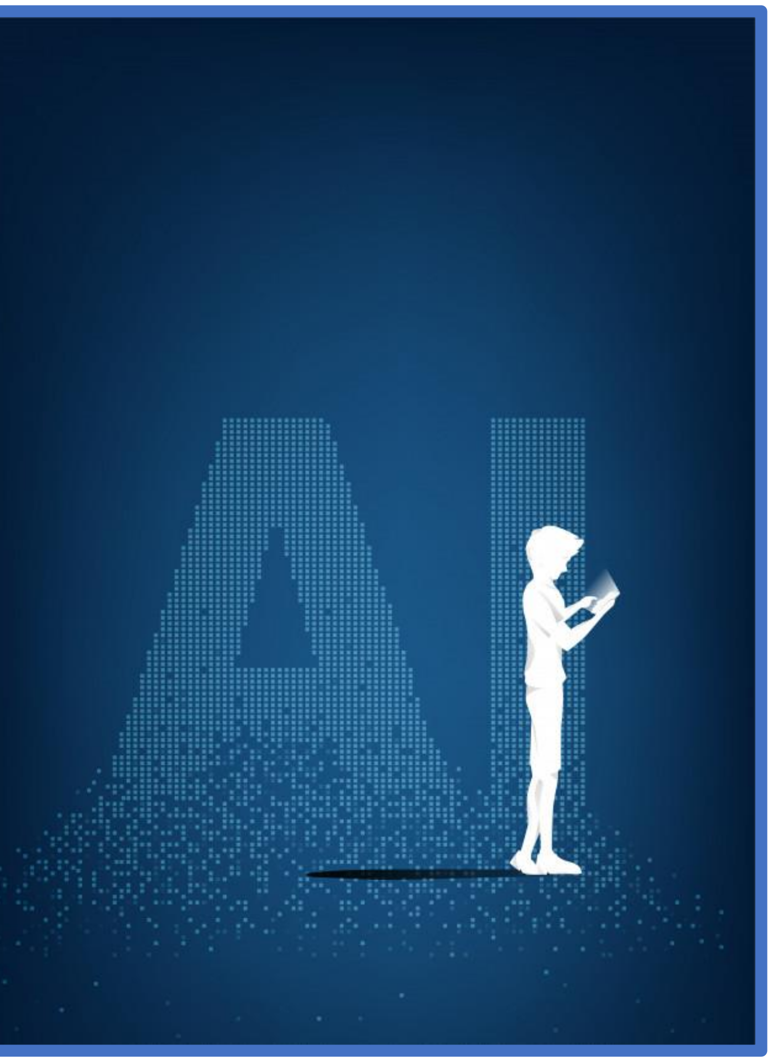
Joost Verbraeken

Supervised by Dr. Pouwelse and Dr. de Vos

Practical Byzantine-resilient, yet decentralized federated learning

A thesis written by Joost Verbraeken examining the state-of-the-art and proposing Pro-Bristle, a new technique to improve byzantine-resilience in asynchronous non-i.i.d. settings



Abstract

Introduction

(glossary)

1.1. Why machine learning?

Statistics is a branch of mathematics concerned with explain and gathering insights from historical data, for example to understand consumer preferences, determine the core components of human personalities, or to illustrate how economic policies affect the society. However, statistics has its limitations: it can be notoriously hard to create a highly accurate model, the statistical techniques available for making predictions about the future are relatively limited, and the use-cases are limited to clearly specified domains (in contrast to fuzzy domains such as the generation of completely new songs based on previous songs). Machine learning gained traction over the last few decades thanks to the discovery of several novel and powerful techniques, such as Support Vector Machines and Random Forests. These techniques are used for a wide variety of tasks such as multimedia recommendation, handwriting recognition, speech-to-text systems, digital translators, etc., but these methods depend on carefully extracted features and often yield sub-optimal accuracy. In the last decade, neural networks became popular thanks to their versatility, ability to learn to extract proper features themselves, and highly effective predictions. Neural networks consist of a series of layers, each with a number of artificial neurons. Each neuron is linked to a number of other neurons in the previous/next layer by a connection associated with a certain weight. When a neuron is activated, it checks if the combined activation it gets from all nodes in the last layer exceeds a certain threshold (i.e. its bias) and then propagates this activation to the nodes in the next layer to which it is connected. These weights and biases are constantly adjusted in the neural network through a process called back-propagation so that the network actually starts to “learn”.

1.2. Why distributed learning?

Training a neural network on a single machine is possible when the amount of data is relatively limited. However, for more complex applications (such as self-driving cars, image recognition, or music generation) the amount of training data required can easily exceed the maximum capacity of a single machine. [25] describes in detail how a new scientific field called *Distributed Learning* aims to distribute the training data and/or the neural network across many nodes (also called *clients*, *machines*, or *peers*), often implemented by combining an army of *slave* nodes that perform the calculations given by a *master* node (often called the *parameter server*). The master node communicates with the slave nodes, iteratively combines their results, and updates the individual slave nodes with the result. This technique gained rapid popularity and is nowadays the backbone behind most industry-grade machine-learning implementations [26] (although there is also a multitude of other distributed learning architectures, each with their own advantages / disadvantages [25]).

1.3. Why federated learning?

Although distributed learning is highly effective in teaching neural networks to accomplish complex tasks, it still depends on as much data as possible to get the most accurate results. The data to train popular neural networks often comes from smartphones, which on one hand produce enormous quantities of data which enables improved representation and generalization of machine-learning models, but on the other hand pose a significant problem because of three key reasons: (a) sending all kinds of data generated by smartphones over the internet consumes a lot of bandwidth, (b) training a neural network on data generated by billions of smartphones is computationally extremely intensive for a single *master* node, and (c) sending potentially sensitive information across the internet to the cloud raises privacy concerns, and is in certain cases not even allowed by several regulations such as the US HIPAA laws [27] and Europe’s GDPR law [28].”

These challenges motivated the development of a new type of distributed learning called *federated learning* where smartphones update the neural network with their data on-device and send back to the server the updated model rather than their data [29]. “Federated Learning brings the code to the data, instead of the data to the code” [30]. From this perspective, federated learning is closely related to Mobile Edge Computing (MEC) in the sense that computations are pushed to the edge of the network to reduce bandwidth consumption and improve privacy.

Thanks to these advantages, federated learning is used for a wide range of applications nowadays including next-word-prediction on keyboards such as Gboard [31-37], “wake word detection which enables voice assisting apps to detect wake word without risking exposure of sensitive user data” [38], speech recognition [39], wireless communications [40, 41], security applications (such as malware classification [42], human activity recognition [43], anomaly detection [44], and intrusion detection [45]), transport applications (such as data sharing between self-driving cars [46-48], preventing data leakage [49], traffic flow prediction [50], and the detection of attacks in aerial vehicles [51]), object detection [52], and health applications [53-57]

Federated learning environments have a number of notable characteristics [30, 58]:

- **Massively distributed**: the total number of nodes can easily be in the order of millions **(or even billions in the case of GBoard)**.
- **Unbalanced / non-i.i.d. data**: different peers may possess different classes distributed in different ratios.
- **Unreliable**: since federated learning is often applied to smartphones, the nodes may go offline/online at any moment and the network connection may be slow.

However, we would like to build a system that not only seeks to properly handle the challenges imposed by the characteristics mentioned above but is also practical to be used in real-life. Therefore, we want to design the system around three additional principles:

1.4. Additional design principles

Decentralized

Practically all federated learning systems employ a centralized architecture characterized by a central trusted authority [30], often called a *parameter server*, that communicates with all nodes (generally smartphones). The machine learning process is as follows: (1) definition of the ML model (e.g. a CNN or RNN) by the developer in terms of hyperparameters, (2) distribution of the model by the master to the slaves, (3) local training of the model by each slave, (4) aggregation of all models by the master, (5) iteratively repeating steps 2, 3, and 4 [59]. The training process may stop when a sufficiently high accuracy is obtained or may be trained continuously as more data becomes available to the slaves.

Unfortunately, such a centralized architecture has several significant drawbacks [59-61]. The parameter server is not only a single point of failure susceptible to crashes or hacks, but it may also become a performance bottleneck when there are too many devices participating in the network. This problem accelerated research that aims to remove the parameter server entirely and train the network in a decentralized fashion [60, 62-64] where each node both trains and aggregates incoming parameters to learn the model [65]. Since these nodes are independent rather than being instructed by a master node, we will not refer to them as slaves but just as "nodes" or "peers" in the rest of the paper.

Byzantine-resilient

Even the most efficient and stable decentralized federated learning system is worthless for practical applications when the model can be ruined by Byzantine nodes, where *Byzantine* refers to the broadest class of faults in system components. The malicious model can be accidental (e.g. crashes, faulty sensors, computation errors, noisy transmission, nodes that are lagging behind, non-i.i.d. data, etc.; all of which the probability to occur at least once grows with the number of nodes [66]) or malicious on purpose (e.g. data poisoning or sophisticated model poisoning attacks; more on this in Section Data poisoning vs model poisoning attacks). This scenario, where the nodes don't know which of the other nodes are benign or corrupt, is the infamous "Byzantine Generals Problem" [67]. Without a Byzantine Fault Tolerance (BFT) mechanism, even a single malicious node that only takes moderate values to make its actions hard to detect can significantly degrade the performance of the federated model [68, 69].

Unfortunately, it's relatively easy to initiate a simple poisoning attack where a node just sends incorrectly updated parameters on purpose because of 3 major reasons [70]: (1) authentication mechanisms are often not feasible since federated systems often span across countries, (2) because the whole purpose of federated learning is to keep the training data private, it is impossible to audit the reliability of the training data, and (3) in real-world situations that dataset is often (very) non-i.i.d. which makes it challenging to distinguish between an attack and an unusual data class.

Byzantine resilience can be divided into weak and strong Byzantine resilience[71]. Weak f-Byzantine resilience implies that despite the presence of f Byzantine nodes, the network will almost surely converge to a certain value. Strong f-Byzantine resilience implies that the network does not just converge in the presence of f Byzantine nodes, but also converges to approximately the right value. In this thesis we will focus on strong Byzantine resilience since this is the only provable solution against attackers with significant resources.

An interesting observation made by Haykin [72] is that a "mild" Byzantine worker can actually improve the performance of the system. This has to do with the fact that the optimization function of a neural network is practically never convex and has many local optima. By providing the "wrong" direction, a little bit of noise (or a "mild" Byzantine attack in that regard) can pull the optimization function out of a local minimum so that the network can converge to a better global minimum [73-75]. This is also the reason Stochastic Gradient Descent (SGD) works so well: a randomly drawn sample is inherently more noisy (higher variance) than the average of all samples [76] and may pull the network out of a local minimum. However, stronger Byzantine attacks can pull the network away from the global minimum in which case they ruin the network's performance.

Asynchronous

Distributed learning can happen either in synchronous rounds or in an asynchronous manner.

Although synchronous algorithms may seem to be the natural choice for federated learning (illustrated by the fact that the first federated learning protocol FedAvg [30] and influential subsequent research such as [77] were synchronous), there are a number of limitations as summed up by [78]:

- Some devices assumed to participate in the synchronous round may randomly drop-out, because of the volatile nature of the end-devices.
- Devices that just joined the network and are ready/willing to participate have to wait until the start of the next synchronization and are thus under-utilized.
- Some devices may be significantly slower than other devices due to more training data, less processing power, or less bandwidth.
- When a device is too slow to finish its iteration before the next synchronization, it has to overwrite its local model with the new global model and all of its progress is lost.

For this reason, numerous asynchronous approaches have been proposed [58, 59, 78-84]. These methods either overprovision clients and then accept the first x updates, dynamically update the synchronization time or amount of work per node, or use weighted averaging based on the staleness of incoming model updates. A common reoccurrence is that all of these approaches are dependent on a centralized parameter server, while in this thesis we aim to harness the advantages of a completely decentralized network. Truly decentralized protocols such as [85-87] are in practice always asynchronous because there is no server to synchronize training rounds, but these papers assume that the maximum staleness of peers is bounded.

[88] aims to achieve byzantine consensus in decentralized asynchronous networks, but they do not consider a situation where nodes can randomly join/exit the network.

[89] presents FedProx, a modification of the FedAvg algorithm that is supposed to tackle heterogeneity in FL by considering a variation of computational power and other differences between devices. However, its performance turns out to be sub-par in later research.

Another method based on FedAvg is SAFA [78], which aims to harness the potential efficiency gains of an asynchronous setting while using (a) a pace steering mechanism to reduce the impact of stale models and straggling clients, and (b) an aggregation algorithm that exploits a cache structure to reduce communication costs.

An approach that outperforms the former ones is presented in [90] that performs layer-wise matching and averaging of channels/neurons. It sends at the start of each training rounds global model matching results to the clients and adds additional neurons to the local models to achieve better performance.

To prevent the global model from drifting too much towards the fastest nodes, [91] proposes a mechanism to reduces this impact.

1.5. Key contributions

Overcoming all of the challenges described above is highly non-trivial, since distributing a computation over many peers induces a substantial risk of local crashes, computation errors, stalled processes, biased local datasets, but also possibly Byzantine workers trying to significantly degrade the performance of the system. Especially defending against Byzantine failures in a decentralized non-i.i.d. environment is challenging, while the literature on this topic is relatively sparse. Therefore, we will propose in this thesis Pro-Bristle, a novel algorithm that improves – to the best of the author's knowledge – the current state-of-the-art by achieving (a) decentralized, (b) Byzantine-resilient, (c) asynchronous, and (d) non-i.i.d. federated learning. Our main contribution can be summarized as follows:

- We propose to use a combination between per class performance-based filtering, SRA private-set intersection cardinality, CWR*, deep transfer learning, and a carefully crafted weighted averaging function to achieve a high degree of Byzantine-resilience in non-i.i.d. environments.
- We use per class performance-based filtering, a buffer of recent models, and an exploration vs exploitation strategy to account for highly asynchronous situations.
- We evaluate Pro-Bristle and 5 other GARs in a large number of environments, with varying datasets, degrees of asynchrony, attacks, number of attackers, degrees of non-i.i.d.-ness, usage of transfer learning, and communication protocols.

TODO beste grafiek eruitlichten

There are several types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning. In this paper we will focus on arguably the most popular one, namely the supervised learning: a type of machine learning where the model should learn the correlation between the input data and their corresponding labels based on a set of training samples. From the two main instances of supervised learning (namely classification and regression) we will look at classification problems.

1.6. Overview of paper

First, in Section 2 we aim to give the reader an overview of the relatively unknown and emerging scientific field of Federated Learning. We will look at several types of Byzantine attacks, Byzantine-resilient defenses, and solutions to apply these techniques in a non-i.i.d. setting.

In Section 3, we propose our solution called Pro-Bristle and explain how this gradient aggregation rule (GAR) improves the state-of-the-art, especially in highly Byzantine non-i.i.d. environments.

Subsequently, in Section 4 we describe the implementation of the algorithm and the whole experiment in more detail, including all attacks used to evaluate the GARS and the main issues we encountered while developing the software.

Thereafter, in Section 5 the performance of Pro-Bristle is compared with the performance of other GARs in a large variety of settings. We conclude by discussing the most interesting directions for further research in Section 6 and reiterating the main insights described in this thesis in Section 7.

2. Related Research

In this section, we first give the reader a short primer into federated learning. Then, we give a concise overview of the most important Byzantine attacks and segment them along two dimensions. Subsequently, we provide an extensive overview of Byzantine-resilient defense methods (since the main contribution of this thesis is also a Byzantine-resilient defense method) and categorize these methods into 5 distinct categories. Finally, we briefly examine popular algorithms to enable learning in non-i.i.d. environments.

2.1. Basic idea behind federated learning

In machine learning we aim to minimize the global cost function, risk function, loss function, or score $\ell(\theta)$ by finding the optimal model θ^*:

$$\theta^* = \underset{\theta}{arg\,min} \underset{(x,y)\,\in\, D}{\mathbb{E}} \ell(f_\theta(x), y) \qquad (1)$$

Where θ is the model, D is a distribution on $X\ x\ Y$ and $\ell(\theta; i)$ is the loss of model θ on dataset instance i. This loss function is a proxy for the actual error to be minimized, generally the negative log likelihood of the ground truth class in the case of a classification problem.

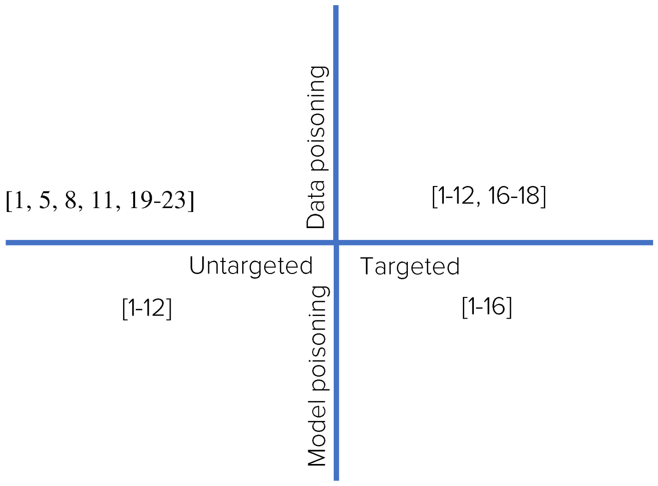
This optimization problem is known as *risk minimization* but solving this problem is unfortunately for more complex models intractable. Therefore, a technique called Empirical Risk Minimization (ERM) is commonly used where take an empirically obtained dataset M i.i.d. sampled from D. Then we can obtain an estimate of the optimal model by calculating:

$$\theta_n = \underset{\theta}{arg\,min} \frac{1}{|M|} \sum_{(x,y)\in M} \ell(f_\theta(x), y) \qquad (2)$$

To minimize this function, a popular technique is called Gradient Descent (GD) that iteratively takes the derivative of the loss function with respect to the training samples and then moves the hyperparameters in the direction of the gradient:

$$\theta^{t+1} = \theta^t - \lambda \nabla_\theta \ell(\theta; i) \qquad (3)$$

However, because the dataset may be large, gradient descent can take a long time to converge. A faster approach, used by almost all learning algorithms today, is to use Stochastic Gradient Descent (SGD) [92] where a subset (a *minibatch*) of the dataset is selected to update the parameters in a particular iteration [93, 94]. As a result, SGD produces faster but noisier updates than GD, but this noise is not necessarily a disadvantage because it also helps the algorithm to escape local minima. An important requirement for SGD to converge is that each minibatch is an unbiased sample of the actual distribution, which is typically achieved through uniform random sampling [95].



