fully automated malware detection is an important requirement to guarantee system safety and user protection.

Our goal is to develop automated procedures that are able to detect if a file is malware and identifies the family to which it belongs. There are various ways to achieve these objectives going from human reasoning to fully automated techniques. A very popular approach is based on the use of distinctive information known as a signature. The signature of a malware is its DNA. This is what we can use to recognize it among a set of cleanware as well as to distinguish it from other malware.

Signature-based malware detection normally proceeds in two phases. In the training phase, a number of samples of the malware to be detected are obtained. Then, a database of clean binaries, or goodware, that should not be detected as malware, is built. The signatures of each malware and goodware samples are extracted and compared to try to find common characteristics of the malware signatures that do not appear in the goodware signatures. The characteristics that distinguish the malware from the goodware are used to build a signature for the malware. In the classification phase, this signature is then compared against similar signatures of malware to determine whether the unclassified binary's signature indicates malicious behavior.

The main challenges to implement the procedures above are (1) to acquire databases of both malware and cleanware, (2) to develop methodologies to compute signatures that are powerful enough to distinguish malware from cleanware, (3) to extract and generalize signatures so that they can detect larger malware families beyond the specific instances inside the training set, and (4) to efficiently enable malware detection using the obtained signatures.

We now examine the related research question in detail.

5.2.2 Improving Malware Signatures for better Detection.

Our first research question is as follows:

How to design an efficient, correct, and precise procedure for malware detection? How to generate representative signatures from existing families? How to detect such signature in a given file?

A series of recent works (see [21] for a survey) have proposed a new approach to malware classification. The methodology uses concolic execution [71, 165] to extract a symbolic representation of many behaviors of a malware sample. These behaviours are used to reconstitute the System Call Dependency Graph (SCDG) of the sample. The approach is repeated for each malware of a given family, each generating an SCDG for that malware. The GSPAN [189] machine learning algorithm is used to idenfity common features between these SCDGs. Such commonalites become the signature for the malware family that indicates the key common behaviour. When a new sample needs to be examined, one uses concolic execution to extract its SCDG and then apply GSPAN on both this SCDG and the SCDG of each family. The new sample belongs to the family for which GSPAN finds enough similarities. Observe that other machine learning algorithms can be used. Observe also that concolic analysis can be used to directly assist malware analysists in their daily life [17]. Consequently, any improvement in the global methodology will also have a positive impact on this practical work. Recent results have showed that the learning-based approach can outpeform existing malware analysis techniques on a wide range of case studies (see [21, 158] for illustration). The use of symbolic analysis permits to overcome the limit of static analysis based machine learning outlined in [9]. We believe that those results are just the beginning of a promising research that will drastically improve malware detection. This opens the door to explore a wide range of research and application opportunities exploiting private federated learning.

The above procedure contains several implementation secrets and challenges that open a wide range of research directions. For example, due to the curse of dimensionality concolic execution can only extract a representation of certain behaviors of the system. Thus fine tuning the procedure so that it extracts behaviors of interest that are suitably representative for the sample program's behaviour can vastly improve malware detection and classification outcomes. In addition, learning algorithms such as GSPAN should be parameterized in an appropriate way to provide suitable generality without over-fitting. A preliminary study of those two research questions has been presented in [163].

5.2.3 Efficiency and Effectiveness of ML-based Malware Detection.

Is machine learning for malware detection a viable procedure for industry? Is the approach faster, more robust, more precise? What are the metrics to achieve success and adoption?

The end goal is to reach industrial deployment using this approach and take advantage of distributed and heterogeneous systems. One major step to achieving this is to improve the efficiency and the accuracy of the concolic procedure. This should include taking care of both obfuscation [146] and Virtualization [159] that are known as being penalizing for the solvers embedded in the concolic analyzer. In addition to improving the concolic execution itself, we will consider the integration of guided symbolic techniques such as Statistical Model Checking (SMC) [106]. This allows for both distibuting the analysis and guiding the concolic search towards unexplored behaviors and then exploiting statistical algorithms to combine the results. Further, to always guarantee greater efficiency, a distributed version of the algorithm should be developed that can be deployed on multiple diverse and easily available devices such as FPGAs. Such algorithm already exists for formal models of hardware [147]. The success metrics here are clear: the new approach shall be more precise than existing approaches, and this shall be done in a faster way.

5.2.4 Privacy-preserving Collaboration for Malware Detection using ML

How to guarantee that several contributors can participate without forcing them to reveal their secrets? How can we guarantee that contributors behave in an honest way and do not poison the resulting model?

To complement the above directions, a federated extension of this machine learning procedure can be developed. This can take advantage of the possibility for contributors to exchange their behavioural models (currently SCDGs) but not the procedure which allowed the models to be built and even less the malware database used. The idea is classic in the sense that each participant will train their own model and then the results will be aggregate in a central way.

What is more difficult is to define this aggregation knowing that individual models may be trained on wide range of architectures and types of files. Thus, even detals of the model that is to be shared may need to account for privacy of the sharer. Again, metrics of success are clear: the new approach shall be robust to adversarial attacks and guarantee the privacy of each participant. Observe that this approach has already been considered for the very restricted case of machine learning applied to android malware detection [61]. Authors observed that, in addition to adversarial attacks, one major challenge will be to guarantee that local data are not corrupted [112].

5.3 Deployment Challenges

5.3.1 Availability of Data for Training

Training deep learning models require large amounts of data, which might be challenging to achieve in practice due to privacy concerns and regulations such as General Data Protection Regulation (GDPR) in the European Union. The data collection is especially challenging at the time of model development, when the users cannot benefit from the developed model yet and, hence, have no incentive to provide their consent for data collection. Oftentimes such a collection needs to be performed with the help of paid test users, which naturally limits the amount of data that can be collected.