The most straight-forward way to apply stochastic gradient descent in a distributed or federated setting is to use a single master node (parameter server) that distributes and integrates the global model and a number of slave nodes that train the model that they obtained from the master node and send the result back to the master node [96, 97].

In distributed machine learning the most trivial implementation is called Bulk Synchronous Parallel [98] and in federated learning FedAvg [30]. FedAvg simply aggregates the models owned by the peers by coordinate-wise weighted averaging. It was introduced by Google [43] and is still researched extensively from both an applied and theoretical perspective [59].

2.2. Byzantine attacks

FedAvg, as described in the previous section, takes the average of all models at every iteration. Obviously, when a single Byzantine node transmits a model with extremely low or high values, the average significantly changes, and the model become worthless. Such an attack is easy to detect, but there are many more sophisticated attacks that can, even with a single Byzantine attacker, considerably reduce the model's performance and are much harder to detect [68, 99].

Byzantine attacks can be classified based on certain properties as either a data poisoning or a model poisoning attack, and as either an untargeted or a targeted attack.

Data poisoning vs model poisoning attacks

Data poisoning attacks such as [1-12, 19-23, 100-103] try to degrade the performance of the learnt model to such a degree that the model becomes worthless by training the network with dirty samples. They were introduced by [7] to destroy support vector machines and later extended to many other ML algorithms including neural networks. Without appropriate Byzantine-resilient defense mechanisms, a malicious agent can relatively easily manipulate the global model. The best researched type of attack is a convergence prevention attack [13] where the attacker wants to prevent the network from converging and reduce its accuracy up to a point that the model might be utterly ineffective indiscriminately for testing examples. One might think that sending completely random numbers is an effective attack. However, because the mean of completely random numbers is 0, the network will still converge when the standard deviation isn't too extreme [104] (in fact, adding noise to the parameters is a popular method called *differential privacy* that is used to improve the user's privacy[105-107]). A scenario where a malicious agent injects malicious data into a benign client's dataset (better known as a data injection attack) is also considered data poisoning. Another notable example of a data poisoning attack is a label-flip attack, where the labels of two or more classes are changed [5, 108, 109].

While data poisoning attacks are based on the manipulation of training data, model poisoning attacks such as [1, 2, 5, 8, 13-18, 68, 69, 110, 111] manipulate the model's parameters directly before sending it to other nodes. Consequently, every data poisoning attack can be imitated with a model poisoning attack [112], but model poisoning attacks give the attacker full control over every single parameter and can thus be much more effective as recent research has shown [6, 68, 99, 111]. They can even be used to replace the entire global model with a model of the attacker's choice (model replacement attack), given a carefully chosen scaling factor [5, 68]. However, there are also simple model poisoning attack such as the Gaussian attack, where some of the gradient vectors are replaced by random vectors sampled from a Gaussian distribution with large variances.

Untargeted vs targeted attacks

Another way to classify Byzantine attacks as done by [112] is to group them into untargeted and targeted attacks. Whereas untargeted attacks such as [1, 5, 8, 11, 19-23, 68, 69, 100, 110, 111] aim to prevent convergence and reduce the global model's accuracy [68, 69, 110, 111], targeted attacks such as [1-18] aim to alter the model's behavior in specific situations while keeping the total accuracy as high as possible to mislead Byzantine-defense mechanisms [5, 6]. Without proper defense mechanisms federated learning is susceptible to both untargeted and targeted attacks [108].

Targeted attacks are also sometimes called (semantic) backdoors, Trojan threats, or stealth attacks, where the backdoor can target either a single class (a label-flip attack) or a class of samples (e.g. an almost invisible attacker-chosen pattern of pixels, i.e. a trigger) causing an image to be classified incorrectly. A particularly effective attack is described by Bhagoji et al. [14] who use an alternating minimization strategy (alternating between training loss minimization and the boosting of updates for the malicious objective). A more sophisticated attack is proposed by Xie et al. in [16] who notes that all backdoor attacks until then used embeddings of the same global trigger pattern for all adversarial parties, called centralized backdoor attacks by the authors. They then propose distributed backdoor attacks (DBA) where the global trigger pattern is decomposed into local patterns and which is then embedded to different adversarial parties, thus making the attack harder to detect, easier to bypass robust aggregation rules, and more effective. In line with this contribution, [13] shows that targeted model poisoning attacks can become both significantly more effective and harder to detect when adversaries are able to collude.

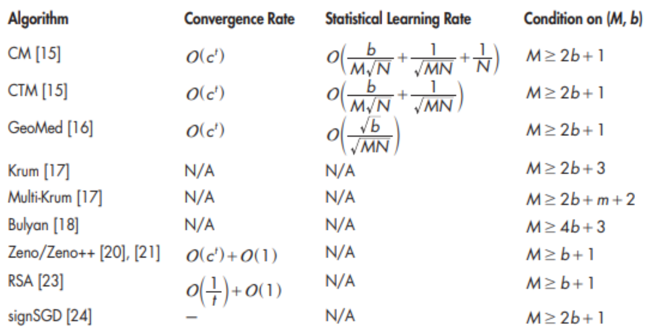
As mentioned in Section Byzantine-resilient, a little random noise can actually improve the convergence of stochastic gradient descent. That might lead one to think that simply clearing away large deviations might be an effective defense mechanism. However, [13] the authors show that this assumption is incorrect and propose another powerful attack "capable of defeating all state-of-the-art defenses" based on injecting values that are just within the perturbation range (the range of values that the Byzantine-defense mechanism allows).

Targeted attacks are hard to detect, because the accuracy of the model may not necessarily be impacted for any of the samples that any peer has available, but only for samples with, for example, a specific pattern that no-one except for the attacker knows about. More specifically, detecting backdoors in a model is an NP-hard problem, by a reduction from 3-SAT [113], and unlikely to be detected using gradient based techniques. [113] explains how it is relatively easy to develop a so-called *edge-case backdoor* which forces a model to consistently misclassify seemingly easy inputs that are unlikely to be part of the regular training data. Because these targeted model poisoning attacks only need to modify a small part of the model [112], they look quite similar as benign updates and require fewer adversaries than untargeted attacks, being already effective with even a single-shot attack under certain conditions [5].

2.3. Byzantine-resilient defenses

There are several types of Byzantine-resilient defense mechanisms that are in the literature often segmented into distance-based defenses (based on the calculation of some kind of distance between potential malicious attack vectors and some other vector(s), usually efficient but also vulnerable to elaborately designed Byzantine attacks [13, 111]) and performance-based defenses (based on testing the accuracy of a potentially malicious model on a small representative dataset, which is usually computationally quite intensive and clearly dependent on the availability of a test dataset but also usually quite effective) [114]. Another way of segmenting the these algorithms is based on whether or not they are centralized (require a central parameter server) or decentralized, or by their degree of dimensional Byzantine resilience [114] (namely, the maximum number of tolerated Byzantine workers).

To structure the myriad of Byzantine defense mechanisms which we will call Gradient Aggregation Rules (GARs) from now on in accordance with the literature, we will use more fine-grained categories in this thesis, based on the fundamental principle that the algorithms employ.



Algorithm	Convergence Rate	Statistical Learning Rate	Condition on (*M*, *b*)
CM [15]	$O(c^t)$	$O\left(\frac{b}{M\sqrt{N}} + \frac{1}{\sqrt{MN}} + \frac{1}{N}\right)$	$M \geq 2b+1$
CTM [15]	$O(c^t)$	$O\left(\frac{b}{M\sqrt{N}} + \frac{1}{\sqrt{MN}}\right)$	$M \geq 2b+1$
GeoMed [16]	$O(c^t)$	$O\left(\frac{\sqrt{b}}{\sqrt{MN}}\right)$	$M \geq 2b+1$
Krum [17]	N/A	N/A	$M \geq 2b+3$
Multi-Krum [17]	N/A	N/A	$M \geq 2b+m+2$
Bulyan [18]	N/A	N/A	$M \geq 4b+3$
Zeno/Zeno++ [20], [21]	$O(c^t)+O(1)$	N/A	$M \geq b+1$
RSA [23]	$O\left(\frac{1}{t}\right)+O(1)$	N/A	$M \geq b+1$
signSGD [24]	–	N/A	$M \geq 2b+1$

Distance-based screening

Screening potentially malicious incoming model updates for their distance with respect to the peer's own trusted model is by far the most popular to evade Byzantine attacks which should not come as a surprise. They are often highly efficient, do not depend on an extra dataset, special hardware features, or additional server, and they can do an excellent job to guard against relatively simple attacks. However, although this class of algorithms is effective against simple attacks such as Gaussian and label-flip attacks, they are doing a poor with regard to more advanced attacks [115]. This is due to an implicit and somewhat incorrect assumption made by distanced-based GARs, namely that close distances between model parameters implies similar performance. Additionally, gradient updates might differ significantly in a non-i.i.d. environment between nodes which result in large distances, leading distance-based GARs to reject these updates as outliers. Moreover, when the peer's won model is extremely stale, all incoming models are regarded as outliers making it hard for the stale model to catch up. Therefore, in the other sections other methods that are computationally much more expensive are evaluated that might be more effective than distance-based GARs, especially in non-i.i.d. and asynchronous settings.

Detecting outliers in non-distributed settings has been studied extensively for a long time [116], generally to be able to sanitize the data from poisoned or otherwise anomalous data [117]. In recent years, much progress has been made in terms of improved accuracy in high-dimensional settings [118-120]. For example, [12] uses a clustering technique to measure the difference between benign and malicious updates. However, these techniques are not suited for the distributed setting on which we are focusing.

A particularly influential algorithm is Krum[68] which selects the model that is most similar to all other models as the new global model. Even when the model being selected is malicious, the performance should not degrade too much in theory because it is close to all other models. Despite theoretical guarantees for the convergence for certain objective functions, Krum seems to perform quite bad in comparison to other algorithms [121] and often converges to an ineffective model [110]. The deficient performance stems from the ability of Byzantine workers to introduce a substantial change in a single parameter without significantly influencing the total distance due to the typically high dimensionality of the parameters [110]. [13] elaborates upon this insight and argues that, since only a single model is selected and even the best benign worker will have a few parameters that reside far from the mean, the GAR performs worse than other GARs that do integrate data from multiple models into the final model. The authors also briefly discuss Multi-Krum, which achieves similar accuracy at a faster rate by using an average of m local gradients obtained by Krum.

[99] presents two simple distance-based GARs, namely Coordinate-wise Trimmed Mean (CTM) (also evaluated by [122, 123]) which simply cuts off the smallest and largest b values in each dimension of the incoming vectors, and Coordinate-wise Median (CM) or Marginal Median which takes the median in every dimension. CM does not need at least *2b+1* values like CTM, but does incur a performance hit because every dimension must be sorted to obtain the median.

[114] also evaluated CM and compared it under con-convex settings with two other approaches, namely the Geometric Median and Mean around the Median. The Geometric Median is defined as [68, 69, 114]:

$$\underset{v \in \mathbb{R}^d}{argmin} \sum_{i=1}^{n} \|v - \tilde{v}_i\| \tag{4}$$

Which can be interpreted as the point for which the square distance to all other points in an n-dimensional space is minimized. The Mean around the Median is defined as the mean value of the $n - q$ indices closest to the median, where q is an arbitrary value. The authors find that Krum, Multi-Krum, and the Geometric Median perform worst, the Marginal Median has considerable variance, and the Mean around the Median performs best. The Geometric median not only performs poorly, but also dominates the training time in large-scale settings [124].

A variant on the *Geometric Median* is the (*Geometric) Median of Means* [69, 125-128], which first partitions all received vectors into *k* batches, then computes the mean for each batch, and finally takes the geometric median of the k batch means. [69] extends the techniques described in [129] with arbitrary/adversarial outliers. They only consider strongly convex losses which they try to remedy by using mini batches. However, their algorithm fails even when there is only a single Byzantine node in each mini-batch and is thus not very reliable.

Mhamdi et al. [110] try to combine the strengths of Krum and CM in Bulyan, a GAR that iteratively applies Krum to select a number of models and then a variant of CM to aggregate them into a single model. More specifically, Bulyan finds for every dimension the n parameters closest to the median and then takes their mean value. A notable disadvantage of Bulyan is its speed and the stringent condition that it imposes on the number of Byzantine nodes, namely $\#nodes \geq 4\,x\,\#nodes_{byzantine} + 3$. A year later, the authors extend Bulyan to Multi-Bulyan similarly to the extension of Krum to Multi-Krum, but unfortunately they don't report on the results [71].

In order to reduce the communication necessary for the aforementioned GARs, Bernstein et al. developed SignSGD [130] which only transmits the sign of every dimension of the gradient at every iteration. Since the global model is updated with an element-wise majority vote on the signs of the received gradients, the algorithm is in fact a median-based algorithm which makes it also robust against certain Byzantine failure and guarantees convergence given certain conditions imposed on the noise [131]. However, these conditions are typically not in order in a typical federated learning setting where the data is distributed non-i.i.d.[132]. Sohn et al. make SignSGD more robust against MITM-attacks, but do not address the case where nodes themselves are malicious[133].

In contrast to SignSGD, the work done by Li et al. [134], confusingly called RSA just like the cryptosystem, is able to handle heterogenous datasets in a Byzantine distributed setting and prevent incorrect gradient aggregation by letting every node store and update a local version of the global model which are then aggregated at the server by means of an ℓ_p-norm regularization term which regularizes the magnitudes of malicious messages.

An interesting distance-based method is described in [135] where the authors construct a graph where the nodes (representing models) are connected by a vertex only when their Euclidean distance is small enough and subsequently solve the maximum clique model to find the set of models that are similar to each other and therefore probably benign. Unfortunately, the authors only evaluate trivial label-flip attacks which make it hard to estimate the effectiveness of the algorithm in a more challenging environment.

All GARs described until now assume a federated setting where a single parameter server iteratively updates the global model. However, these algorithms do not translate well into a decentralized setting (which is the focus of this thesis) because decentralized GARs require consensus between all peers which is usually not required for distributed learning. To the best of our knowledge, there are only three papers that attempt to achieve Byzantine-resilient decentralized learning by adopting a truly distance-based strategy, namely ByRDiE [136], BRIDGE [137], and some extension of BRIDGE [138]. The CM algorithms described before (namely the ones presented in [99, 122, 123]) are suboptimal in vector-valued problems, because simply minimizing the objective function along each coordinate independent of all other coordinates yields the wrong solution (unless all dimensions are truly independent, which is generally not the case). This limitation is overcome in ByRdiE [136] by cyclically updating every coordinate one by one in a decentralized manner and subsequently applying trimmed-mean screening to obtain the final coordinate for each dimension. Given a strictly convex loss function, ByRDiE is proven to always converge (although not necessarily to an optimal solution). However, although ByRDiE might be efficient in terms of required training samples, ByRDiE is inefficient in terms of network communication because it only updates one coordinate at a time and the update step depends on the updates of other coordinates [121]. Therefore, [137] present BRIDGE which combines CTM with SGD (Stochastic Gradient Descent) to accomplish decentralized Byzantine-resilience with significantly less network communication for highly dimensional problems. The review paper [121] shows better performance for BRIDGE than for CTM, which is highly surprising because BRIDGE boils down to CTM in distributed environment. Upon closer examination this happens because the authors use a 0.7 connection ratio between the nodes to evaluate BRIDGE and just a $4\,x\,\#\max\,\_byzantine\_nodes + 1 = 4\,x\,2 + 1 = 9;\quad 9\,/\,20\,nodes = 0.45$ connection ratio between the nodes to evaluate CTM. [138] shows that the performance of BRIDGE can be improved by adding a total variation (TV) norm penalty to allow some outliers to be able to handle non-i.i.d. data. This probably reduces the ability of the algorithm to defend against noise attacks, but unfortunately the authors have conveniently omitted these results from the Results section. [139] also builds upon BRIDGE and extends the solution to non-i.i.d. settings, but does this by re-introducing the central server that we want to omit in a decentralized setting (more information about non-i.i.d. approaches will be discussed in Section 2.4).

One of the most recent papers about the topic is [140] which select the models with the smallest Euclidean norm to be averaged for the updated model, but this paper evaluates its performance by measuring the loss function, which is extremely noisy and relatively unreliable compared to an evaluation using a validation dataset. This makes it hard to properly estimate its performance.

Performance screening

Although distance-based screening methods can be quick and effective to filter out "unusual" models, they will not include benign models when these models are quite different from the other models (e.g. in a non-i.i.d. or highly asynchronous environment) and allow an attacker to let the model drift towards a bad solution. Performance-based solution such as [4, 117, 141-144] detect malicious models based on a negative impact on the model's accuracy given a test dataset. A major advantage of performance-based solutions is that whereas many other GARS assume that the number of adversarial workers is always less than half of the total number of workers, performance-based solutions typically ensure convergence even in the presence of a large number of adversaries [144, 146, 160, 200, 242]{Zhao, 2020 #138}. Of papers, we want to seriously criticize the paper written by Zhao et al. [143] because, aside from the fact that it contains serious grammar errors and completely incorrect references, it also includes a major error about when label-flipping attacks are preferred above backdoor attacks. The authors say that label-flipping attacks are more effective in a scenario where data samples with the same label are quite similar while the latter is more suitable for scenarios where samples with the same label are quite diverse. This is incorrect: you want to use label-flipping attacks as an effective way to fool or prevent convergence of a model without any serious byzantine-resilient defense mechanisms while you want to use backdoor attacks to trick the model to misclassify certain input data without letting anyone notice that you are malicious (see Section Byzantine attacks). The authors also assume that agents share which labels they own, which is absurd: the whole purpose of a federated learning environment is that the user's data (including the labels) stays private.

[141] presents RONI (Reject On Negative Influence, which removes training examples with a negative impact on the accuracy of the model) and [20] presents TRIM (which finds a subset of the training dataset given a pre-specified size and set of hyperparameters that maximizes the accuracy and is, according to the authors, more effective than RONI), both of which were originally intended to filter out bad training data on a single node. [111] converted and applied RONI and TRIM to a federated setting and found that in that setting RONI gives slightly better performance.

Xie et al. presented Zeno [145] (for synchronous environments) and Zeno++ [146] (for asynchronous environments), both of which use a centralized oracle that estimates based on a validation dataset the true gradient and only keeps the k gradients most similar to this estimation. The performance of both GARs is quite good according to [121], but they depend on a centralized parameter server and need access to a sufficiently large unbiased validation dataset.

[147] takes a very different approach and proposes a GAR called PDGAN that uses a Generate Adversarial Network (GAN) to reconstruct the training data used by the peers to train the network. Based on this data, the accuracy of the received models can be estimated reliably. However, since the training data used by the peers is supposed to stay private, it is actually quite disturbing that GANs are able to reconstruct this data [14, 148], and are, in that regard, also a "highly impactful and prioritized" [149] attack in their own right.

Another recently presented GAR that is an important inspiration for this thesis is Mozi [115]. It first applies a distance-based strategy to quickly select a candidate pool of probably benign nodes, and then screens the resulting nodes based on their performance on a test dataset (performance screening).

Pruning

Since backdoor attacks (see Section Data poisoning vs model poisoning attacks) are extremely challenging to detect, let alone defend against, an entirely different class of defense mechanisms called "pruning" defenses has been proposed that specifically aims to prevent these backdoor attacks [150-152]. Pruning defenses use a representative subset of the global dataset (partially violating the FL assumption [112]) to evaluate which neurons in the neural network are inactive. These neurons are important to find and subsequently remove, because they enable attackers to create a backdoor in the model [7].

Unfortunately, even when these inactive neurons are removed from the model, more adaptive poisoning attacks are still possible [153]. After all, the boundary between a neuron being unused or being actively used is vague.

There are several other methods aimed at detecting backdoor [142, 150, 151, 154-159], but these methods either assume that there is a central server that can access the whole training dataset and scan the samples for malicious samples (which clearly is not possible in a federated learning setting) or access to a holdout set of similarly distributed data (which cannot help defend against more sophisticated model poisoning attacks as discussed in Section Byzantine attacks).

Behavioral-based

An original way to defend against targeted poisoning by sybil clones is presented in [108] and named FoolsGold. The authors first observe that, when sybils collude to poison a model, their "behavior is more similar to each other than the similarity observed amongst the honest clients". Based on this insight, they present FoolsGold which detects and rejects poisoned contributions. However [143] shows that FoolsGold is unable to recognize against a powerful single-client attack and can be evaded by decomposing a distributed attack into several orthogonal vectors.

Whereas all GARs discussed until now make it as difficult as possible for an attacker to manipulate the system, there is also a wide variety of GARs that take a different approach and aim to eliminate any incentive for a node to attack the system. A trivial approach where a parameter server simply assigns a reputation based on a performance-based screening procedure per node (such as [160]) doesn't work well, because a malicious attacker can first build up an excellent reputation, and then suddenly completely ruin the model, empowered to do so thanks to its good reputation. A better approach turns out to be rewarding and punishing of participants based on their contributions, something that can be facilitated in decentralized environments by means of a distributed ledger [46, 161-175], usually a blockchain. This ledger can also be used to save global model parameters to enhance the security of the system [166, 171].

Kang et al. introduced in [176] reputation as a means to determine the reliability of every node and subsequently proposed a GAR based on these reliability scores[177], using RONI to calculate reputations in i.i.d. environments and FoolsGold to calculate reputations in non-i.i.d. environments. For this to work, the authors (implicitly) assume an environment where nodes have a strong identity and where there are many different parameter servers learning different tasks that share reputation opinions of nodes over a public blockchain.

[178] also assign a reputation to nodes that contribute well, but their algorithm is seriously flawed: the authors use KRUM to determine if an update is benign (which is highly unreliable [121]) and then increase/decrease a node's reputation when the update is accepted/rejected respectively, implying that you can make theoretically for every good contribution also a bad contribution (in practice a single bad contribution can damage the model significantly while the impact of a single good contribution is generally very limited).

Whereas all former approaches assume that the individual workers might be Byzantine, [179] assumes a centralized setting where the parameter server might be Byzantine. They use a blockchain to audit all model updates from all peers so that everyone can verify if the parameter server aggregates the model updates correctly. The authors also train an autoencoder to recognize outliers (i.e. Byzantine attack). This seems to work quite well, but the autoencoder is only effective after it has been trained properly which may take many iterations.

Another blockchain-based approach is described by [161] called HoldOut SGD which first splits the nodes into a set of workers that use their data to train the model for a single iteration, and a voting committee that votes for the best proposals and stores this information on the blockchain (similarly to [164, 180]. The voting committee is usually selected based on Proof-of-Stake and Verifiable Random Functions (VRFs)). The method is fully decentralized but defends only against up to a factor of 1/3rd Byzantine workers. The technique is hardly scalable to a high number of nodes, because every node in the voting committee has to evaluate every single update, and because either all voting committee nodes are waiting until the workers are finished or vice versa a significant amount of time is spent idling for every node.

Where the blockchain papers mentioned above are of good quality, one has to be very careful when search for literature about this subject. There are many papers where a blockchain is used without properly understanding its (dis)advantages. For example, in [46] the authors say that they want to address privacy issues by using a blockchain, but simply using a blockchain doesn't magically improve the user's privacy. The authors also states that Directed-Acyclic-Graphs (DAGs) are a certain kind of blockchain (which is incorrect, it are different technologies. Stating that DAGs and blockchains are both examples of Distributed Ledger Technology (DLT) would have been correct) and that DAGs use cumulative PoW, which is also incorrect: newly added data DAGs usually just reference and validate previous transactions without any PoW involved.

A particularly good paper where the authors really make use of the strengths of DLT is [162] where they use a Tangle to represent the approved transactions as nodes in a Directed Acyclic Graph (DAG). Every new transaction first verifies two previous transactions by using one of the defense mechanisms described in this section and includes the updated model parameters.

There is also a considerable body of literature that uses behavioral techniques to incentive nodes with high quality training data to participate in the training process such as [168, 170, 173-177, 181-193] and Stackelberg game methods [181, 182, 194, 195], but since these methods are not supposed to defend against Byzantine attacks, we leave them out of the scope of this thesis. Additionally, the underlying assumption, namely that agents should get some kind of incentive to participate in a federated training process, does not seem to apply in many popular applications, such as Gboard, Captcha, or Google Fit.

Other

There are several papers discussing innovative defense mechanisms that are not easy to classify into a particular category. There are a few papers that use Trusted Execution Environments (TEEs) to achieve some form of security such as [168, 196-199]. [196] uses secure aggregation based on the Secure Multiparty Computation (SMC) algorithm to aggregate the values of untrusted nodes without revealing these values, enabling a parameter server that every party can trust. [197] also discusses how TEEs can be used as a defense technique, later used by [198, 199] to create a generic framework that can be used to integrate TEEs in a federated learning environment. Weng et al. [168] developed DeepChain that on one hand uses a blockchain to incentivize parties to participate in the training process, and on the other hand uses a combination of Intel Software Guard Extensions (SGX) enclaves and homomorphic cryptographic functions to provide a safe and privacy-preserving environment. Their solution work well, but is computationally also very expensive, limiting its use cases.

There are also a few solutions that model defending against Byzantine attacks as a learning problem. [200] uses a Recurrent Neural Network (RNN) and an auxiliary dataset to aggregate gradients in a Byzantine-resilient manner. The idea is that a machine learning approach can detect attacks that is hard for other algorithms to accomplish. Unfortunately, since their RNN is a "black box", the authors are unable to give any theoretical guarantees. [201] uses variational autoencoders with spectral anomaly detection to detect malicious updates based on their low-dimensional embeddings. By removing the noisy and irrelevant features, the anomalous (malicious) model updates can be distinguished from the benign updates in a low-dimensional latent feature space.

A final type of defense mechanisms we would like to highlight are defenses based on replicating the same training over multiple nodes [124, 133, 202, 203]. When all nodes are benign, they will obviously report the exact same results. While the accuracy of these mechanisms is often illustrated with rigorous theoretical guarantees, they generally assume a centralized server with either a copy of the data or the ability to globally shuffle the data, which makes the algorithm inappropriate for a decentralized federated environment. For example, DRACO [124] lets the parameter server send to multiple workers the same chunk of data and uses majority voting to find the correct evaluation. Against a small number of Byzantine attackers DRACO is very robust, but the algorithm scales poorly to a greater number of attackers. For example, when there are just 5 attackers, each chunk already needs to be calculate $5\ x\ 2 + 1 = 11$ times.

2.4. Non-i.i.d.

Whereas in regular distributed learning environments, a characteristic of a typical federated learning environment is that the shards of the slaves are non-i.i.d. (not independent and identically distributed) [68, 69, 99]. For example, it is possible that device A has class X and Y, and device B has class Y and Z. As a result, the model of device B will be quite different from the model of device A, making it hard for device A to determine if device B's model is malicious or not. To make matters worse, a trivial average of the parameter updates yields a considerably worse model than a model that would have been trained by a single node on class X, Y, and Z [30, 204, 205].

The challenge of building a single global model by combining multiple local models without reducing their accuracy is closely related to *multi-task learning[206]*. Multi-task Learning or Continual Learning (CL) is concerned with preventing *Catastrophic Forgetting* or *Catastrophic Inference*, a phenomenon where the neural network completely forgets what it has learnt before when it is taught a new task. Instead, the network should be capable of *Lifelong learning*: continuously acquire new knowledge, refine existing knowledge, and prevent new tasks from interfering with existing knowledge.

Figure 1 is a Venn diagram created by Lesort and Lomonaco [24] categorizing the existing CL methods into 4 partially overlapping categories:

Architectural approaches seek to allocate additional neural nodes whenever they are required or freeze specific weights[207-210], but this requires the developer to know the number of tasks / samples per task a-priori and leads to scalability issues for large neural networks.

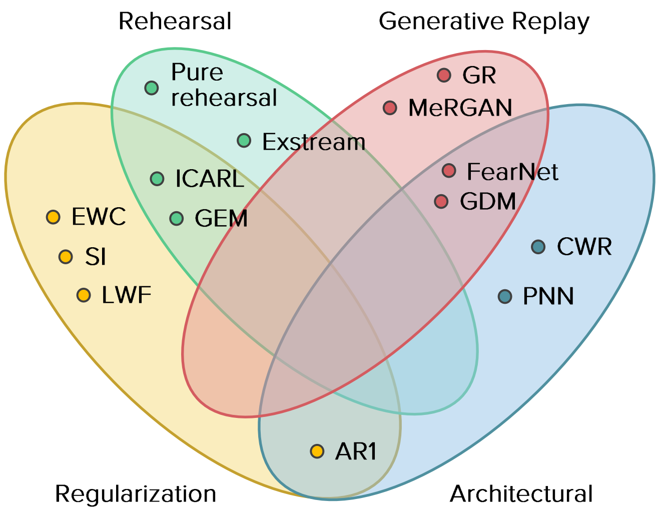


FIGURE 1. VENN DIAGRAM OF EXISTING CL METHODS [24]

Regularization techniques minimize the extent to which the most important weights are overwritten by the training on a new model. Elastic Weight Consolidation (EWC) [210], which was based on Learning without Forgetting (LwF) [211] is an influential regularization technique that extends the loss function with a quadratic penalty on the change in parameters that are important for the formerly learned tasks. The authors set the importance of the parameters to the diagonal of the Fisher information matrix, which works well for learning permutations of the same task, but not for learning entirely new categories incrementally [212]. Several improvements have been made since such as [213-216].

Rehearsing old samples interleaved with new samples is also an effective way to prevent catastrophic forgetting. [205] concluded that globally sharing just 5% of the training data can result in a 30% increase in accuracy. These training samples can be selected randomly or carefully to be as representative of the coreset as possible. However, this approach increases the amount of memory needed to store all samples[217-220].

Generative replay is a variant on rehearsing old samples where a *Generative Adversarial Network* (GAN) is used to artificially generate samples that have a similar distribution as the past experiences. These samples are then intertwined with the new empirical training samples just like in rehearsal-based strategies.

The approaches discussed until now are generic multi-task learning techniques, but there has also been research into similar techniques specifically for federated learning environments.

[29] is an example of a non-i.i.d. approach for federated learning, but the authors use clusters which do not work well on high-dimensional data (such as neural networks): the authors simply throw away all parameters of the neural network except for the first 288 parameters in the first layer. The technique presented by [205] is more effective and uses rehearsal: they assume that a small amount of i.i.d. data is available that can be shared across all peer nodes (which is generally a realistic assumption).

In specific situations where the loss function is convex and its conjugate dual is expressible, research has shown that dual coordinate ascent approaches such as Mocha en Cocoa can yield superior results [205, 221-223]. Mocha [223] for instance handles non-i.i.d. datasets while also tackling the challenge of fault tolerance, stragglers, and communication efficiency. The algorithm models the relation between the tasks by adding a loss term and subsequently uses a primal-dual formulation to solve the optimization problem. However, like many other multi-task learning algorithms, it assumes that all peers participate in every training round which makes these algorithms harder to apply in a truly federated setting.

A particularly popular approach seems to be to use Elastic Weight Consolidation ([224-228] which, as explained in this section before, penalizes large changes of parameters important for previously learned tasks. It is somewhat surprising that more recent methods such as CWR(+), LWF, or AR1 have not been investigated yet because these methods perform significantly better in non-federated environments than EWC[229].

3. Proposed solution: Pro-Bristle

In this section, we will first delineate the components of Pro-Bristle, the main contribution of this thesis, and explain how these solutions relate to each other (see Figure 3). Subsequently, we will zoom in on the non-i.i.d. components of Pro-Bristle since these constitute a particularly complex part of the algorithm. We will also devote attention to the implementation of the private-set intersection cardinality. The section is concluded with the pseudocode of Pro-Bristle, formally describing the same algorithm as visualized in Figure 3.

The main contribution of this work is the development of a new GAR named Pro-Bristle (Practical yet RObust Byzantine-Resilient decentralIzed StochasTic federated LEarning). To explain how Pro-Bristle works we will first recapitulate the four characteristics and four additional design principles that we listed in Section Introduction).

A federated learning environment is characterized by several challenges:

- The network is massively distributed
- The nodes are unreliable
- The data is distributed non-i.i.d.

Additional design principles that we will follow for the design of Pro-Bristle are:

- The network should be decentralized
- The algorithm should be Byzantine-resilient
- The nodes should be able to work asynchronously
- The algorithm should be communication-efficient

These challenges/design principles are overcome/integrated in Pro-Bristle by using a mix of interrelated technologies. These challenges and corresponding solutions are mapped in Figure 2:

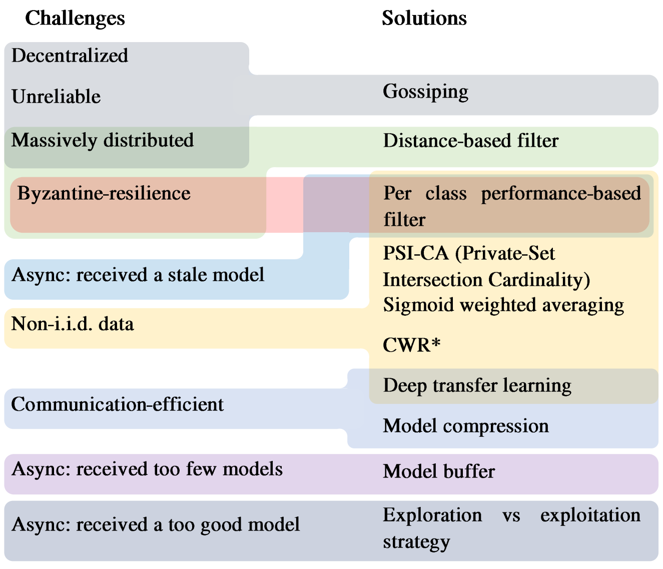

Challenges
Solutions
Decentralized
Unreliable
Gossiping
Massively distributed
Distance-based filter
Byzantine-resilience
Per class performance-based filter
Async: received a stale model
PSI-CA (Private-Set Intersection Cardinality)
Sigmoid weighted averaging
Non-i.i.d. data
CWR*
Deep transfer learning
Communication-efficient
Model compression
Async: received too few models
Model buffer
Async: received a too good model
Exploration vs exploitation strategy

FIGURE 2. MAPPING OF CHALLENGES ADDRESSED BY EACH SOLUTION

3.1. Components

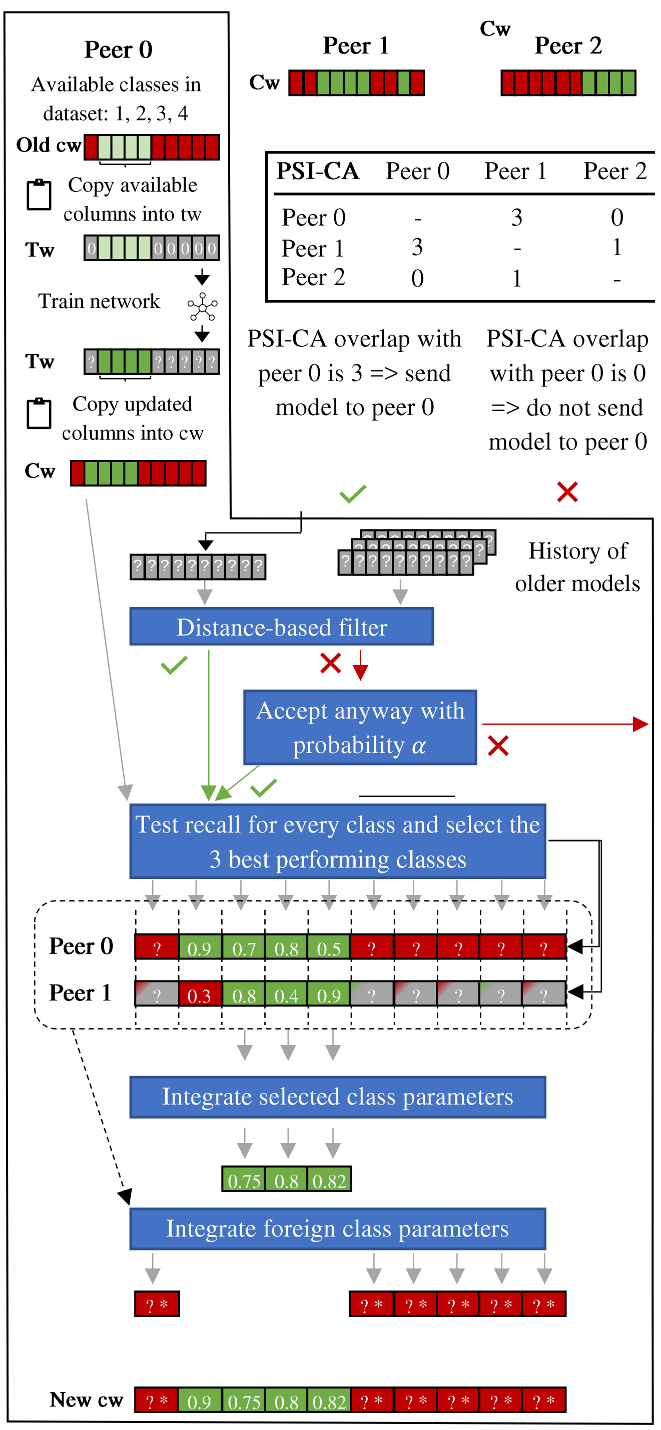
In contrast to practically all existing works on federated learning until now we will use **gossiping** to make a massively distributed number of nodes learn together in a decentralized and scalable way. With gossiping we mean that every node sends at every iteration to a few other nodes an update {Hegedűs, 2019 #268}{Hu, 2019 #267}. These nodes are typically different from the nodes from which the node received an update. Gossiping also makes the fact that nodes are unreliable less relevant, since gossiping happens with "a random node"; if some node happens to be offline, the other nodes will just choose other nodes to gossip with. Note that due to the nature of gossiping, every node has a (slightly) different model.