The problem related to lack of data may even appear for large datasets. In particular, if the dataset is imbalanced, i.e., the number of observations per class is not equally distributed, the problem of lack of data will affect the minority class, for which there are fewer samples. The problem of imbalanced data can appear in such applications as detection of rare diseases or natural disasters like earthquakes.

These challenges can be adressed with the help of open source datasets (if such datasets exist for the target task), through the use of syntethically generated data, and by means of applying transfer learning methods. Synthetic data can be created with the help of data generation algorithms, such as agent-based modeling that explains an observed behavior, and then reproduces random data using the same model.

Transfer learning enables one to train an initial model on the related task for which the sufficient amount of data is available, and fine-tune the model with the training using small amount of data on a target task. That's said, transfer leanrning transfers knowledge from the domain of related task to the domain of target task, and reduces the amount of data required for training in the target domain. In security applications, transfer learning was already applied to solve the imbalanced data issue in intrusion detection [62], vulnerability detection [117, 120, 142] and IoT attack detection systems [184].

5.3.2 Lack of Transparency

Complex machine and deep learning algorithms, also referred to as black-box models, lack transparency meaning that it is often hard to explain why they made a certain prediction. The more complex the model is, the more adjustable model parameters one needs to configure, and there is no one-to-one mapping between input features and parameters. As a result, it may be challenging to figure out what an algorithm learned during training and which of the data points have more significant influence on the outcome of classification.(LRP) method [14] outputs a heatmap over the input features that indicates their relevance to the model output. DeepLIFT [146] framework additionally treats positive and negative contributions differently, which improves effectiveness in revealing dependencies. Most approaches were evaluated using Convolutional Neural Networks (CNNs), while their applicability to other networks remains open.

Lack of transparency limits the explainability of ML models – or, the extent to which their internals can be understood by humans. For instance, deep learning models for vulnerability detection in smart contracts [82, 120, 201] can classify contracts into vulnerable and non-vulnerable categories and even detect vulnerability types, yet do not help to understand why such a classification was made and, hence, to identify the vulnerable code that needs to be fixed.

The area of research that deals with improving the explainability of ML models is Explainable AI (XAI). It aims to develop methods for black-box models to make it possible to understand the reasons behind model predictions. For instance, the Local Interpretable Model-Agnostic Explanations (LIME) method [154] outputs a binary vector along with a classification decision indicating if an input feature contributed to classification.

5.3.3 Bias in Training Data and Fairness

Training on biased data is likely to result in a biased model, hence it represents a serious deployment challenge. There are many reasons for having a bias in data, as systematically analyzed by Mehrabi et al. [129]. For instance, using imbalanced data can create biases against minority classes. Furthermore, datasets are collected and processed by humans, who may bring their stereotypical and prejudicial bias in data. Additionally, bias can be introduced through poorly designed data collection surveys that could lead participants to answer questions wrongly (e.g., when they lack the knowledge to answer correctly or do not want to provide the correct answer). Moreover, bias may be caused by incorrect selection of samples for training from a large population, which may result in a dataset that does not adequately represent the world the model will operate on.

The challenge of bias in training data is tackled by the area of research for fair machine learning. Numerous methods were developed for solving problems of fairness in different domains of machine learning, such as classification [66, 75, 95, 102, 131], clustering [15, 42], and adversarial learning [107, 188].

learning. Numerous methods were developed for solving problems of fairness in different domains of machine learning, such as classification [66, 75, 95, 102, 131], clustering [15, 42], and adversarial learning [107, 188].

The challenge of bias in training data will likely get more significance in the future, especially because more and more people without deep technical knowledge are involved in building machine learning models and deploying them in a large variety of applications. This trend increases the risk of biased models penetrating critical areas of our society, such as medicine, law, and security applications.

5.3.4 Concept Drift

Concept drift is a phenomena that affects scenarios where the probability distributions of a popoulation from which the data set was sampled from change over time. The conventional approach to deal with this problem is to retrain the model once it experiences significant performance degradation. This strategy is, however, not agile enough for security applications, which have to deal with adversaries who change their attack strategies rapidly to evade detection by ML-based systems.

To deal with the concept drift in adversarial settings, Thomas et al. [177] propose to retrain the model continuously during operation. Kantchelian et al. [96] suggest keeping human operators in the loop to deal with model aging. Maggi et al. [122] and Jordaney et al. [92] develop methods for detection of concept drift in deployed models, which enables re-training before the model significantly degrades. Almeida et al. [155] propose to tackle the concept drift challenge by using dynamic classifier selection for testing each instance. They show that selecting the most promising classifier/ensemble can deal with drifts regardless of how the selected classifier was trained. While proposed solutions were shown effective in targeted application scenarios (e.g., malware detection [92] or identification of malicious URLs in the web [177]), the question if they generalize to other use cases still remain open.

5.3.5 Joint Deployment of Multiple Security and Privacy Solutions.

Much of the research in ML security and privacy has focused on a specific challenge, such as protecting training data against membership inference, or resistance to adversarial examples, or making it possible to establish model ownership. But in a real deployment scenario, several such protection mechanisms need to be deployed simultaneously. Understanding the impact of one specific protection mechanism on others, and optimizing multiple mechanisms for joint deployment is an interesting and relevant open problem.

5.3.6 Lack of Computational Resources.

Training of machine learning models is a computationally-intensive process. Especially this is true for Deep Neural Networks with high learning capacity that have millions of parameters that are tuned when processing millions of data samples. Accelerating their training is a major challenge in practice and solutions range from hardware acceleration (cf. Section 4) to techniques for parallel and distributed deep learning [18, 98]. Large-scale training could further benefit from a combi-nation of parallelized training with cloud computing through elasticity and rapid availability of computational resources in clouds. Yet, cloud solutions for AI training are not mature enough to offer secure and privacy-preserving training, which poses additional deployment challenges. To enable trustworthy training in cloud environments, cloud providers need to deploy multiple security and privacy solutions jointly, which is not yet achieved in practice, as we explained in Section 5.3.5..

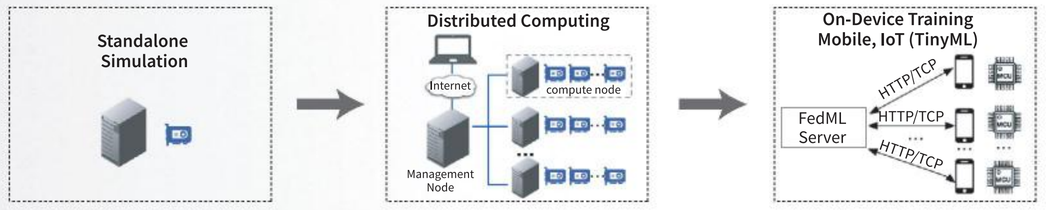
5.4 Open Source Frameworks and Tooling

To enable wide use of our research results, it is important to provide prototypes and then mature selected prototypes into software that is sufficiently mature for wider use. Since each research covers different aspects of the PrivateAI landscape, this can lead to piece-meal prototyping. To overcome this risk, we plan to also invest into open source frameworks and tooling. This answers the following research question:

How can software components be optimally packaged and integrated to enable efficient and usable deployment in practice?

An ideal research library for federated optimization research should support various platforms for realistic evaluation, obtaining realistic system performances like training time, communication cost, and computational cost.
(1) It should keep pace with advances in academia, supporting various newly published federated optimization algorithms.
(2) Write once, run everywhere. It is better to support seamless cross-platform migration with one time code writing. Researchers in optimization theory can focus on the innovations of algorithms, and let the library do the implementation.
(3) The programming interface should be flexible to enable diverse network topologies, flexible information exchange among workers/clients, and various training procedures.
(4) Supporting diverse datasets and models to fairly evaluate federated optimization algorithms is also a plus.

The diagram in Figure 4 illustrates this design goal. A representative research library aiming at these goal is FedML [78]. If researchers can first finish their algorithmic concept in a standalone simulation supported by FedML. They can then easily try their algorithms in a larger scale datasets and models in distributed computing platforms. Afterwards, for IoT platform, there is no need to reimplement the code in another programming language like Java and C++. Simply reusing the code at the distributed computing, FedML can support the IoT on-device training. With such a pipeline, theory-oriented optimization researchers can also obtain system-wise performance evaluation.



• Single process
• Runs on any server
• Good for small models (LR, 2CNN, Bi-LSTM)
• And small datasets (FMNIST, Shakespeare)

• Multiple-processes in parallel
• Can be seamlessly distributed across nodes
• Train big models (ResNet, MobileNet, Efficient Net, Transformer-based)
• And large datasets (CIFAR10/100, Google Landmark, COCO, ImageNet

• Multiple-processes explicit message passing
• Great for resource constrained edg
devices (Smartphone and IoT)-low memory, low computational power, limited communication band width
• Python-centric simplicity

Figure 4: An ideal research library for federated optimization research [64]

There are also serveral other federated learning libraries in the recent years. Distributed training libraries in PyTorch Distributed [150], TensorFlow [5], MXNet [40], and distributed training-specialized libraries such as Horovod [166] and BytePS [151] are designed for distributed training in data centers. Although simulation-oriented federated learning libraries such as TensorFlow-Federated (TFF)[85], PySyft [156], and LEAF [30] are developed, they only support centralized topology-based federated learning algorithms like FedAvg [128] or FedProx [157]. Furthermore, they only provide low-level communication APIs (e.g., TTF) or simulate a federation of nodes using a single machine, making them unsuitable or difficult to develop federated learning algorithms that require the exchange of auxiliary information and customized training procedures. Production-oriented libraries such as FATE [190] and PaddleFL [121] are released by industry. However, they are not designed as flexible frameworks that allow topology changes or allow drop-in replacement of different kinds of averaging algorithms.These features are necessary to support algorithmic innovation for open federated learning problems.

A further step to close the gap between research and production engineering is to support system deployment and on-device training runtime engine. In practice, frequent and large scale deployment of updates, monitoring, and debugging is challenging; running ML work-loads on end user device is hampered by the lack of a portable, fast, small footprint, and flexible runtime engine for on-device training. Research libraries like FedML definitely should support these two core modules. For more detailed challenges and practice in these topics, we refer to the guidance provided by [94, Section 7].

REFERENCES

[1] Protecting VM Register State with SEV-ES. https://www.amd.com/system/files/TechDocs/Protecting%20VM%20Register%20State%20with%20SEV-ES.pdf, 2017.[Online; accessed 28-October-2020].

[2] AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf, 2020. [Online; accessed 28-October-2020].

[3] Arm Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A. https://developer.arm.com/documentation/ddi0615/latest/, 2021. [Online; accessed 27-July-2021].

[4] Intel Trust Domain CPU Architectural Extensions. https://software.intel. com/content/dam/develop/external/us/en/documents-tps/intel-tdx-cpu-architectural-specification.pdf, 2021. [Online; accessed 27-July-2021].

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[6] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[7] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In 27th {USENIX} Security Symposium ({USENIX} Security 18), pages 1615–1631, 2018.

[8] Alekh Agarwal, John Langford, and Chen-Yu Wei. Federated residual learning. arXiv preprint arXiv:2003.12880, 2020.

[9] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. When malware is packin' heat; limits of machine learning classifiers based on static analysis features. In 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020. The Internet Society, 2020.

[10] Nitin Agrawal, Reuben Binns, Max Van Kleek, Kim Laine, and Nigel Shadbolt. Exploring design and governance challenges in the development of privacy-preserving computation. CoRR, abs/2101.08048, 2021.