To provide Byzantine-resilience, we first observe that in a perfect world (non-i.i.d., all peers working synchronously on the same iteration, and with a balanced dataset) for every node A, all benign models that node A receives will be reasonably close to node A's own model. However, Byzantine models that the node receives can be either within or outside this distance. This inspires us to first apply a **distance-based filter** to get rid of some Byzantine attacks, and then apply a **per class performance-based filter** to filter out more sophisticated Byzantine attacks. Note that the performance-based filter depends on the availability of private trusted data samples. This means that Pro-Bristle completely ignores models from peers until enough private data samples are available to test other models with the desired level of confidence.

In contrast to the assumption until now, the world is not perfect in real-life (as illustrated in Section Why federated learning?). For example, in a gossiping decentralized environment, it is natural for the data to be distributed non-i.i.d. To be effective in detecting Byzantine attacks in a non-i.i.d. environment, we combine a set of technologies (namely **SRA PSI-CA**, **sigmoid weighted averaging**, **CWR***, and **deep transfer learning**) in a new way. This part of Pro-Bristle is complex enough to deserve its own section (see Section "Non-i.i.d."). The last component, namely **deep transfer learning**, is not only an essential component of Pro-Bristle, but is also used to significantly reduce the bandwidth needed to train the model in combination with GZipped **model compression**.

An asynchronous system can be faster than a synchronous system, especially when the computation power or bandwidth differs significantly between nodes. A faster node does not have to wait for the synchronization step and a slower node does not lose its progression after the next synchronization step after all. To account for an asynchronous environment, we will first differentiate between three sub-problems:

- *Too few peer models received to reliably perform distance-based screening.* After an iteration, the number of models received from other nodes is arbitrary. When iterations are computed fast (for example, because the node has significant computational resources) and



PSI-CA	Peer 0	Peer 1	Peer 2
Peer 0	-	3	0
Peer 1	3	-	1
Peer 2	0	1	-

Peer 0
Available classes in dataset: 1, 2, 3, 4
Old cw
Copy available columns into tw
Tw
Train network
Tw
Copy updated columns into cw
Cw
Peer 1
Cw
Cw
Peer 2
PSI-CA overlap with peer 0 is 3 => send model to peer 0
PSI-CA overlap with peer 0 is 0 => do not send model to peer 0
History of older models
Distance-based filter
Accept anyway with probability α
Test recall for every class and select the 3 best performing classes
Peer 0
Peer 1
Integrate selected class parameters
Integrate foreign class parameters
New cw

FIGURE 3. HIGH-LEVEL OVERVIEW OF PRO-BRISTLE

models are received only slowly (for example, because the available bandwidth is limited), the number of received models might be small. When, say, only two models are received, trivial distance-based screening procedures obviously do not work because there are insufficient received models to compare with each other. We propose to add a **model buffer** to keep track of a fixed number of recently received models to make the distance-based screening procedure more robust. Applying this idea in a federated setting is not entirely new because it was also explored by Yang et al. [83]. However, the paper of Yang et al. (a) assumes a centralized instead of a decentralized setting, (b) does not explicate what advantage using multiple buffers exactly gives in their solution (in fact, it is entirely unclear), and (c) is unable to update its model directly after receiving an update, resulting in subsequent local training on a (slightly) outdated model.

- *Stale model received.* A model that is received might be outdated and stale. In this case, its performance will be subpar and therefore be filtered out either by the **distance-based filter** or the **performance-based filter**.
- *Extremely accurate model received that is so different from the peer's own model that it is filtered out by the distance-based screening method.* A model that is received might be way better than the current model, causing it to be filtered out by the distance-based screening method. To solve this, we propose to use a popular strategy that has to the best of the authors' knowledge never been applied in this context before: **exploration vs exploitation**. Based on an exploration ratio α, the distance-based screening filter should randomly accept models that are "not close enough" to be considered otherwise. The performance-screening filter will then notice the supposedly superior performance of this model. We also propose that to use weighted averaging to aggregate the models based on their performance: when a model is received that performs extremely well, the node should shift its current model very significantly towards this model.

Non-i.i.d. components

To properly detect Byzantine attacks in a **non-i.i.d.** environment, we need to address two challenges:

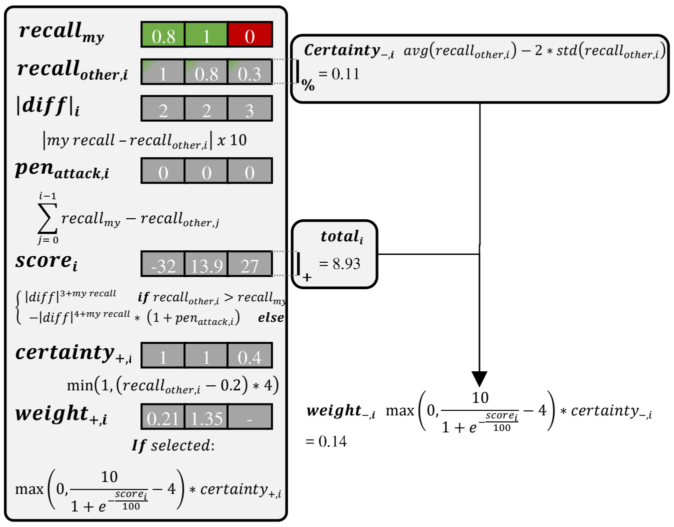
- We need to be able to prevent Byzantine attacks, which is non-trivial in a non-i.i.d. environment because the model of a peer with completely different data distribution will likely be recognized as aberrant and therefore malicious by both the distance-based and performance-based screening procedure.
- Even when we are able to filter out Byzantine attacks, we still need to be able to merge non-Byzantine models trained on a different data distribution and prevent

catastrophic forgetting. Catastrophic forgetting means that when a neural network is trained for a certain set of classes and thereafter is trained for another set of classes, it completely overwrites (forgets) the first set of classes. In a federated environment this results in nodes constantly overwriting each other, resulting in mediocre performance.

These challenges are addressed by Pro-BRISTLE as illustrated in Figure 3. First, we take a step back and determine if a node has enough overlap in its data distribution with another node to properly check the accuracy of the received model. Unfortunately, in a federated setting it is not possible to simply compare the data of a pair of nodes, because this data is private. Therefore, we use **Private Set Intersection Cardinality (PSI-CA)**, explained in more detail in …) to check the overlap between the datasets. Only when nodes have sufficient overlap, they will gossip with each other. In the tests we will assume that the network is strongly connected for evaluation purposes (as mentioned in Section 4.8), but in practice Pro-Bristle also works fine when the network is not connected. For example, in a hypothetical situation where half of the nodes (i.e. set A) only has classes 1, 2, and 3, and where the other half of the nodes (i.e. set B) only has classes 4, 5, and 6, the nodes in sets A and B will simply learn a different model. As soon as a few peers of A or B one of the obtain a sufficiently high number of samples of the other set, the model will slowly converge towards a single model shared by all nodes in both A and B.

Moreover, every node uses **CWR*** for its non-i.i.d. learning which differentiates between short-term memory tw and long-term memory cw. Unfortunately, CWR* requires all nodes to agree on a frozen set of non-output layers which is impossible in a typical federated learning scenario, because the whole purpose of federated learning is to actually learn all layers. However, when we make an additional assumption, we can solve this problem: we assume that for the dataset that we want to learn, there exists another publicly available dataset with approximately the same *features* (for example, English words in the case of a language-based dataset, or pixel patterns in the case of an image-based dataset). In many cases, this assumption is realistic considering the rapidly increasing popularity of transfer learning. Given that this assumption applies to the dataset being learned, we can used transfer learning to learn the non-output layers offline and then transmit these layers to all nodes where they are frozen to be used by CWR*.

Whereas Mozi simply calculates the accuracy of every model after applying the distance-based filter, we choose to measure the accuracy of every model for every class of which our node has a sufficient number of samples. Thanks to our PSI-CA result, we know the overlap between the datasets of both peers and select the best-performing subset of classes with the same size as this overlap. From now on, we will differentiate between this set of selected classes and the other "foreign" classes. Figure 4 shows on the left side an example of how three selected classes are



$recall_{my}$
0.8
1
0
$recall_{other,i}$
1
0.8
0.3
$|diff|_i$
2
2
3
$|my\ recall - recall_{other,i}|\ x\ 10$
$pen_{attack,i}$
0
0
0
$\sum_{j=0}^{i-1} recall_{my} - recall_{other,j}$
$score_i$
-32
13.9
27
$\begin{cases} |diff|^{3+my\ recall} & if\ recall_{other,i} > recall_{my} \\ -|diff|^{4+my\ recall} * (1 + pen_{attack,i}) & else \end{cases}$
$certainty_{+,i}$
1
1
0.4
$\min(1, (recall_{other,i} - 0.2) * 4)$
$weight_{+,i}$
0.21
1.35
-
$If\ selected:$
$\max\left(0, \frac{10}{1+e^{-\frac{score_i}{100}}} - 4\right) * certainty_{+,i}$
$Certainty_{-,i}$ $avg(recall_{other,i}) - 2 * std(recall_{other,i})$
$|_{\%}$ = 0.11
$total_i$
$|_{+}$ = 8.93
$weight_{-,i}$ $\max\left(0, \frac{10}{1+e^{-\frac{score_i}{100}}} - 4\right) * certainty_{-,i}$
= 0.14

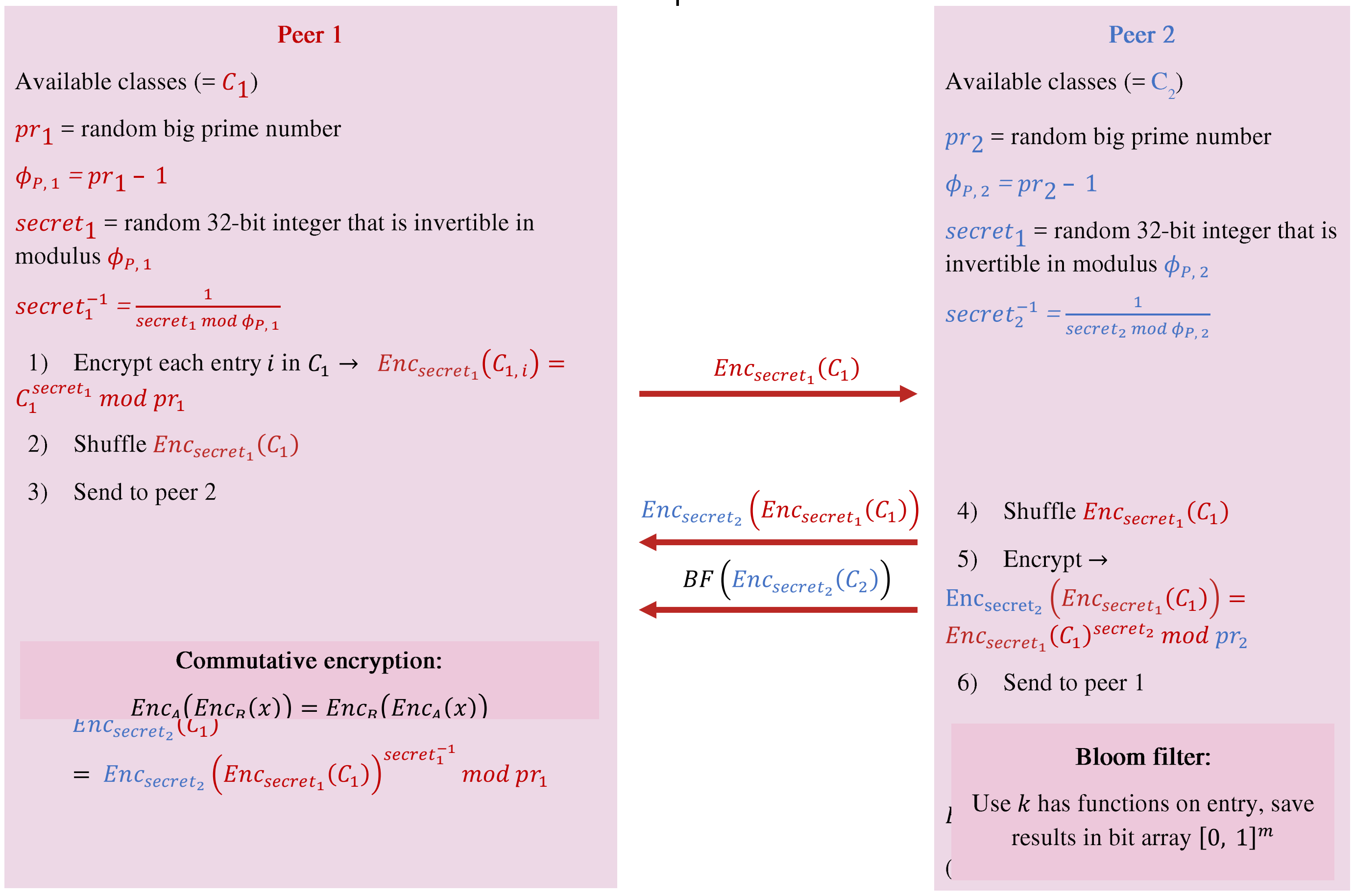
FIGURE 4. INTEGRATION OF PARAMETERS

integrated into the peer's own model. To prevent a sybil attack where a large group of sybils slowly degrade the model's performance, an attack penalty $pen_{attack,i}$ is applied that penalizes subsequent performance degradations per class. The right side shows how the weight is calculated that is applied to the foreign classes, i.e. the classes for which our peer cannot (reliably) check their performance.

Finally, simple weighted averaging is used to calculate the new cw parameters that are used in the next iteration.

Private Set Intersection Cardinality (PSI-CA)

Pro-Bristle uses estimates of the overlap between the classes of the peers to potentially greatly reduce the number of peers communicating with each other in non-i.i.d. situations and also significantly improve the performance of the performance-screening phase. The Private Set Intersection Cardinality (PSI-CA) is implemented similarly as [245] and visualized in Figure 5. However, whereas the authors of [245] use Pohlig-Hellman exponential cipher [249] as commutative encryption method which is computationally expensive, we will use the more efficient SRA as proposed by Shamir, Rivest and Adleman {Shamir, 1981 #250} and later rediscovered (or not properly referenced) by Cristofaro et al. {De Cristofaro, 2012 #252}. Despite the fact that one peer cannot directly determine the exact classes that another peer owns, it is easy to determine indirectly by simply submitting an exhaustive set of PSI-CA requests [247]. To prevent this from happening, we require peers to submit at least 4 classes (the higher the number, the more confidentiality; can be padded with randomly generated noise when the peer does not have enough classes) and enforce a limit on the number of PSI-CA requests accepted per peer (for example, 1 per month) and also on the number of PSI-CA request accepted in total (for example, 1 per day; necessary because an attacker may create multiple distinct nodes that collaborate to determine the peer's classes). Given enough nodes and time, an attacker can unfortunately always determine the classes owned by some particular peer due to the nature of the class-overlap problem, but these measures make it significantly more difficult to do so for a large number of nodes.