[11] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. in Neural Information Processing Systems NIPS, 2017.

[12] Ittai Anati, Shay Gueron, Simon P. Johnson, and Vincent R. Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In Workshop on Hardware and Architectural Support for Security and Privacy (HASP), 2013.

[13] ARM Limited. Security technology: building a secure system using TrustZone technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2008.

[14] Buse Gul Atli, Yuxi Xia, Samuel Marchal, and N. Asokan. Waffle: Watermarking in federated learning. 2021.

[15] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. Scalable fair clustering. In Proceedings of the 36th International Conference on Machine Learning (Proceedings of MachineLearning Research), page 405–413, 2019.

[16] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pages 2938–2948. PMLR, 26–28 Aug 2020.

[17] Roberto Baldoni, Emilio Coppa, Daniele Cono D'Elia, and Camil Demetrescu. Assisting malware analysis with symbolic execution: A case study. In Shlomi Dolev and Sachin Lodha, editors, Cyber Security Cryptography and Machine Learning - First International Conference, CSCML 2017, Beer-Sheva, Israel, June29-30, 2017, Proceedings, volume 10332 of Lecture Notes in Computer Science, pages 171–188. Springer, 2017.

[18] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. ACM Computing Surveys, 2019.

[19] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, Machine Learning and Knowledge Discovery in Databases, pages 387–402, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[20] A. Binder, S. Bach, G. Montavon, K.-R. Müller, and W. Samek. Layer-wise relevance propagation for deep neural network architectures. In Information Science and Applications (ICISA), 2016.

[21] Fabrizio Biondi, Thomas Given-Wilson, Axel Legay, Cassius Puodzius, and Jean Quilbeuf. Tutorial: An overview of malware detection and evasion techniques. In Tiziana Margaria and Bernhard Steffen, editors, Leveraging Applications of Formal Methods, Verification and Validation. Modeling - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part I, volume 11244 of Lecture Notes in Computer Science, pages 565–586. Springer, 2018.

[22] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 118–128, 2017.

[23] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. MP2ML: a mixed-protocol machine learning framework for private inference. In Melanie Volkamer and Christian Wressnegger, editors, ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25-28, 2020, pages 14:1–14:10. ACM, 2020.

[24] Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. Ngraph-he: A graph compiler for deep learning on homomorphically encrypted

data. In Proceedings of the 16th ACM International Conference on Computing Fron-tiers, CF ’19, page 3–13, New York, NY, USA, 2019. Association for Computing Machinery.

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

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

[27] Ferdinand Brasser, Patrick Koeberl, Brahim El Mahjoub, Ahmad-Reza Sadeghi, and Christian Wachsmann. TyTAN: Tiny Trust Anchor for Tiny Devices. In IEEE/ACM Design Automation Conference (DAC), 2015.

[28] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software Grand Exposure: SGX Cache Attacks Are Practical. In USENIX Workshop on Offensive Technologies (WOOT), 2017.

[29] Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, Tegan Maharaj, Pang Wei Koh, Sara Hooker, Jade Leung, Andrew Trask, Emma Bluemke, Jonathan Lebensold, Cullen O’Keefe, Mark Koren, Théo Ryffel, JB Ru-binovitz, Tamay Besiroglu, Federica Carugati, Jack Clark, Peter Eckersley, Sarah de Haas, Maritza Johnson, Ben Laurie, Alex Ingerman, Igor Krawczuk, Amanda Askell, Rosario Cammarota, Andrew Lohn, David Krueger, Charlotte Stix, Peter Henderson, Logan Graham, Carina Prunkl, Bianca Martin, Elizabeth Seger, Noa Zilberman, Seán Ó hÉigeartaigh, Frens Kroeger, Girish Sastry, Rebecca Kagan, Adrian Weller, Brian Tse, Elizabeth Barnes, Allan Dafoe, Paul Scharre, Ariel Herbert-Voss, Martijn Rasser, Shagun Sodhani, Carrick Flynn, Thomas Krendl Gilbert, Lisa Dyer, Saif Khan, Yoshua Bengio, and Markus Anderljung. Toward trustworthy ai development: Mechanisms for supporting verifiable claims, 2020.

[30] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečn`y, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097, 2018.

[31] Rosario Cammarota, Matthias Schunter, Anand Rajan, Fabian Boemer, Ágnes Kiss, Amos Treiber, Christian Weinert, Thomas Schneider, Emmanuel Stapf, Ahmad-Reza Sadeghi, Daniel Demmler, Huili Chen, Siam Umar Hussain, Sadegh Riazi, Farinaz Koushanfar, Saransh Gupta, Tajan Simunic Rosing, Kamalika Chaudhuri, Hamid Nejatollahi, Nikil Dutt, Mohsen Imani, Kim Laine, Anuj Dubey, Aydin Aysu, Fateme Sadat Hosseini, Chengmo Yang, Eric Wallace, and Pamela Norton. Trustworthy ai inference systems: An industry research view, 2020.

[32] Shicong Cen, Huishuai Zhang, Yuejie Chi, Wei Chen, and Tie-Yan Liu. Con-ver-gence of distributed stochastic variance reduced methods without sampling extra data. IEEE Transactions on Signal Processing, 68:3976–3989, 2020.

[33] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. arXiv preprint arXiv:1912.11279, 2019.

[34] Cheng Chen, Ziyi Chen, Yi Zhou, and Bhavya Kailkhura. Fedcluster: Boost-ing the convergence of federated learning via cluster-cycling. arXiv preprint arXiv:2009.10748, 2020.

[35] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876, 2018.

[36] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H. Lai. SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution. In IEEE European Symposium on Security and Privacy (EuroS&P), 2019.

[37] Huili Chen, Cheng Fu, Bita Darvish Rouhani, Jishen Zhao, and Farinaz Koushan-far. Deepattest: an end-to-end attestation framework for deep neural networks. In 2019 ACM/IEEE 46th Annual International Symposium on Computer Architec-ture (ISCA), pages 487–498. IEEE, 2019.

[38] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In IJCAI, pages 4658–4664, 2019.

[39] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushan-far. Deepmarks: A secure fingerprinting framework for digital rights man-age-ment of deep learning models. In Proceedings of the 2019 on International Conference on Multimedia Retrieval, pages 105–113, 2019.

[40] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274, 2015.

[41] Xiangyi Chen, Tiancong Chen, Haoran Sun, Zhiwei Steven Wu, and Mingyi Hong. Distributed training with heterogeneous data: Bridging median and mean based algorithms. arXiv preprint arXiv:1906.01736, 2019.

[42] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. Proportionally fair clustering. In International Conference on Machine Learning, page 1032–1041, 2019.

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

[44] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In International Conference on the Theory and Application of Cryptology and Information Security, pages 409–437. Springer, 2017.

[45] Christopher A Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. Capc learning: Confidential and private collaborative learning. 2021.

[46] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Proceedings of the 36th International Conference on Machine Learning, 2019.

[47] Fergus Dall, Gabrielle De Micheli, Thomas Eisenbarth, Daniel Genkin, Nadia Heninger, Ahmad Moghimi, and Yuval Yarom. CacheQuote: Efficiently Re-covering Long-term Secrets of SGX EPID via Cache Attacks. Transactions on Cryptographic

Hardware and Embedded Systems, 2018.

[48] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, pages 485–497, 2019.

[49] Tom Woller David Kaplan, Jeremy Powell. AMD Memory Encryp-tion. https://developer.amd.com/wordpress/-media/2013/12/AMD_Memory_ Encryption_Whitepaper_v7-Public.pdf, 2016. [Online; accessed 28-October-2020].

[50] Daniel Demmler, T. Schneider, and Michael Zohner. Aby - a framework for efficient mixed-protocol secure two-party computation. In NDSS, 2015.

[51] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In Advances in Neural Information Processing Systems, 2020.

[52] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021.

[53] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to en-crypted data with high throughput and accuracy. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, vol-ume 48 of ICML'16, page 201–210, 2016.

[54] Ahmed Roushdy Elkordy and A. Salman Avestimehr. Secure aggregation with heterogeneous quantization in federated learning. preprint arXiv:2009.14388, 2020.

[55] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In International Conference on Knowledge Discovery and Data Mining, 2004.

[56] Dmitry Evtyushkin, Jesse Elwell, Meltem Ozsoy, Dmitry V. Ponomarev, Nael B. Abu-Ghazaleh, and Ryan Riley. Iso-X: A Flexible Architecture for Hardware-Managed Isolated Execution. In IEEE/ACM International Symposium on Microar-chitecture (MICRO), 2014.

[57] Dmitry Evtyushkin, Ryan Riley, Nael CSE Abu-Ghazaleh, ECE, and Dmitry Ponomarev. BranchScope: A New Side-Channel Attack on Directional Branch Predictor. In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2018.

[58] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized feder-ated learning: A meta-learning approach. In Advances in Neural Information Processing Systems, 2020.

[59] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 1322–1333, 2015

[60] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. arXiv preprint arXiv:1808.04866, 2018.

[61] Rafa Galvez, Veelasha Moonsamy, and Claudia Díaz. Less is more: A privacy-respecting android malware classifier using federated learning. CoRR, abs/2007.08319, 2020.

[62] Aryya Gangopadhyay, Iyanuoluwa Odebode, and Yelena Yesha. A domain adaptation technique for deep learning in cybersecurity. In OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", pages 221–228, 2019.

[63] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? arXiv preprint arXiv:2003.14053, 2020.

[64] Zahra Ghodsi, Akshaj Kumar Veldanda, Brandon Reagen, and Siddharth Garg. CryptoNAS: Private inference on a relu budget. In Advances in Neural Informa-tion Processing Systems, volume 33, pages 16961–16971, 2020.

[65] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In Advances in Neural Information Processing Systems, 2020.

[66] Naman Goel, Mohammad Yaghini, and Boi Faltings. The cost of fairness in binary classification. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[67] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. STOC '87, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.

[68] Ian Goodfellow and Nicolas Papernot. The challenge of verification and testing of machine learning. http://www.cleverhans.io/security/privacy/ml/2017/06/14/verification.html, 2017. [Online accessed on March 31st, 2021].

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

[70] Ben Gras, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. Translation Leak-aside Buffer: Defeating Cache Side-channel Protections with TLB Attacks. In USENIX Security Symposium, 2018.

[71] Fabio Gritti, Lorenzo Fontana, Eric Gustafson, Fabio Pagani, Andrea Continella, Christopher Kruegel, and Giovanni Vigna. SYMBION: interleaving symbolic with concrete execution. In 8th IEEE Conference on Communications and Network Security, CNS 2020, Avignon, France, June 29 - July 1, 2020, pages 1–10. IEEE, 2020.

[72] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of dis-tributed learning in Byzantium. In International Conference on Machine Learning, pages 3521–3530. PMLR, 2018.

[73] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck R Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. arXiv preprint arXiv:1910.13598, 2019.

[74] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. arXiv:2002.05516, 2020.

[75] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS), page 3315–3323, 2016.

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

[77] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. 2020.

[78] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xi-aoy-ang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518, 2020

[79] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11993–12002, 2020.

[80] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via resampling. arXiv preprint arXiv:2006.09365, 2020.

[81] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using Innovative Instructions to Create Trustworthy Software Solutions. In Workshop on Hardware and Architectural Support for Security and Privacy (HASP), 2013.