Peer 1
Available classes (= C_1)
pr_1 = random big prime number
$\phi_{P,1} = pr_1 - 1$
$secret_1$ = random 32-bit integer that is invertible in modulus $\phi_{P,1}$
$secret_1^{-1} = \frac{1}{secret_1 \bmod \phi_{P,1}}$
1) Encrypt each entry i in $C_1 \rightarrow Enc_{secret_1}(C_{1,i}) = C_1^{secret_1} \bmod pr_1$
2) Shuffle $Enc_{secret_1}(C_1)$
3) Send to peer 2
Commutative encryption:
$Enc_A(Enc_B(x)) = Enc_B(Enc_A(x))$
$Enc_{secret_2}(C_1)$
$= Enc_{secret_2}(Enc_{secret_1}(C_1))^{secret_1^{-1}} \bmod pr_1$
$Enc_{secret_1}(C_1)$
$Enc_{secret_2}(Enc_{secret_1}(C_1))$
$BF(Enc_{secret_2}(C_2))$
Peer 2
Available classes (= C_2)
pr_2 = random big prime number
$\phi_{P,2} = pr_2 - 1$
$secret_1$ = random 32-bit integer that is invertible in modulus $\phi_{P,2}$
$secret_2^{-1} = \frac{1}{secret_2 \bmod \phi_{P,2}}$
4) Shuffle $Enc_{secret_1}(C_1)$
5) Encrypt → $Enc_{secret_2}(Enc_{secret_1}(C_1)) = Enc_{secret_1}(C_1)^{secret_2} \bmod pr_2$
6) Send to peer 1
Bloom filter:
Use k has functions on entry, save results in bit array $[0,1]^m$

FIGURE 5. VISUALIZATION OF SRA PRIVATE SET INTERSECTION CARDINALITY

3.2. Pseudocode

Notations

We use bold lower-case letters such as $\boldsymbol{x}$ to represent vectors, lower-case letters such as γ to represent scalars and functions, and upper-case curlicue letters such as $\mathcal{N}$ to represent sets. Aggregated vectors are denoted by a line over vectors such as m. All operations between vectors are element-wise operations in this paper (except inner products of vectors).

```
Input: initial estimate $x_0$, dataset $D^{train}$ containing an arbitrary non-validation subset of the node's collected data, dataset $D^{test}$ containing a small, trusted set of samples for each class, history buffer size $\gamma$, exploration ratio $\frac{\beta}{\alpha}$, transfer network $\Psi$, weight diff constant $\eta$

$\mathcal{N}^s \leftarrow$ getSimilarPeersWithClassOverlap()
$l \leftarrow$ initialize loss function by deep transfer from $\Psi$
For $t = 0, 1, 2, \ldots$ do
    Stochastically sample $\xi_i(t)$ from $D_i^{train}$
    $\nabla l(\boldsymbol{x}_i(t), \xi_i(t)) \leftarrow$ Compute the local gradient
    $\mathcal{N}_i^n(t) \leftarrow$ All models received from peers $j \in \mathcal{N}^s$
    $\mathcal{N}_i^r(t) \leftarrow$ The $\gamma$ most recent models received in previous iterations
    If $|\mathcal{N}_i^n(t)| > 0$ then
        For $j\ in\ (\mathcal{N}_i^n(t) \cup \mathcal{N}_i^r(t))$ do
            $d_{i,j} \leftarrow \left\|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\right\|$
        End for
        $\mathcal{N}_i^d(t) \leftarrow \left(argmin_{\substack{\mathcal{N}^* \subseteq \mathcal{N}_i \\ |\mathcal{N}^*| = \alpha}} \sum_{j \in \mathcal{N}^*} d_{i,j}\right) \setminus \mathcal{N}_i^r(t)$
        $\mathcal{N}_i^e(t) \leftarrow \mathcal{N}^*{}_{\mathcal{N}^*\ is\ a\ random\ subset\ where\ |\mathcal{N}^*| = \beta} \overset{\subseteq}{} \left(\mathcal{N}_i^n(t) \setminus \mathcal{N}_i^d(t)\right)$
        $\mathcal{N}_i^c(t) \leftarrow \mathcal{N}_i^d(t) \cup \mathcal{N}_i^e(t)$
        For $j \in i \cup \mathcal{N}_i^c(t)$ do
            For $c\ in\ classes(D_i)$
                $recall_{j,c}(t) \leftarrow recall(l, \boldsymbol{x}_j(t), D_i^{test})$
            End for
            If $j \neq i$ then
                $C_j(t) \leftarrow argmin_{|C| = cardinality(\mathcal{N}_j^s(t))}^{c\ in\ classes(D_i)} recall_{j,c}(t)$
                $certainty_j(t) \leftarrow \max\left(0,\ average\left(recall_{j,c \in C_j(t)}(t)\right) - 2 * std\left(recall_{j,c \in C_j(t)}(t)\right)\right)$
                For $c\ in\ C_j(t)$
                    $weightdiff_{j,c}(t) \leftarrow |recall_{j,c}(t) - recall_{i,c}(t)|\ x\ \eta$
                    $seqAttackPenalty_{j,c}(t) \leftarrow getSeqAttackPenalty(C_j(t), c, recall)$
                    If $recall_{j,c}(t) > recall_{i,c}(t)$
                        $score_{j,c}(t) \leftarrow weightdiff_{j,c}(t)^{3 + recall_{i,c}}$
                    Else
                        $score_{j,c}(t) \leftarrow -weightdiff_{j,c}(t)^{4 + recall_{i,c}} * (1 + seqAttackPenalty_{j,c}(t))$
                    End if
                    $weights_{j,c}(t) \leftarrow \max\left(0, \frac{10}{1 + e^{-\frac{score_{j,c}(t)}{100}}} - 4\right) * certainty_j(t)$
                End for
                $peerWeight_j(t) \leftarrow \max\left(0, \frac{10}{1 + e^{-\frac{\sum_{c \in C_j(t)} score_{j,c}(t)}{100}}} - 4\right) * certainty_j(t)$
            End if
        End for
        $\boldsymbol{x}_i(t) \leftarrow weightedAvg(\boldsymbol{x}(t), weights_{j,c}(t), C)$
        $\boldsymbol{x}_i(t) \leftarrow simpleAvg(\boldsymbol{x}(t), weights_{j,c}(t), classes(D_i) - C)$
    End if
    $\boldsymbol{x}_i(t+1) \leftarrow \boldsymbol{x}_i(t) - \lambda \nabla l(\boldsymbol{x}_i(t), \xi_i(t))$
End for
```

4. Implementation

In this section, we aim to give the reader a good understanding of how Pro-Bristle was implemented and the experiments were conducted. At the end of the section, we also describe the biggest issues that we encountered during the development of the system.

4.1. Datasets

From all papers that we read as part of the literature review for this thesis, two datasets turned out to be extremely popular in the literature, namely the **MNIST** dataset [1, 3, 7, 12, 13, 15, 16, 29, 81, 89-91, 99, 100, 108, 110, 111, 113, 114, 121, 124, 130, 132, 135-140, 143, 144, 147, 148, 151, 153, 154, 156, 158, 160, 161, 164, 165, 167, 168, 178, 198-201, 204, 205, 230] and the **CIFAR-10** dataset [5, 10, 13, 16, 79, 81-84, 86, 110, 113, 114, 124, 130, 131, 133, 140, 142, 143, 145, 146, 152, 158, 161, 198, 200, 202, 204, 205, 225].

The MNIST dataset consists of 60,000 gray-scale training images and 10,000 test images of 28x28 px that represent handwritten digits. Even though MNIST does not represent a typical federated learning dataset, it is very popular and, thanks to its status as one of the most popular machine learning datasets, makes it possible to compare our results with a large body of established literature. To transfer-learn MNIST, we first train the network on the EMNIST-Letters dataset which is similar to MNIST, but contains the 26 letters of our alphabet rather than digits.

The CIFAR-10 dataset also consists of 60,000 training images and 10,000 test images. These images are 32x32 px and RGB-colored, showing pictures of ten distinct types of objects such as cars, airplanes, and dogs. CIFAR-10 turns out to be significantly more challenging to learn than MNIST, which might be useful to properly investigate the power of new algorithms. To transfer-learn CIFAR-10, we first train the network on CIFAR-100. CIFAR-100 is a similar dataset as CIFAR-10, but contains 100 classes rather than 10 classes. Similarly to [231] we reduce the conceptual overlap between CIFAR-10 and CIFAR-100, by excluding super-classes of CIFAR-100 that are conceptually similar to CIFAR-10 classes: vehicle 1, vehicle 2, small mammals, medium-sized mammals, and large carnivores.

We also include a realistic federated learning dataset, namely the **WISDM** dataset, one of the most popular HAR (Human Activity Recognition) datasets [232]. This dataset consists of 1,098,207 recordings of people performing one of the six included activities. Every recording consists of at least 544 measurements of the acceleration sensor. To transfer-learn this dataset we pretrain the network similarly to [233] on the MobiAct dataset where we again exclude overlapping classes with the WISDM dataset.

4.2. Machine Learning part

For MNIST and CIFAR-10, we use the same CNN architectures as used by [30] with the only difference that we use Leaky ReLu instead of the regular ReLu as activation function for the hidden layers, since the former one suffers less from the vanishing gradients problem. For the output function we use the softmax function and as loss function, we use negative loglikelihood.

MNIST

Layer	Details
Convolution	Kernel: <5, 5>, stride: <1, 1>
Max pooling	Kernel: <2, 2>, stride: <2, 2>
Convolution	Kernel: <5, 5>, stride: <1, 1>
Max pooling	Kernel: <2, 2>, stride: <2, 2>
Dense	#nodes: 500
Output	#nodes: 10

CIFAR

Layer	Details
Convolution	Kernel: <3, 3>, stride: <1, 1>
Batch normalization	
Max pooling	Kernel: <2, 2>, stride: <2, 2>
Convolution	Kernel: <2, 2>, stride: <1, 1>
Batch normalization	
Max pooling	Kernel: <3, 3>, stride: <1, 1>
Convolution	Kernel: <2, 2>, stride: <1, 1>
Batch normalization	
Max pooling	Kernel: <2, 2>, stride: <2, 2>
Convolution	Kernel: <2, 2>, stride: <1, 1>
Batch normalization	
Max pooling	Kernel: <2, 2>, stride: <2, 2>
Output	#nodes: 10, dropout: 0.8

WISDM

Layer	Details
1D convolution	Kernel: <3>, #nodes: 64
1D max pooling	Kernel: <2>, stride: <2>
1D convolution	Kernel: <3>, #nodes 64
Global max pooling	
Output	#nodes: 10

4.3. Gradient Aggregation Rules

To properly compare our proposed solution with existing methods, we implemented five other gradient aggregation rules (described in detail in Section **Error! Reference source not found.**):

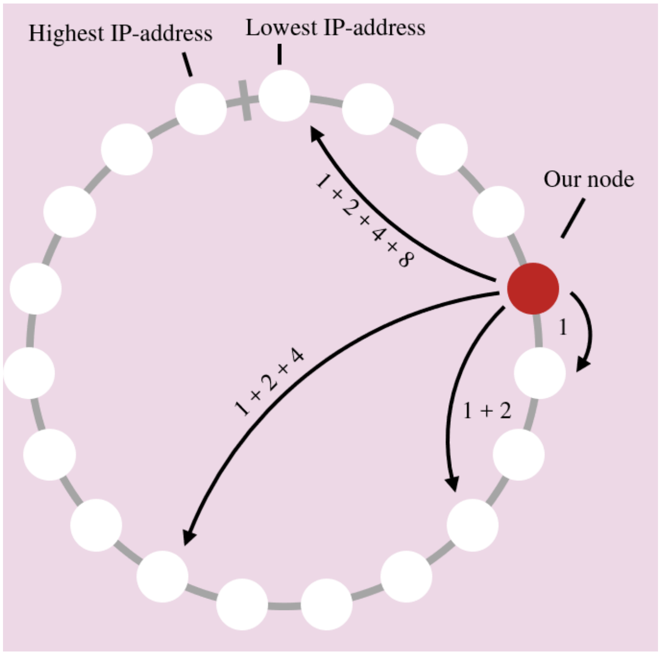
- **FedAvg** [30]. FedAvg is equivalent to simple averaging, is researched extensively, and very often used as baseline to compare other GARs against.
- **CM (Coordinate-wise Median)** [99]. CM is perhaps the simplest, but also a very effective Byzantine-resilient defense mechanism, as shown by [121].
- **Krum** [68]. Krum is an extremely popular GAR that selects the model with the minimal local sum of Euclidean distances.
- **Bridge** [137]. A very recent survey paper [121], published in May 2020, concluded that Bridge was the best performing GAR in decentralized settings.
- **MOZI** [115]. MOZI was published shortly after [121]'s survey and uses a hybrid between distance-based and performance-screening to achieve superior results.

4.4. Environment

We use two separate ways to test the performance of Pro-Bristle, namely in a local simulation and in a truly decentralized environment. In the former case, we run a single program that iteratively trains and combines up to 250 models to simulate a small-scale federated setting. This approach is not only relatively fast, but also makes it easy to accurately control a variety of settings, such as little computation power, low bandwidth, nodes that randomly join/exit, etc. We also emulate 16 completely independent smartphones to test if the results are comparable in a "real" setting. Unfortunately, this limit of 16 emulators is hardcoded in the Android emulator executable which makes it unpractical to run more emulators simultaneously. However, 16 emulators are enough to accurately measure the performance of different GARs and, if the programs works well on 16 emulators, gives us confidence that the code works as intended and scales to a higher number of nodes given the gossiping nature of the system and the cheap distance-based filter.

4.5. Network connectivity

We implemented 4 TODO



4.6. Network protocol

The nodes that use federated learning to collaboratively learn a model communicate with each other over a network. Since we aim to run everything completely decentralized, it is non-trivial for nodes to find and communicate with each other in a fault-tolerant and effective way. Therefore, we use IPv8[234, 235], a well-established decentralized peer-to-peer (P2P) middleware stack that is used by i.a. the popular Tribler media sharing system[236, 237]. Furthermore, we extended IPv8 with two significant performance improvements to make the system more effective.

The first improvement is an extension to the Trivial File Transfer Protocol (TFTP) that enables parallel transmission of multiple files between the same 2 nodes. This was implemented by assigning a unique file identifier to each file and prepending every data packet with this identifier to keep track of all packets.

The second improvement is, to speed up the slow transmission times of TFTP, the first Kotlin implementation of the micro Transport Protocol (µTP). This protocol aims to mitigate the poor latency and congestion control problems found in regular TCP implementations, while providing reliable and ordered packet delivery. It sends multiple packets simultaneously and automatically slows down the transmission when the network seems to get congested.

4.7. Task automation

Creating and starting all emulators, and installing, starting, initializing, running, and evaluating the federated learning program on every emulator, is infeasible to do by hand for a large number of emulators. Therefore, we created a separate coordinator that automates these tasks. Based on the current operating system it executes several scripts (for example to create new emulators that are reset to factory settings, or to increase the local network buffers to decrease the uncontrolled/unintended packet loss to speed up the network communication) to create and run all tests consecutively. The nodes are instructed to perform specific tasks as stated in a dedicated JSON file and subsequently communicate their evaluations to the coordinator, who writes the evaluations to a CSV file.

A separate Kotlin script was used to gather and process all evaluations, and a Python script was used to generate based on these evaluations the figures as shown in this paper.

4.8. Threat Model

To evaluate the Byzantine-resilience of the GARs discussed in this thesis, we setup several Byzantine agents that aim to degrade the model's accuracy. We assume that, in non-i.i.d. environments, the Byzantine agents do not know which classes the peer under attack has. This assumption is dependent on the PSI-CA as discussed in detail in Section Private Set Intersection Cardinality (PSI-CA) above. Furthermore, we assume that the nodes being attacked have sufficient training data to evaluate and thus consider the integration of peer models (as discussed in Section 3.1) and that the attackers do not use backdoor attacks since these are NP-hard to detect (see Section 2.2). In non-i.i.d. situations we also assume that for every node x the set of all nodes to which node x is (indirectly) connected covers all classes of the entire dataset. Note that Pro-Bristle also works fine when node x and its (indirect) neighbors only have a subset of all possible classes, or even when node x is completely isolated or fully surrounded by Byzantine nodes. The only disadvantage of such a situation is that node x only learns to train the model on the classes that are available and that, when node x would like to predict a new class, its model is not trained to do so.

2-label-flip and all-label-flip attack

As described in Section 2.2, a popular data poisoning attack is the label-flip attack where the labels of two or more classes are changed [5, 108, 109]. For this thesis, we evaluate both a label-flip attack where 2 labels are flipped, and a label-flip attack where all labels are flipped.

Noise attack

As explained in Section 2.2, simply sending random noise with a small variance is ineffective in preventing convergence because the mean of the noise equals 0. When the noise has a larger variance, it may indeed prevent convergence, but also makes the noise attack easy to detect. Therefore, we opt to center the noise around a value just slightly different from 0, namely 0.2. This attack will not be easily detected by the GARs evaluated in this paper as we will see.

Krum attack [111]

Whereas data poisoning attacks generally do not make assumptions about the GARs employed, model poisoning attacks often target a specific GAR. Fang et al. presents in [111] an effective attack against Krum by iteratively sending an attack vector that will be *just* accepted by Krum whilst inflicting maximum damage to the peer's model. As recommended by the authors, we used $b = 2$.

Trimmed Mean attack [111]

In the same paper as the aforementioned Krum attack, Fang et al. also describe a model poisoning attack against the Trimmed Mean GAR. The attack determines for each parameter of the model the gradient direction and then creates an attack vector that is exactly the opposite direction, scaled per parameter depending on the values of the other benign peers. As recommended by the authors, we used $b = 2$.

4.9. Biggest issues encountered

While working on my thesis, I encountered several major drawbacks that did cost me a significant amount of time. I want to highlight a few important ones to illustrate this:

- It was extremely hard to make separate local emulators communicate to each other. First of all, it turned out that TFTP (the only network protocol available in IPv8) was unable to send and receive multiple files to/from the same peer simultaneously. After I rewrote it, it turned out that it was way too slow to transmit the entire model via localhost between all peers for every iteration. Therefore, I had to implement an entirely new network protocol, namely UTP. It is extremely time-consuming and intense to get a network protocol to run correctly, because there are many threads doing lots of things in parallel on multiple emulators and when the connection suddenly crashes after several minutes, it requires a lot of effort to debug where it is going wrong (for example, I had to replace HashMap by ConcurrentHashMap to prevent threading issues, but this caused deadlocks so I had to use proper mutexes and coroutines). Unfortunately, modern debugging software is still uncapable of breaking the program's execution at the moment of the crash and then go a few steps back, which complicates the debugging process significantly. It took more than a week to find out why packets sometimes got lost when transmitted over localhost: the local network buffers were too small.
- Another source of problems for network communication over localhost was the CPU scheduler of Linux. Originally, a peer sent a message to another peer and then registered that it had send the message. However, when the CPU scheduler decided to pause the peer just after it had sent a message to another peer, then let the other peer respond, and then resuming the execution of the former peer, then the response of the other peer was received before the next line (registering that the peer had sent the message to the other peer), causing the program to crash in a way that was terrible to debug.

- Another issue on which I could not find anything on the internet, and which took days to solve was that the debugger cannot attach to the emulator when Android Studio and IntelliJ Idea are running simultaneously. I have reported this bug to IntelliJ.
- Another issue that I came across that also seems to be limited to only my app is that the performance profiler of Android Studio cannot stop profiling when a coroutine is being executed. It works correctly when I would change the coroutine to a thread. I have also reported this bug to IntelliJ.
- A very irritating limitation of developing for Android is that Google hard-coded a limit of 16 emulators to run simultaneously. The only way to run more emulators is to either run emulators inside other emulators, or to change the limit in the C++-code and recompile the emulator software. For the sake of time, I decided to just stick to 16 emulators.
- There are no proper deep-learning libraries available for Java and there are no Java ports available to proper deep-learning libraries in other languages. The best library currently available for Java is DeepLearning4j (DL4J), which is not being (seriously) maintained anymore{, #315}. Apart from the fact that fixing all dependencies took days (since all tutorials and documentation was outdated) and fixing issues such as incorrect internal rounding errors (adding and then subtracting gives in DL4J a different result than first subtracting and then adding) and hard-coded obsolete URLs inside the library was non-trivial, it also has a serious bug somewhere in its memory management, causing the library to crash when it trains multiple network on different threads simultaneously. Since this error occurs somewhere buried deep inside the C-layer, I decided to divide the experiments over 16 separate emulators and run them on every emulator sequentially.