[82] TonTon Hsien-De Huang. Hunting the ethereum smart contract: Color-inspired inspection of potential attacks. arXiv preprint arXiv:1807.01868, 2018.

[83] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving

[84] Siam U Hussain, Mohammad Samragh, Xinqiao Zhang, and Farinaz Koushanfar. On the Application of Binary Neural Networks in Oblivious Inference. In Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[85] Alex Ingerman and Krzys Ostrowski. TensorFlow Federated, 2019.

[86] Intel Corporation. Intel Software Guard Extensions Programming Refer-ence. https://software.intel.com/sites/default/-files/managed/48/88/329298-002. pdf, 2014.

[87] Mojan Javaheripi, Mohammad Samragh, Gregory Fields, Tara Javidi, and Farinaz Koushanfar. Cleann: Accelerated trojan shield for embedded neural networks. In 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pages 1–9. IEEE, 2020.

[88] Malhar S. Jere, Tyler Farnan, and Farinaz Koushanfar. A taxonomy of attacks on federated learning. IEEE Security Privacy, 19(2):20–28, 2021.

[89] Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. Deepreduce: Relu reduction for fast private inference. In International Confer-ence on Machine Learning, 2021.

[90] Hengrui Jia, Christopher A Choquette-Choo, and Nicolas Papernot. En-tangled watermarks as a defense against model extraction. arXiv preprint arXiv:2002.12200, 2020.

[91] Yihan Jiang, Jakub Konečn`y, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488, 2019.

[92] Roberto Jordaney, Kumar Sharad, Santanu K. Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In USENIX Security Symposium, 2017.

[93] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. Gazelle: A low latency framework for secure neural network inference. In 27th USENIX Security Symposium (USENIX Security 18), pages 1651–1669, 2018.

[94] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cor-mode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gib-bons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2019.

[95] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudiceremover regularizer. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 35–50, 2012.

[96] Alex Kantchelian, Sadia Afroz, Ling Huang, Aylin Caliskan Islam, Brad Miller Michael Carl Tschantz, Rachel Greenstadt, Anthony D. Joseph, and J. D. Tygar. Approaches to adversarial drift. In Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security (AISec), page 99–110, 2013.

[97] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In Advances in Neural Information Processing Systems, 2019.

[98] Jin Kyu Kim, Qirong Ho, Seunghak Lee, Xun Zheng, Wei Dai, Garth A. Gibson, and Eric P. Xing. Strads: a distributed framework for scheduled model parallel machine learning. In Proceedings of the European Conference on Computer Systems (EuroSys), pages 1–17, 2016.

[99] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In 40th IEEE Symposium on Security and Privacy (S&P'19), 2019.

[100] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. TrustLite: A Security Architecture for Tiny Embedded Devices. In European Conference on Computer Systems (EuroSys), 2014.

[101] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. NIPS Workshop on Private Multi-Party Machine Learning (2016).

[102] Emmanouil Krasanakis, Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, and Yiannis Kompatsiaris. The cost of fairness in binary classification. In Proceedings of the 2018 WorldWide Web Conference (WWW), page 853–862, 2018.

[103] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and

Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. arXiv preprint arXiv:1910.12366, 2019.

[104] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified Robustness to Adversarial Examples with Differential Privacy. arXiv preprint arXiv:1802.03471, 2018.

[105] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing. In USENIX Security Symposium, 2017.

[106] Axel Legay, Anna Lukina, Louis-Marie Traonouez, Junxing Yang, Scott A. Smolka, and Radu Grosu. Statistical model checking. In Bernhard Steffen and Gerhard J. Woeginger, editors, Computing and Software Science - State of the Art and Perspectives, volume 10000 of Lecture Notes in Computer Science, pages 478–504. Springer, 2019.

[107] Blake Lemoine, Brian Zhang, and M. Mitchell. Mitigating unwanted biases with adversarial learning. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society (AIES), 2018.

[108] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. arXiv preprint arXiv:1809.03113, 2018.

[109] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differen-tially private meta-learning. In International Conference on Learning Represen-tations, 2020.

[110] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 1544–1551, 2019.

[111] Ping Li, Jin Li, Zhengan Huang, Tong Li, Chong-Zhi Gao, Siu-Ming Yiu, and Kai Chen. Multi-key privacy-preserving deep learning in cloud computing. Future Generation Computer Systems, 74:76–85, 2017.

[112] Shenghui Li, Edith C. H. Ngai, Fanghua Ye, and Thiemo Voigt. Auto-weighted robust federated learning with corrupted data sources. CoRR, abs/2101.05880, 2021.

[113] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127, 2018.

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

[115] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In 27th USENIX Security Symposium (USENIX Security 18), 2018.

[116] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. Oblivious neural network predic-tions via minionn transformations. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 619–631, 2017.

[117] Shigang Liu, Guanjun Lin, Lizhen Qu, Jun Zhang, Olivier DeVel, Paul Montague, and Yang Xiang. CD-VulD: Cross-domain vulnerability discovery based on deep domain adaptation. In IEEE Transactions on Dependable and Secure Computing, 2020.

[118] Qian Lou, Yilin Shen, Hongxia Jin, and Lei Jiang. {SAFEN}et: A secure, accurate and fast neural network inference. In International Conference on Learning Representations, 2021.

[119] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. arXiv preprint arXiv:1912.00888, 2019.

[120] Oliver Lutz, Huili Chen, Hossein Fereidooni, Christoph Sendner, Alexandra Dmitrienko, Ahmad Reza Sadeghi, and Farinaz Koushanfar. ESCORT: Ethereum Smart COntRacTs Vulnerability Detection using Deep Neural Network and Transfer Learning. ArXiv | arXiv:2103.12607v1, mar 2021.

[121] Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. Frontiers of Data and Domputing, 1(1):105–115, 2019.

[122] Federico Maggi, William Robertson, Christopher Kruegel, and Giovanni Vigna. Protecting a moving target: Addressing web application concept drift. In International Symposium on Recent Advances in Intrusion Detection (RAID), page 21–40, 2009.

[123] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. arXiv preprint arXiv:2104.10706, 2021.

[124] Amirhossein Malekijoo, Mohammad Javad Fadaeieslam, Hanieh Malekijou, Morteza Homayounfar, Farshid Alizadeh-Shabdiz, and Reza Rawassizadeh. Fedzip: A compression framework for communication-efficient federated learn-ing. arXiv preprint arXiv:2102.01593, 2021.

[125] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:2002.10619, 2020.

[126] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative Instructions and Software Model for Isolated Execution. In Workshop on Hardware and Architectural Support for Security and Privacy (HASP), 2013.

[127] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. 2017.

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

[129] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019.

[130] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In 2019 IEEE Symposium on Security and Privacy (SP), pages 691–706. IEEE, 2019.

[131] Aditya Krishna Menon and Robert C. Williamson. The cost of fairness in binary classification. In Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research), 2018.

[132] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In 29th USENIX Security Symposium (USENIX Security 20), pages 2505–2522. USENIX Association, August 2020.

[134] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. CacheZoom: How SGX Amplifies The Power of Cache Attacks. In Conference on Cryptographic Hardware and Embedded Systems (CHES), 2017.

[135] Payman Mohassel and Peter Rindal. ABY 3: A mixed protocol framework for machine learning. In ACM SIGSAC Conference on Computer and Communications Security, pages 35–52, 2018.

[136] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable priva-cy-preserving machine learning. In 38th IEEE Symposium on Security and Privacy, pages 19–38. IEEE, 2017.

[137] Mathias Morbitzer, Manuel Huber, Julian Horsch, and Sascha Wessel. SEVered: Subverting AMD's virtual machine encryption. In Proceedings of the 11th European Workshop on Systems Security, 2018.

[138] Sasi Kumar Murakonda and Reza Shokri. Ml privacy meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning. arXiv preprint arXiv:2007.09339, 2020.

[139] Milad Nasr, Reza Shokri, and Amir 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), pages 739–753. IEEE, 2019.

[140] National Institute of Standards and Technology (NIST). Guide to cyber threat information sharing. 2016.

[141] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Ahmad-Reza Sadeghi, Thomas Schneider, et al. Flguard: Secure and private federated learning. arXiv preprint arXiv:2101.02281, 2021.

[142] Van Nguyen, Trung Le, Tue Le, Khanh Nguyen, Olivier DeVel, Paul Montague, Lizhen Qu, and Dinh Phung. Deep domain adaptation for vulnerable code function identification. In International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2019.

[143] NIST. Trojai leaderboard. https://pages.nist.gov/trojai/, 01 2021.
[144] Job Noorman, Pieter Agten, Wilfried Daniels, Raoul Strackx, Anthony Van Her-rewege, Christophe Huygens, Bart Preneel, Ingrid Verbauwhede, and Frank Piessens. Sancus: Low-cost Trustworthy Extensible Networked Devices with a Zero-software Trusted Computing Base. 2013.

[145] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In 25th {USENIX} Security Symposium ({USENIX} Security 16), pages 619–636, 2016.

[146] Mathilde Ollivier, Sébastien Bardin, Richard Bonichon, and Jean-Yves Marion. How to kill symbolic deobfuscation for free (or: unleashing the potential of path-oriented protections). In David Balenson, editor, Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019, pages 177–189. ACM, 2019.

[147] Kosuke Oshima, Takeshi Matsumoto, and Masahiro Fujita. Hardware implemen-tation of BLTL property checkers for acceleration of statistical model checking. In Jörg Henkel, editor, The IEEE/ACM International Conference on Computer-Aided Design, ICCAD'13, San Jose, CA, USA, November 18-21, 2013, pages 670–676. IEEE, 2013.

[148] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[149] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018.

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

[151] Yanghua Peng, Yibo Zhu, Yangrui Chen, Yixin Bao, Bairen Yi, Chang Lan, Chuan Wu, and Chuanxiong Guo. A generic communication scheduler for distributed dnn training acceleration. In Proceedings of the 27th ACM Symposium on Operating Systems Principles, pages 16–29, 2019.

[152] Ponemon Institute LLC. The value of threat intelligence: Annual study of north american & united kingdom companies, 2019.

[153] Saurav Prakash and Amir Salman Avestimehr. Mitigating byzantine attacks in federated learning. arXiv preprint arXiv:2010.07541, 2020.

[154] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 1135–1144, 2016.

[155] Paulo R.L.Almeida, Luiz S.Oliveira, Alceu S.Britto Jr., and Robert Sabourin. Adapting dynamic classifier selection for concept drift. Expert Systems with Applications, 104:67–85, 2018.

[156] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning. arXiv preprint arXiv:1811.04017, 2018.

[157] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. ArXiv, abs/1812.06127, 2018.

[158] Najah Ben Said, Fabrizio Biondi, Vesselin Bontchev, Olivier Decourbe, Thomas Given-Wilson, Axel Legay, and Jean Quilbeuf. Detection of mirai by syntac-tic and behavioral analysis. In Sudipto Ghosh, Roberto Natella, Bojan Cukic, Robin S. Poston, and Nuno Laranjeiro, editors, 29th IEEE International Sympo-sium on Software Reliability Engineering, ISSRE 2018, Memphis, TN, USA, October
15-18, 2018, pages 224–235. IEEE Computer Society, 2018.