5. Results

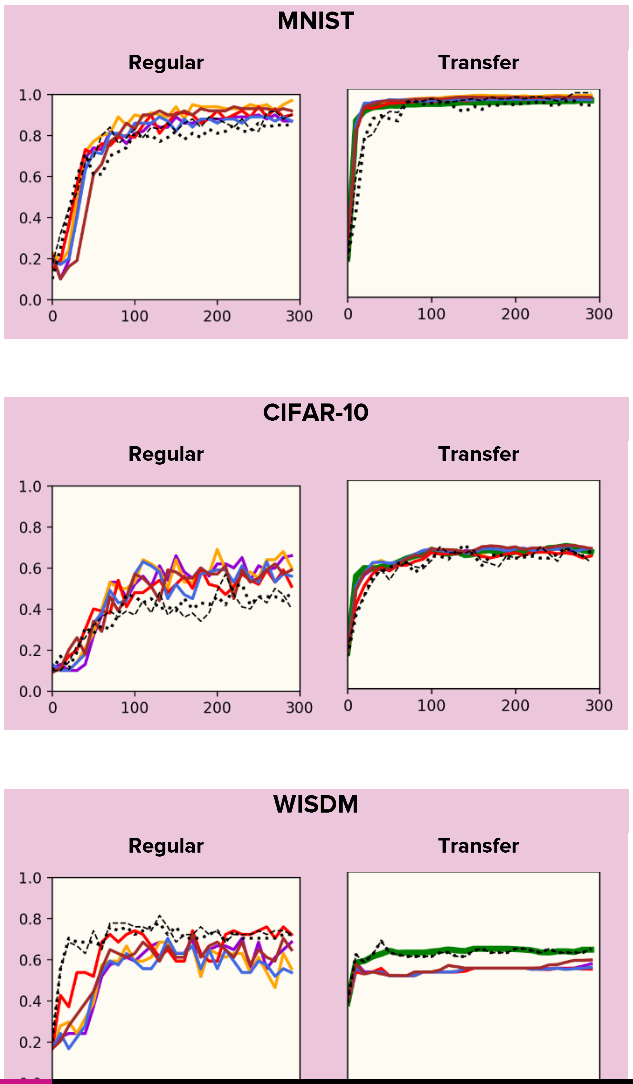
As illustrated in Section 3, Pro-Bristle is hard to compare to existing methods. For instance, it makes the (realistic) assumption that a large public dataset with roughly similar low-level features is available, which is used to pre-initialize the neural network. Since this is an "unfair" advantage with respect to existing GARs that are not dependent on a deep transfer stage, we first show for each experiment the performance of existing GARs and then the performance of existing GARs enhanced by transfer learning and the performance of our proposed GAR Pro-Bristle. Additionally, in contrast to almost all existing work on federated learning, we also evaluate the performance when a node simply ignores all incoming benign/malicious updates. We consider both **Local Min** scenarios where the node has the same tiny amount of data as in the other federated experiments, and **Local Max** scenarios where the node has access to all training data available to all nodes.

The number of parameters that we can vary in the experiments is obviously extremely large. To keep the number of experiments manageable, we will mostly consider scenarios with 10 nodes, with the MNIST dataset, and with all-label-flip attacks. We will first look at the performance of the GARs on the 3 datasets described in Section 4.1 in both a peaceful and a typical Byzantine setting. Subsequently, we evaluate much more Byzantine scenarios where we vary between the type of attack, an i.i.d. vs non-i.i.d. environment, and the number of attackers.

Synchronous, Benign, i.i.d. Environment

MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* DATASET CONFIGURATION ***dataset**: under study, batch size: 5, training data distribution: 10 for each class, test samples per class: 10* NEURAL NETWORK CONFIGURATION *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* TRAINING CONFIGURATION ***#iterations**: under study up to 300, **GAR**: under study, **deep transfer**: under study, communication pattern: all, asynchrony mode: none, #iterations before sending: 1* ATTACK CONFIGURATION *#attackers: 0, attack mode: none*

In a "perfect" environment where all nodes communicate in synchronous rounds, are benign, and the data is distributed i.i.d., all GARs seem to benefit from transfer learning and outperform nodes that ignore updates from their peers. An interesting exception is the WISDM dataset where existing GARs perform worse than just local training, except for Pro-Bristle which performs better. The reason for this is hard to verify, but might be related to the fact that the WISDM dataset contains considerably less training samples than MNIST and CIFAR-10 which might cause local overfitting, and that the WISDM network is recurrent, which TODO

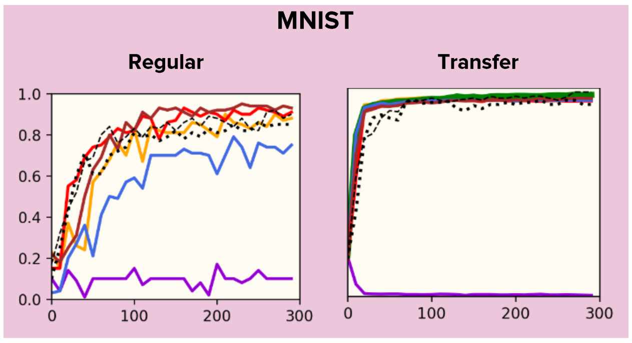


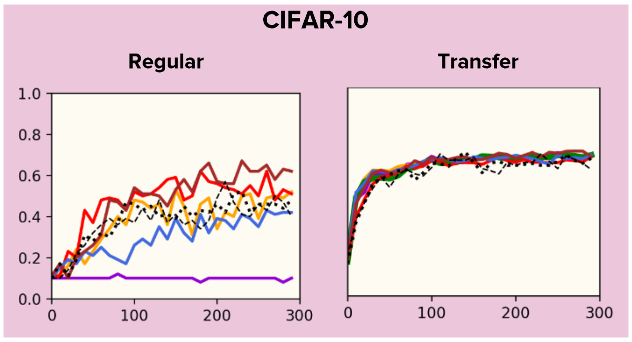
Synchronous, Byzantine All-Label-Flip, i.i.d. Environment

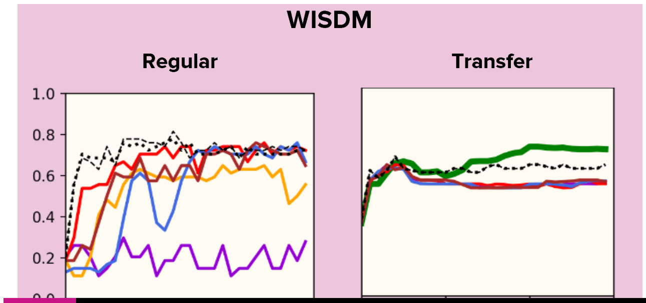
MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* DATASET CONFIGURATION ***dataset**: under study, batch size: 5, training data distribution: 10 for each class, test samples per class: 10* NEURAL NETWORK CONFIGURATION *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* TRAINING CONFIGURATION ***#iterations**: under study up to 300, **GAR**: under study, **deep transfer**: under study, communication pattern: all, asynchrony mode: none, #iterations before sending: 1* ATTACK CONFIGURATION *#attackers: 4, attack mode: all-label-flip*

In a Byzantine environment with a relatively strong all-label-flip attack, it is clear that simple averaging and Bridge are unable to mitigate the attack and that median-based averaging also results in mediocre performance. Krum performs well except in the CIFAR-10 experiment with deep transfer, and Mozi and Pro-Bristle perform well consistently.

🟪 Average 🟧 Bridge 🟦 Median 🟥 Krum 🟫 Mozi 🟩 Pro-Bristle ■ ■ ■ Local min ▬ ▬ Local max



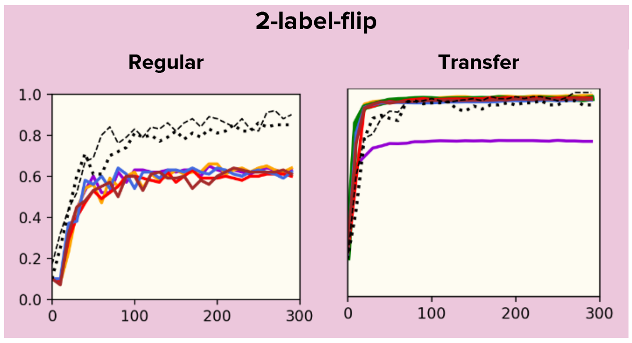


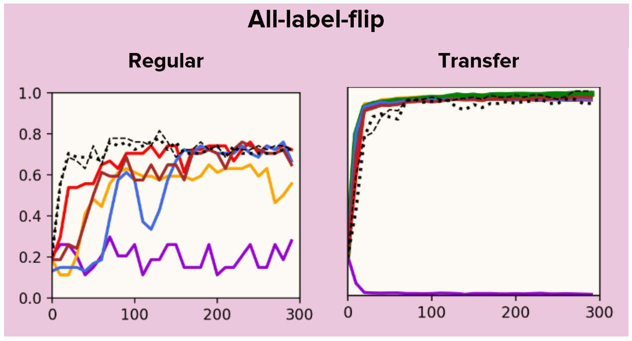


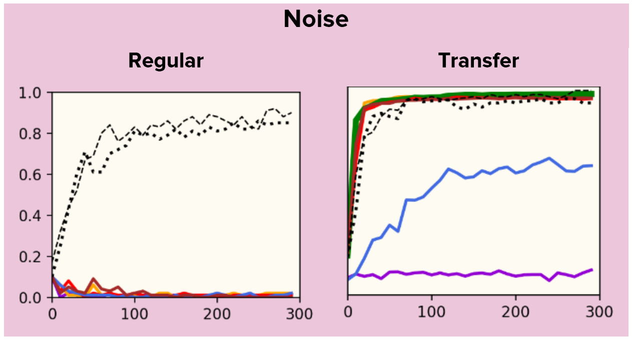
Synchronous, Byzantine, i.i.d. Environment

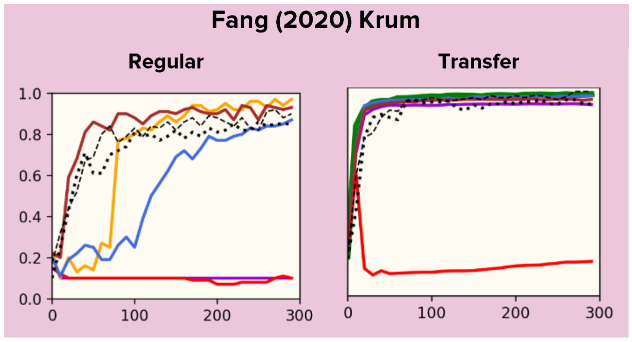
MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* **DATASET CONFIGURATION** *dataset: mnist, batch size: 5, training data distribution: 10 for each class, test samples per class: 10* **NEURAL NETWORK CONFIGURATION** *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* **TRAINING CONFIGURATION** ***#iterations**: under study up to 300, **GAR**: under study, **deep transfer**: under study, communication pattern: all, asynchrony mode: none, #iterations before sending: 1* **ATTACK CONFIGURATION** *#attackers: 4, **attack mode**: under study*

We will further investigate the extent to which the GARs are able to provide protection against Byzantine attacks. A trivial 2-label-flip attack clearly misleads simple averaging but is successfully mitigated by all GARs. A simple biased noise attack manages to completely destroy the model when no GAR is used but is again also successfully mitigated by all GARs. The more advanced Fang (2020) attacks as described in Section 4.8 are softened by Krum and Mozi, but still manage to seriously degrade the model's performance. The median attack interesting enough performs mediocre in the default non-transfer learning, but greatly benefits from using deep transfer. Pro-Bristle performs reasonable against the Fang (2020) Krum attack, and very well against the Fang (2020) Trimmed Mean attack.

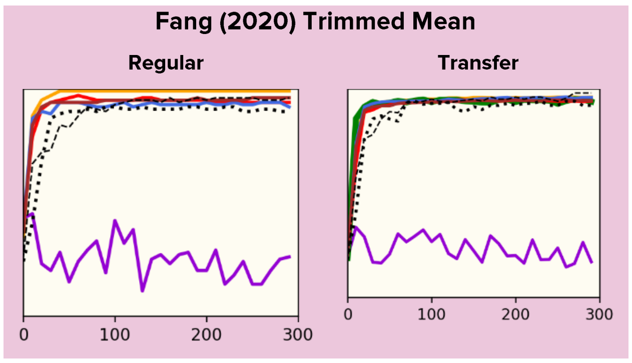






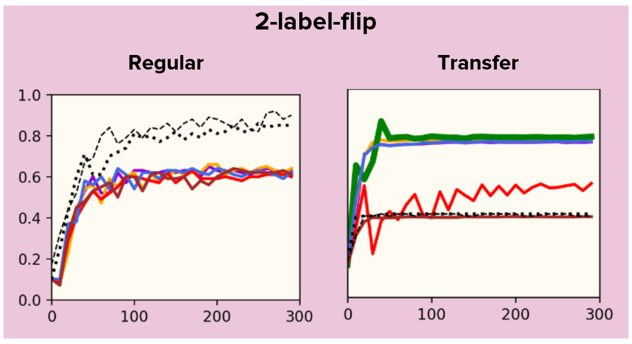


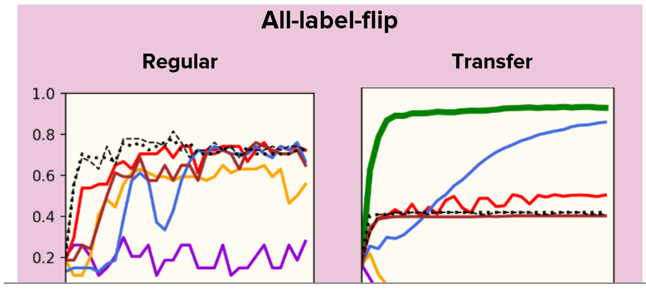
🟪 Average 🟧 Bridge 🟦 Median 🟥 Krum 🟫 Mozi 🟩 Pro-Bristle ▪▪▪▪▪▪ Local min ▬ ▬ ▬ Local max



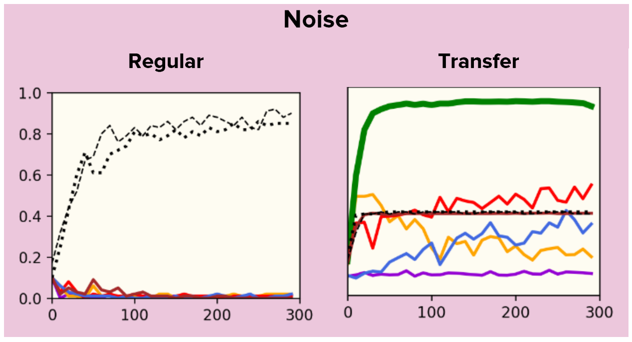
MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* DATASET CONFIGURATION *dataset: mnist, batch size: 5, training data distribution: 4 succeeding classes per node (every next node takes the 4 classes shifted by 1 class compared to the previous node) with 10 samples for each class, test samples per class: 10* NEURAL NETWORK CONFIGURATION *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* TRAINING CONFIGURATION ***#iterations**: under study up to 300, **GAR**: under study, **deep transfer**: under study, communication pattern: all, asynchrony mode: none, #iterations before sending: 1* ATTACK CONFIGURATION *#attackers: 4, **attack mode**: under study*

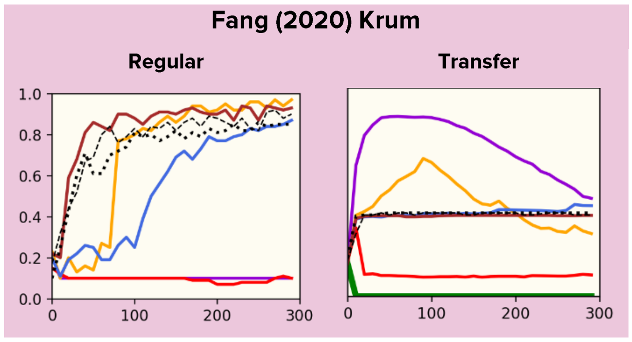
We will further investigate the extent to which the GARs are able to provide protection against Byzantine attacks. A trivial 2-label-flip attack clearly misleads simple averaging but is successfully mitigated by all GARs. A simple biased noise attack manages to completely destroy the model when no GAR is used but is again also successfully mitigated by all GARs. The more advanced Fang (2020) attacks as described in Section 4.8 are softened by Krum and Mozi, but still manage to seriously degrade the model's performance. The median attack interesting enough performs mediocre in the default non-transfer learning, but greatly benefits from using deep transfer. Pro-Bristle performs reasonable against the Fang (2020) Krum attack, and very well against the Fang (2020) Trimmed Mean attack.

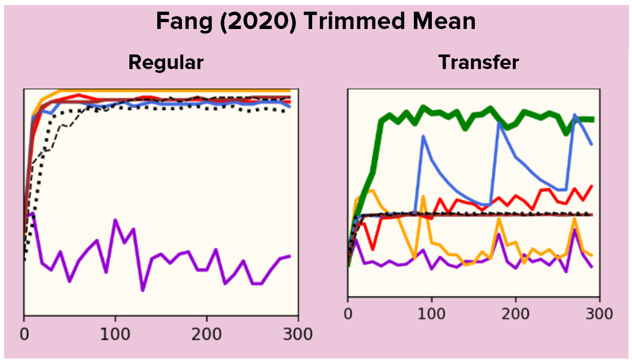




Synchronous, Byzantine, non-i.i.d. Environment







Synchronous, Byzantine with varying num. of attackers, non-i.i.d. Environment

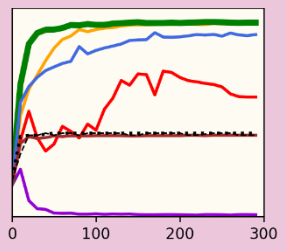
MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* DATASET CONFIGURATION *dataset: mnist, batch size: 5, training data distribution: 4 succeeding classes per node (every next node takes the 4 classes shifted by 1 class compared to the previous node) with 10 samples for each class, test samples per class: 10* NEURAL NETWORK CONFIGURATION *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* TRAINING CONFIGURATION ***#iterations**: under study up to 300, **GAR**: under study, **deep transfer**: under study, communication pattern: all, asynchrony mode: none, #iterations before sending: 1* ATTACK CONFIGURATION ***#attackers**: under study, attack mode: all-label-flip*

🟪	🟧	🟦	🟥	🟫	🟩	■ ■ ■ ■ ■ ■ ■	▬ ▬ ▬ ▪
Average	Bridge	Median	Krum	Mozi	Pro-Bristle	Local min	Local max

We will further investigate the extent to which the GARs are able to provide protection against Byzantine attacks. A trivial 2-label-flip attack clearly misleads simple averaging but is successfully mitigated by all GARs. A simple biased noise attack manages to completely destroy the model when no GAR is used but is again also successfully mitigated by all GARs. The more advanced Fang (2020) attacks as described in Section 4.8 are softened by Krum and Mozi, but still manage to seriously degrade the model's performance. The median attack interesting enough performs mediocre in the default non-transfer learning, but greatly benefits from using deep transfer. Pro-Bristle performs reasonable against the Fang (2020) Krum attack, and very well against the Fang (2020) Trimmed Mean attack.