[159] Jonathan Salwan, Sébastien Bardin, and Marie-Laure Potet. Symbolic deobfusca-tion: From virtualized code back to the original. In Cristiano Giuffrida, Sébastien Bardin, and Gregory Blanc, editors, Detection of Intrusions and Malware, and Vulnerability Assessment - 15th International Conference, DIMVA 2018, Saclay, France, June 28-29, 2018, Proceedings, volume 10885 of Lecture Notes in Computer Science, pages 372–392. Springer, 2018.

[160] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[161] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. IEEE transactions on neural networks and learning systems, 31(9):3400–3413, 2019.

[162] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), 2017.

[163] Stefano Sebastio, Eduard Baranov, Fabrizio Biondi, Olivier Decourbe, Thomas Given-Wilson, Axel Legay, Cassius Puodzius, and Jean Quilbeuf. Optimizing symbolic execution for malware behavior classification. Comput. Secur., 93:101775, 2020.

[164] F. Seide, H. Fu, Jasha Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. in Inter speech, 2014.

[165] Koushik Sen. Concolic testing: a decade later (keynote). In Harry Xu and Walter Binder, editors, Proceedings of the 13th International Workshop on Dynamic Analysis, WODA@SPLASH 2015, Pittsburgh, PA, USA, October 26, 2015, page 1. ACM, 2015.

[166] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. arXiv preprint arXiv:1802.05799, 2018.

[167] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pages 1310–1321, 2015.

[168] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In 2017 IEEE Symposium on Security and Privacy (SP), pages 3–18. IEEE, 2017.

[169] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning impor-tant features through propagating activation differences. In 34th International Conference on Machine Learning (ICML), pages 3145–3153, 2017.

[170] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In Advances in Neural Information Processing Systems, 2017.

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

[172] Raoul Strackx, Job Noorman, Ingrid Verbauwhede, Bart Preneel, and Frank Piessens. Protected Software Module Architectures. In ISSE Securing Electronic Business Processes. 2013.

[173] Lili Su and Jiaming Xu. Securing distributed machine learning in high dimen-sions. arXiv preprint arXiv:1804.10140, 2018.

[174] Haijian Sun, Xiang Ma, and Rose Qingyang Hu. Adaptive federated learning with gradient compression in uplink noma. IEEE Transactions on Vehicular Technology, 2020.

[175] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. Distributed mean estimation with limited communication. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 3329–3337. PMLR, 06–11 Aug 2017.

[176] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In International Conference on Learning Representations, 2014.

[177] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time URL spam filtering service. In IEEE Symposium on Security and Privacy, page 447–462, 2011.

[178] Florian Tramer, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Joern-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In Proceedings of the 37th International Conference on Machine Learning, 2020.

[179] Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. arXiv preprint arXiv:1806.03287, 2018.

[180] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. In Advances in Neural Information Processing Systems, 2019.

[181] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In 25th {USENIX} Security Symposium ({USENIX} Security 16), pages 601–618, 2016.

[182] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In USENIX Security Symposium, 2018.

[183] Stephan van Schaik, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Mali-cious Management Unit: Why Stopping Cache Attacks in Software is Harder Than You Think. In USENIX Security Symposium, 2018.

[184] Ly Vu, Quang Uy Nguyen, Diep N Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz. Deep transfer learning for IoT attack detection. IEEE Access, 8:107335–107344, 2020.

[185] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. Proceedings on Privacy Enhancing Technologies, 2019(3):26–49, 2019.

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

[187] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. in Neural Information Processing Systems NIPS, 2017,.

[188] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. Fairgan: Fairness-aware generative adversarial network. In 2018 IEEE International Conference on Big Data (Big Data), page 570–575, 2018.

[189] Xifeng Yan and Jiawei Han. gspan: Graph based substructure pattern mining.

In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan, pages 721–724. IEEE Computer Society, 2002.

[190] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 13(3):1–207, 2019.

[191] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended)? In FOCS, pages 162–167, 1986.

[192] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. ICML, 2018.

[193] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 5693–5700, 2019.

[194] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learn-ing by local adaptation. arXiv preprint arXiv:2002.04758, 2020.

[195] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In International Conference on Learning Representations, 2021.

[196] Qiao Zhang, Chunsheng Xin, and Hongyi Wu. Privacy preserving deep learning based on multi-party secure computation: A survey. IEEE Internet of Things Journal, 2021.

[197] Xinqiao Zhang, Huili Chen, and Farinaz Koushanfar. TAD: trigger approxima-tion based black-box trojan detection for AI. CoRR, abs/2102.01815, 2021.

[198] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. arXiv preprint arXiv:2005.11418, 2020.

[199] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582, 2018.

[200] Ligeng Zhu and Song Han. Deep leakage from gradients. In Federated Learning, pages 17–31. Springer, 2020.

[201] Yuan Zhuang, Zhenguang Liu, Peng Qian, Qi Liu, Xiang Wang, and Qinming He. Smart contract vulnerability detection using graph neural networks. 2020.

IMAGES REFERENCES

[1] Circuit board Image by Gerd Altmann from Pixabay
[2] CPU and computer chip concept from iStock
[3] Monitor binary Image by Gerd Altmann from Pixabay
[4] Binary code Image by Gerd Altmann from Pixabay
[5] CPU with Binary Numbers and Blueprint from iStock
[6] Camera Image by Gerd Altmann from Pixabay
[7] Board Circuit Image by Gerd Altmann from Pixabay
[8] Artificial Intelligence Image by Gerd Altmann from Pixabay
[9] Web Network by Image by Gerd Altmann from Pixabay
[10] Conductor Circuit board Image by Gerd Altmann from Pixabay
[11] Artificial Intelligence by Vecteezy
[12] Artificial Intelligence brain Image by Gerd Altmann from Pixabay
[13] Technological Face Image by Gerd Altmann from Pixabay
[14] Eye Biometric Image by Gerd Altmann from Pixabay
[15] Block chain network Image by Pete Linforth from Pixabay