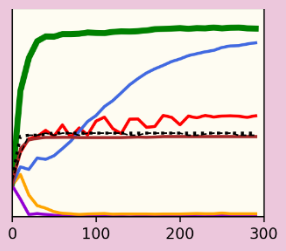
10% all-label-flip attackers

Regular **Transfer**



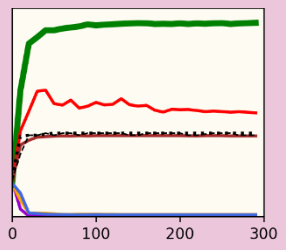
30% all-label-flip attackers

Regular **Transfer**



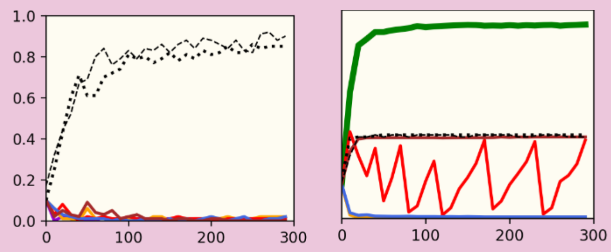
50% all-label-flip attackers

Regular **Transfer**



70% all-label-flip attackers

Regular **Transfer**



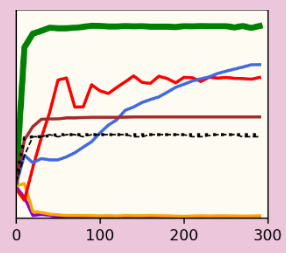
Synchronous, Byzantine, non-i.i.d. Environment with varying comm. protocols

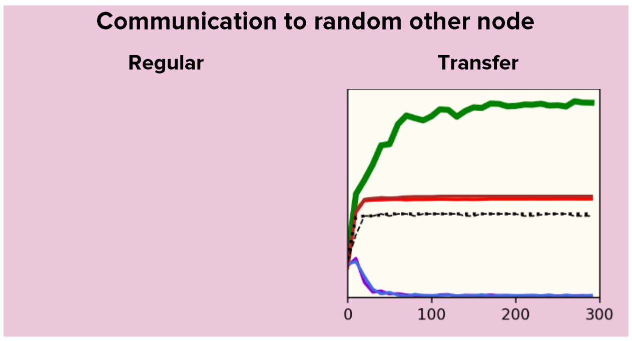
MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* **DATASET CONFIGURATION** *dataset: mnist, batch size: 5, training data distribution: 4 succeeding classes per node (every next node takes the 4 classes shifted by 1 class compared to the previous node) with 10 samples for each class, test samples per class: 10* **NEURAL NETWORK CONFIGURATION** *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* **TRAINING CONFIGURATION** ***#iterations**: under study up to 300, **GAR**: under study, **deep transfer**: under study, **communication pattern**: under study, asynchrony mode: none, #iterations before sending: 1* **ATTACK CONFIGURATION** *#attackers: 10, attack mode: all-label-flip*

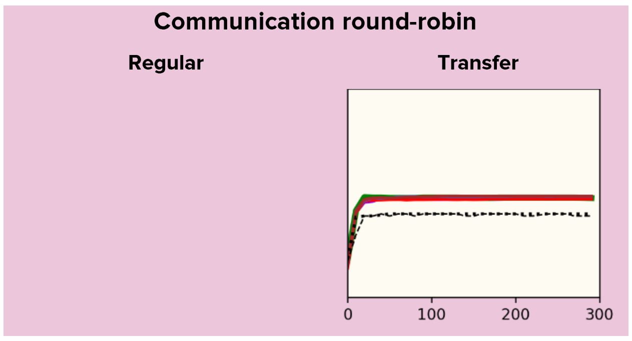
We will further investigate the extent to which the GARs are able to provide protection against Byzantine attacks. A trivial 2-label-flip attack clearly misleads simple averaging but is successfully mitigated by all GARs. A simple biased noise attack manages to completely destroy the model when no GAR is used but is again also successfully mitigated by all GARs. The more advanced Fang (2020) attacks as described in Section 4.8 are softened by Krum and Mozi, but still manage to seriously degrade the model's performance. The median attack interesting enough performs mediocre in the default non-transfer learning, but greatly benefits from using deep transfer. Pro-Bristle performs reasonable against the Fang (2020) Krum attack, and very well against the Fang (2020) Trimmed Mean attack.

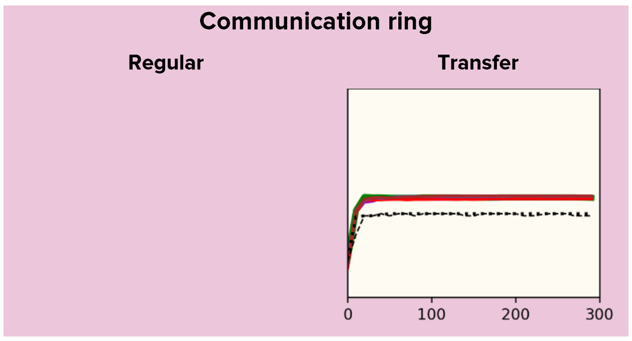
Communication to all other nodes

Regular **Transfer**





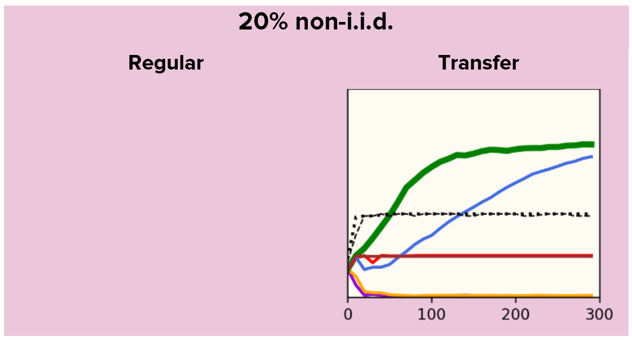


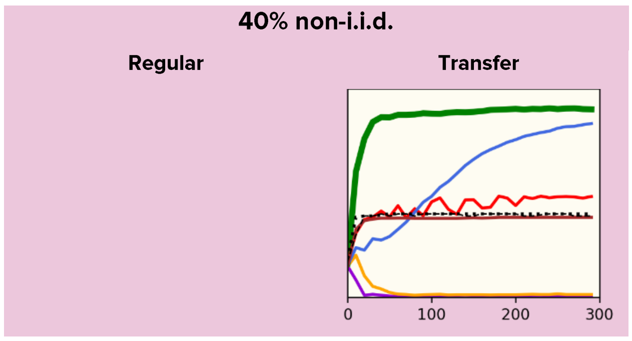


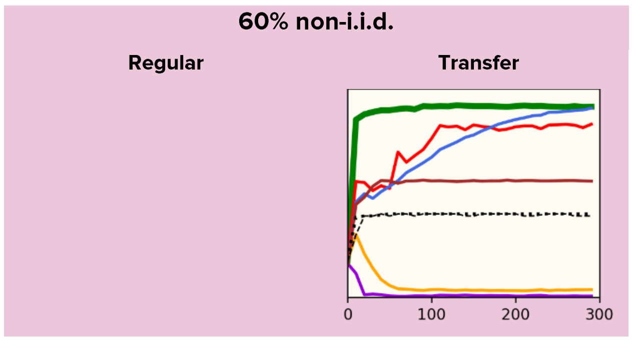
Synchronous, Byzantine Environment with various degrees of non-i.i.d.-ness

MLCONFIGURATION *#nodes: 10, architecture, see Section 4.2* DATASET CONFIGURATION *dataset: mnist, batch size: 5,* ***training data distribution****: under study, test samples per class: 10* NEURAL NETWORK CONFIGURATION *Optimizer: Adam, learning rate: 0.001, momentum: none, l2: 0.005* TRAINING CONFIGURATION ***#iterations****: under study up to 300,* ***GAR****: under study,* ***deep transfer****: under study,* ***communication pattern****: under study, asynchrony mode: none, #iterations before sending: 1* ATTACK CONFIGURATION *#attackers: 10, attack mode: all-label-flip*

We will further investigate the extent to which the GARs are able to provide protection against Byzantine attacks. A trivial 2-label-flip attack clearly misleads simple averaging but is successfully mitigated by all GARs. A simple biased noise attack manages to completely destroy the model when no GAR is used but is again also successfully mitigated by all GARs. The more advanced Fang (2020) attacks as described in Section 4.8 are softened by Krum and Mozi, but still manage to seriously degrade the model's performance. The median attack interesting enough performs mediocre in the default non-transfer learning, but greatly benefits from using deep transfer. Pro-Bristle performs reasonable against the Fang (2020) Krum attack, and very well against the Fang (2020) Trimmed Mean attack.







🟪 Average 🟨 Bridge 🟦 Median 🟥 Krum 🟫 Mozi 🟩 Pro-Bristle ▪▪▪ Local min — — Local max

6. Discussion

Reputatie filter voor de distance-based filter => nog sneller

Meer variabelen evalueren

- #hidden layers
- Meer datasets
- Zonder momentum
- Met/zonder batch normalization

AR1 of andere non-i.i.d. technieken

Toepassing op non-classification problems

7. Conclusion

8. References

1. Biggio, B., Nelson, B., and Laskov, P., 'Poisoning Attacks against Support Vector Machines', *arXiv preprint arXiv:1206.6389*, 2012.
2. Chen, X., Liu, C., Li, B., Lu, K., and Song, D., 'Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning', *arXiv preprint arXiv:1712.05526*, 2017.
3. Koh, P.W. and Liang, P., 'Understanding Black-Box Predictions Via Influence Functions', *arXiv preprint arXiv:1703.04730*, 2017.
4. Suciu, O., Marginean, R., Kaya, Y., Daume III, H., and Dumitras, T., 'When Does Machine Learning {Fail}? Generalized Transferability for Evasion and Poisoning Attacks', in, *27th {USENIX} Security Symposium ({USENIX} Security 18)*, (2018)
5. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V., 'How to Backdoor Federated Learning', in, *International Conference on Artificial Intelligence and Statistics*, (PMLR, 2020)
6. Bhagoji, A.N., Chakraborty, S., Mittal, P., and Calo, S., 'Analyzing Federated Learning through an Adversarial Lens', in, *International Conference on Machine Learning*, (PMLR, 2019)
7. Gu, T., Dolan-Gavitt, B., and Garg, S., 'Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain', *arXiv preprint arXiv:1708.06733*, 2017.
8. Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X., 'Trojaning Attack on Neural Networks', 2017.
9. Nelson, B., Barreno, M., Chi, F.J., Joseph, A.D., Rubinstein, B.I., Saini, U., Sutton, C.A., Tygar, J.D., and Xia, K., 'Exploiting Machine Learning to Subvert Your Spam Filter', *LEET*, 2008, 8, pp. 1-9.
10. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T., 'Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks', in, *Advances in neural information processing systems*, (2018)
11. Yang, G., Gong, N.Z., and Cai, Y., 'Fake Co-Visitation Injection Attacks to Recommender Systems', in, *NDSS*, (2017)
12. Shen, S., Tople, S., and Saxena, P., 'Auror: Defending against Poisoning Attacks in Collaborative Deep Learning Systems', in, *Proceedings of the 32nd Annual Conference on Computer Security Applications*, (2016)
13. Baruch, G., Baruch, M., and Goldberg, Y., 'A Little Is Enough: Circumventing Defenses for Distributed Learning', in, *Advances in neural information processing systems*, (2019)
14. Bhagoji, A.N., Chakraborty, S., Mittal, P., and Calo, S., 'Model Poisoning Attacks in Federated Learning', in, *In Workshop on Security in Machine Learning (SecML), collocated with the 32nd Conference on Neural Information Processing Systems (NeurIPS'18)*, (2018)
15. Sun, Z., Kairouz, P., Suresh, A.T., and McMahan, H.B., 'Can You Really Backdoor Federated Learning?', *arXiv preprint arXiv:1911.07963*, 2019.
16. Xie, C., Huang, K., Chen, P.-Y., and Li, B., 'Dba: Distributed Backdoor Attacks against Federated Learning', in, *International Conference on Learning Representations*, (2019)
17. Zou, M., Shi, Y., Wang, C., Li, F., Song, W., and Wang, Y., 'Potrojan: Powerful Neural-Level Trojan Designs in Deep Learning Models', *arXiv preprint arXiv:1802.03043*, 2018.
18. Koloskova, A., Stich, S.U., and Jaggi, M., 'Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication', *arXiv preprint arXiv:1902.00340*, 2019.
19. Biggio, B., Didaci, L., Fumera, G., and Roli, F., 'Poisoning Attacks to Compromise Face Templates', in, *2013 International Conference on Biometrics (ICB)*, (IEEE, 2013)
20. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B., 'Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning', in, *2018 IEEE Symposium on Security and Privacy (SP)*, (IEEE, 2018)
21. Li, B., Wang, Y., Singh, A., and Vorobeychik, Y., 'Data Poisoning Attacks on Factorization-Based Collaborative Filtering', in, *Advances in neural information processing systems*, (2016)
22. Rubinstein, B.I., Nelson, B., Huang, L., Joseph, A.D., Lau, S.-h., Rao, S., Taft, N., and Tygar, J.D., 'Antidote: Understanding and Defending against Poisoning of Anomaly Detectors', in, *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, (2009)
23. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., and Roli, F., 'Is Feature Selection Secure against Training Data Poisoning?', in, *International Conference on Machine Learning*, (2015)
24. Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N., 'Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges', *Information fusion*, 2020, 58, pp. 52-68.
25. Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J.S., 'A Survey on Distributed Machine Learning', *ACM Computing Surveys (CSUR)*, 2020, 53, (2), pp. 1-33.
26. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., and Isard, M., 'Tensorflow: A System for Large-Scale Machine Learning', in, *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, (2016)
27. Medicare, C.f. and Medicaid Services, The Health Insurance Portability and Accountability Act of 1996 (Hipaa)', (1996)
28. Voigt, P. and Von dem Bussche, A., 'The Eu General Data Protection Regulation (Gdpr)', *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.
29. Chen, Z., Tian, P., Liao, W., and Yu, W., 'Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning', *IEEE Transactions on Network Science and Engineering*, 2020.
30. McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B.A., 'Communication-Efficient Learning of Deep Networks from Decentralized Data', in, *Artificial Intelligence and Statistics*, (PMLR, 2017)
31. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, accessed Date Accessed 2017 Accessed
32. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D., 'Federated Learning for Mobile Keyboard Prediction', *arXiv preprint arXiv:1811.03604*, 2018.
33. Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F., 'Applied Federated Learning:

Improving Google Keyboard Query Suggestions', *arXiv preprint arXiv:1812.02903*, 2018.
34. Chen, M., Mathews, R., Ouyang, T., and Beaufays, F., 'Federated Learning of out-of-Vocabulary Words', *arXiv preprint arXiv:1903.10635*, 2019.
35. Ramaswamy, S., Mathews, R., Rao, K., and Beaufays, F., 'Federated Learning for Emoji Prediction in a Mobile Keyboard', *arXiv preprint arXiv:1906.04329*, 2019.
36. Chen, M., Suresh, A.T., Mathews, R., Wong, A., Allauzen, C., Beaufays, F., and Riley, M., 'Federated Learning of N-Gram Language Models', *arXiv preprint arXiv:1910.03432*, 2019.
37. Yuan, B., Ge, S., and Xing, W., 'A Federated Learning Framework for Healthcare Iot Devices', *arXiv preprint arXiv:2005.05083*, 2020.
38. Leroy, D., Coucke, A., Lavril, T., Gisselbrecht, T., and Dureau, J., 'Federated Learning for Keyword Spotting', in, *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (IEEE, 2019)
39. Sim, K.C., Beaufays, F., Benard, A., Guliani, D., Kabel, A., Khare, N., Lucassen, T., Zadrazil, P., Zhang, H., and Johnson, L., 'Personalization of End-to-End Speech Recognition on Mobile Devices for Named Entities', in, *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, (IEEE, 2019)
40. Niknam, S., Dhillon, H.S., and Reed, J.H., 'Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges', *IEEE Communications Magazine*, 2020, 58, (6), pp. 46-51.
41. Chen, M., Poor, H.V., Saad, W., and Cui, S., 'Wireless Communications for Collaborative Federated Learning in the Internet of Things', *arXiv preprint arXiv:2006.02499*, 2020.
42. Lin, K.-Y. and Huang, W.-R., 'Using Federated Learning on Malware Classification', in, *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, (IEEE, 2020)
43. Sozinov, K., Vlassov, V., and Girdzijauskas, S., 'Human Activity Recognition Using Federated Learning', in, *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, (IEEE, 2018)
44. Nguyen, T.D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., and Sadeghi, A.-R., 'Dïot: A Federated Self-Learning Anomaly Detection System for Iot', in, *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, (IEEE, 2019)
45. Cetin, B., Lazar, A., Kim, J., Sim, A., and Wu, K., 'Federated Wireless Network Intrusion Detection', in, *2019 IEEE International Conference on Big Data (Big Data)*, (IEEE, 2019)
46. Lu, Y., Huang, X., Zhang, K., Maharjan, S., and Zhang, Y., 'Blockchain Empowered Asynchronous Federated Learning for Secure Data Sharing in Internet of Vehicles', *IEEE Transactions on Vehicular Technology*, 2020, 69, (4), pp. 4298-4311.
47. Samarakoon, S., Bennis, M., Saad, W., and Debbah, M., 'Federated Learning for Ultra-Reliable Low-Latency V2v Communications', in, *2018 IEEE Global Communications Conference (GLOBECOM)*, (IEEE, 2018)
48. Gulati, A., Aujla, G.S., Chaudhary, R., Kumar, N., and Obaidat, M.S., 'Deep Learning-Based Content Centric Data Dissemination Scheme for Internet of Vehicles', in, *2018 IEEE International Conference on Communications (ICC)*, (IEEE, 2018)
49. Lu, Y., Huang, X., Dai, Y., Maharjan, S., and Zhang, Y., 'Federated Learning for Data Privacy Preservation in Vehicular Cyber-Physical Systems', *IEEE Network*, 2020, 34, (3), pp. 50-56.
50. Liu, Y., James, J., Kang, J., Niyato, D., and Zhang, S., 'Privacy-Preserving Traffic Flow Prediction: A Federated Learning Approach', *IEEE Internet of Things Journal*, 2020.
51. Mowla, N.I., Tran, N.H., Doh, I., and Chae, K., 'Federated Learning-Based Cognitive Detection of Jamming Attack in Flying Ad-Hoc Network', *IEEE Access*, 2019, 8, pp. 4338-4350.
52. Liu, Y., Huang, A., Luo, Y., Huang, H., Liu, Y., Chen, Y., Feng, L., Chen, T., Yu, H., and Yang, Q., 'Fedvision: An Online Visual Object Detection Platform Powered by Federated Learning', in, *AAAI*, (2020)
53. Schneble, W. and Thamilarasu, G., 'Attack Detection Using Federated Learning in Medical Cyber-Physical Systems'.
54. Lu, S., Zhang, Y., and Wang, Y., 'Decentralized Federated Learning for Electronic Health Records', in, *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, (IEEE, 2020)
55. Brisimi, T.S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I.C., and Shi, W., 'Federated Learning of Predictive Models from Federated Electronic Health Records', *International journal of medical informatics*, 2018, 112, pp. 59-67.
56. Xu, J. and Wang, F., 'Federated Learning for Healthcare Informatics', *arXiv preprint arXiv:1911.06270*, 2019.
57. Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H., Albarqouni, S., Bakas, S., Galtier, M.N., Landman, B., and Maier-Hein, K., 'The Future of Digital Health with Federated Learning', *arXiv preprint arXiv:2003.08119*, 2020.
58. Konečný, J., McMahan, H.B., Ramage, D., and Richtárik, P., 'Federated Optimization: Distributed Machine Learning for on-Device Intelligence', *arXiv preprint arXiv:1610.02527*, 2016.
59. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., and McMahan, H.B., 'Towards Federated Learning at Scale: System Design', *arXiv preprint arXiv:1902.01046*, 2019.
60. Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J., 'Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent', in, *Advances in neural information processing systems*, (2017)
61. Xie, X., Ma, L., Wang, H., Li, Y., Liu, Y., and Li, X., 'Diffchaser: Detecting Disagreements for Deep Neural Networks', in, *IJCAI*, (2019)
62. Dobbe, R., Fridovich-Keil, D., and Tomlin, C., 'Fully Decentralized Policies for Multi-Agent Systems: An Information Theoretic Approach', in, *Advances in neural information processing systems*, (2017)
63. Tang, H., Gan, S., Zhang, C., Zhang, T., and Liu, J., 'Communication Compression for Decentralized Training', in, *Advances in neural information processing systems*, (2018)

64. Lalitha, A., Wang, X., Kilinc, O., Lu, Y., Javidi, T., and Koushanfar, F., 'Decentralized Bayesian Learning over Graphs', *arXiv preprint arXiv:1905.10466*, 2019.

65. Nedic, A. and Ozdaglar, A., 'Distributed Subgradient Methods for Multi-Agent Optimization', *IEEE Transactions on Automatic Control*, 2009, 54, (1), pp. 48-61.

66. Harinath, D., Satyanarayana, P., and Murthy, M., 'A Review on Security Issues and Attacks in Distributed Systems', *Journal of Advances in Information Technology*, 2017, 8, (1).

67. Lamport, L., Shostak, R., and Pease, M., The Byzantine Generals Problem', *Concurrency: The Works of Leslie Lamport*, (2019)

68. Blanchard, P., Guerraoui, R., and Stainer, J., 'Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent', in, *Advances in neural information processing systems*, (2017)

69. Chen, Y., Su, L., and Xu, J., 'Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent', *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2017, 1, (2), pp. 1-25.

70. Zhang, Q., Cheng, L., and Boutaba, R., 'Algorithms and Architectures for Parallel Processing', *J. Int. Serv. Appl*, 2010, 1, (1), pp. 7-18.

71. El-Mhamdi, E.-M. and Guerraoui, R., 'Fast and Secure Distributed Learning in High Dimension', *arXiv preprint arXiv:1905.04374*, 2019.

72. Haykin, S., *Neural Networks and Learning Machines, 3/E*, (Pearson Education India, 2010)

73. Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J., 'Adding Gradient Noise Improves Learning for Very Deep Networks', *arXiv preprint arXiv:1511.06807*, 2015.

74. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P.T.P., 'On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima', *arXiv preprint arXiv:1609.04836*, 2016.

75. Kleinberg, R., Li, Y., and Yuan, Y., 'An Alternative View: When Does Sgd Escape Local Minima?', *arXiv preprint arXiv:1802.06175*, 2018.

76. Bottou, L., 'Online Learning and Stochastic Approximations', *On-line learning in neural networks*, 1998, 17, (9), p. 142.

77. Chen, J., Pan, X., Monga, R., Bengio, S., and Jozefowicz, R., 'Revisiting Distributed Synchronous Sgd', *arXiv preprint arXiv:1604.00981*, 2016.

78. Wu, W., He, L., Lin, W., Mao, R., Maple, C., and Jarvis, S.A., 'Safa: A Semi-Asynchronous Protocol for Fast Federated Learning with Low Overhead', *IEEE Transactions on Computers*, 2020.

79. Xie, C., Koyejo, S., and Gupta, I., 'Asynchronous Federated Optimization', *arXiv preprint arXiv:1903.03934*, 2019.

80. Chen, Y., Ning, Y., and Rangwala, H., 'Asynchronous Online Federated Learning for Edge Devices', *arXiv preprint arXiv:1911.02134*, 2019.

81. Sprague, M.R., Jalalirad, A., Scavuzzo, M., Capota, C., Neun, M., Do, L., and Kopp, M., 'Asynchronous Federated Learning for Geospatial Applications', in, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, (Springer, 2018)

82. Mohammad, U. and Sorour, S., 'Adaptive Task Allocation for Asynchronous Federated Mobile Edge Learning', *arXiv preprint arXiv:1905.01656*, 2019.

83. Yang, Y.-R. and Li, W.-J., 'Basgd: Buffered Asynchronous Sgd for Byzantine Learning', *arXiv preprint arXiv:2003.00937*, 2020.

84. Chen, M., Mao, B., and Ma, T., 'Efficient and Robust Asynchronous Federated Learning with Stragglers', in, *Submitted to International Conference on Learning Representations*, (2019)

85. Roy, A.G., Siddiqui, S., Pölsterl, S., Navab, N., and Wachinger, C., 'Braintorrent: A Peer-to-Peer Environment for Decentralized Federated Learning', *arXiv preprint arXiv:1905.06731*, 2019.

86. Hu, C., Jiang, J., and Wang, Z., 'Decentralized Federated Learning: A Segmented Gossip Approach', *arXiv preprint arXiv:1908.07782*, 2019.

87. Hegedűs, I., Danner, G., and Jelasity, M., 'Gossip Learning as a Decentralized Alternative to Federated Learning', in, *IFIP International Conference on Distributed Applications and Interoperable Systems*, (Springer, 2019)

88. Haseltalab, A. and Akar, M., 'Approximate Byzantine Consensus in Faulty Asynchronous Networks', in, *2015 American Control Conference (ACC)*, (IEEE, 2015)

89. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V., 'Federated Optimization in Heterogeneous Networks', *arXiv preprint arXiv:1812.06127*, 2018.

90. Nilsson, A., Smith, S., Ulm, G., Gustavsson, E., and Jirstrand, M., 'A Performance Evaluation of Federated Learning Algorithms', in, *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, (2018)

91. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., and Suresh, A.T., 'Scaffold: Stochastic Controlled Averaging for on-Device Federated Learning', *arXiv preprint arXiv:1910.06378*, 2019.

92. Robbins, H. and Monro, S., 'A Stochastic Approximation Method', *The annals of mathematical statistics*, 1951, pp. 400-407.

93. Kingma, D.P. and Ba, J., 'Adam: A Method for Stochastic Optimization', *arXiv preprint arXiv:1412.6980*, 2014.

94. Mukkamala, M.C. and Hein, M., 'Variants of Rmsprop and Adagrad with Logarithmic Regret Bounds', *arXiv preprint arXiv:1706.05507*, 2017.

95. Damaskinos, G., El Mhamdi, E.M., Guerraoui, R., Guirguis, A.H.A., and Rouault, S.L.A., 'Aggregathor: Byzantine Machine Learning Via Robust Gradient Aggregation', in, *The Conference on Systems and Machine Learning (SysML), 2019*, (2019)

96. Zhang, S., Choromanska, A.E., and LeCun, Y., 'Deep Learning with Elastic Averaging Sgd', in, *Advances in neural information processing systems*, (2015)

97. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., and Su, B.-Y., 'Scaling Distributed Machine Learning with the Parameter Server', in, *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, (2014)

98. Xing, E.P., Ho, Q., Xie, P., and Wei, D., 'Strategies and Principles of Distributed Machine Learning on Big Data', *Engineering*, 2016, 2, (2), pp. 179-195.

99. Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P., 'Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates', *arXiv preprint arXiv:1803.01498*, 2018.
100. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., and Roli, F., 'Towards Poisoning of Deep Learning Algorithms with Back-Gradient Optimization', in, *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, (2017)
101. Wang, B. and Gong, N.Z., 'Attacking Graph-Based Classification Via Manipulating the Graph Structure', in, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, (2019)
102. Mei, S. and Zhu, X., 'Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners', in, *AAAI*, (2015)
103. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D., 'Robust Physical-World Attacks on Deep Learning Visual Classification', in, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2018)
104. Muñoz-González, L., Co, K.T., and Lupu, E.C., 'Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging', *arXiv preprint arXiv:1909.05125*, 2019.
105. Dwork, C., 'Differential Privacy: A Survey of Results', in, *International conference on theory and applications of models of computation*, (Springer, 2008)
106. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., and Zhang, L., 'Deep Learning with Differential Privacy', in, *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, (2016)
107. Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q., and Poor, H.V., 'Federated Learning with Differential Privacy: Algorithms and Performance Analysis', *IEEE Transactions on Information Forensics and Security*, 2020, 15, pp. 3454-3469.
108. Fung, C., Yoon, C.J., and Beschastnikh, I., 'Mitigating Sybils in Federated Learning Poisoning', *arXiv preprint arXiv:1808.04866*, 2018.
109. Tolpegin, V., Truex, S., Gursoy, M.E., and Liu, L., 'Data Poisoning Attacks against Federated Learning Systems', in, *European Symposium on Research in Computer Security*, (Springer, 2020)
110. Mhamdi, E.M.E., Guerraoui, R., and Rouault, S., 'The Hidden Vulnerability of Distributed Learning in Byzantium', *arXiv preprint arXiv:1802.07927*, 2018.
111. Fang, M., Cao, X., Jia, J., and Gong, N., 'Local Model Poisoning Attacks to Byzantine-Robust Federated Learning', in, *29th {USENIX} Security Symposium ({USENIX} Security 20)*, (2020)
112. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., and Cummings, R., 'Advances and Open Problems in Federated Learning', *arXiv preprint arXiv:1912.04977*, 2019.
113. Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D., 'Attack of the Tails: Yes, You Really Can Backdoor Federated Learning', *arXiv preprint arXiv:2007.05084*, 2020.
114. Xie, C., Koyejo, O., and Gupta, I., 'Generalized Byzantine-Tolerant Sgd', *arXiv preprint arXiv:1802.10116*, 2018.
115. Guo, S., Zhang, T., Xie, X., Ma, L., Xiang, T., and Liu, Y., 'Towards Byzantine-Resilient Learning in Decentralized Systems', *arXiv preprint arXiv:2002.08569*, 2020.
116. Huber, P.J., *Robust Statistics*, (John Wiley & Sons, 2004)
117. Cretu, G.F., Stavrou, A., Locasto, M.E., Stolfo, S.J., and Keromytis, A.D., 'Casting out Demons: Sanitizing Training Data for Anomaly Sensors', in, *2008 IEEE Symposium on Security and Privacy (sp 2008)*, (IEEE, 2008)
118. Bhatia, K., Jain, P., and Kar, P., 'Robust Regression Via Hard Thresholding', in, *Advances in neural information processing systems*, (2015)
119. Diakonikolas, I., Kamath, G., Kane, D., Li, J., Moitra, A., and Stewart, A., 'Robust Estimators in High-Dimensions without the Computational Intractability', *SIAM Journal on Computing*, 2019, 48, (2), pp. 742-864.
120. Lai, K.A., Rao, A.B., and Vempala, S., 'Agnostic Estimation of Mean and Covariance', in, *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, (IEEE, 2016)
121. Yang, Z., Gang, A., and Bajwa, W.U., 'Adversary-Resilient Inference and Machine Learning: From Distributed to Decentralized', *stat*, 2019, 1050, p. 23.
122. Su, L. and Vaidya, N.H., 'Fault-Tolerant Distributed Optimization (Part Iv): Constrained Optimization with Arbitrary Directed Networks', *arXiv preprint arXiv:1511.01821*, 2015.
123. Sundaram, S. and Gharesifard, B., 'Distributed Optimization under Adversarial Nodes', *IEEE Transactions on Automatic Control*, 2018, 64, (3), pp. 1063-1076.
124. Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D., 'Draco: Byzantine-Resilient Distributed Training Via Redundant Gradients', *arXiv preprint arXiv:1803.09877*, 2018.
125. Alon, N., Matias, Y., and Szegedy, M., 'The Space Complexity of Approximating the Frequency Moments', *Journal of Computer and system sciences*, 1999, 58, (1), pp. 137-147.
126. Jerrum, M.R., Valiant, L.G., and Vazirani, V.V., 'Random Generation of Combinatorial Structures from a Uniform Distribution', *Theoretical computer science*, 1986, 43, pp. 169-188.
127. Lerasle, M. and Oliveira, R.I., 'Robust Empirical Mean Estimators', *arXiv preprint arXiv:1112.3914*, 2011.
128. Minsker, S., 'Geometric Median and Robust Estimation in Banach Spaces', *Bernoulli*, 2015, 21, (4), pp. 2308-2335.
129. Minsker, S., 'Distributed Statistical Estimation and Rates of Convergence in Normal Approximation', *Electronic Journal of Statistics*, 2019, 13, (2), pp. 5213-5252.
130. Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A., 'Signsgd: Compressed Optimisation for Non-Convex Problems', *arXiv preprint arXiv:1802.04434*, 2018.
131. Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A., 'Signsgd with Majority Vote Is Communication Efficient and Fault Tolerant', *arXiv preprint arXiv:1810.05291*, 2018.
132. Chen, X., Chen, T., Sun, H., Wu, Z.S., and Hong, M., 'Distributed Training with Heterogeneous Data: Bridging Median- and Mean-Based Algorithms', *arXiv preprint arXiv:1906.01736*, 2019.
133. Sohn, J.-y., Han, D.-J., Choi, B., and Moon, J., 'Election Coding for Distributed Learning: Protecting Signsgd against Byzantine Attacks', *arXiv preprint arXiv:1910.06093*, 2019.

134. Li, L., Xu, W., Chen, T., Giannakis, G.B., and Ling, Q., 'Rsa: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets', in, *Proceedings of the AAAI Conference on Artificial Intelligence*, (2019)
135. Cao, D., Chang, S., Lin, Z., Liu, G., and Sun, D., 'Understanding Distributed Poisoning Attack in Federated Learning', in, *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, (IEEE, 2019)
136. Yang, Z. and Bajwa, W.U., 'Byrdie: Byzantine-Resilient Distributed Coordinate Descent for Decentralized Learning', *IEEE Transactions on Signal and Information Processing over Networks*, 2019, 5, (4), pp. 611-627.
137. Yang, Z. and Bajwa, W.U., 'Bridge: Byzantine-Resilient Decentralized Gradient Descent', *arXiv preprint arXiv:1908.08098*, 2019.
138. Peng, J. and Ling, Q., 'Byzantine-Robust Decentralized Stochastic Optimization', in, *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (IEEE, 2020)
139. He, L., Karimireddy, S.P., and Jaggi, M., 'Byzantine-Robust Learning on Heterogeneous Datasets Via Resampling', *arXiv preprint arXiv:2006.09365*, 2020.
140. Gupta, N., Liu, S., and Vaidya, N.H., 'Byzantine Fault-Tolerant Distributed Machine Learning Using Stochastic Gradient Descent (Sgd) and Norm-Based Comparative Gradient Elimination (Cge)', *arXiv preprint arXiv:2008.04699*, 2020.
141. Barreno, M., Nelson, B., Joseph, A.D., and Tygar, J.D., 'The Security of Machine Learning', *Machine Learning*, 2010, 81, (2), pp. 121-148.
142. Tran, B., Li, J., and Madry, A., 'Spectral Signatures in Backdoor Attacks', in, *Advances in neural information processing systems*, (2018)
143. Zhao, L., Hu, S., Wang, Q., Jiang, J., Chao, S., Luo, X., and Hu, P., 'Shielding Collaborative Learning: Mitigating Poisoning Attacks through Client-Side Detection', *IEEE Transactions on Dependable and Secure Computing*, 2020.
144. Jin, R., He, X., and Dai, H., 'Distributed Byzantine Tolerant Stochastic Gradient Descent in the Era of Big Data', in, *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, (IEEE, 2019)
145. Xie, C., Koyejo, S., and Gupta, I., 'Zeno: Distributed Stochastic Gradient Descent with Suspicion-Based Fault-Tolerance', in, *International Conference on Machine Learning*, (PMLR, 2019)
146. Xie, C., Koyejo, S., and Gupta, I., 'Zeno++: Robust Fully Asynchronous Sgd', *arXiv preprint arXiv:1903.07020*, 2019.
147. Zhao, Y., Chen, J., Zhang, J., Wu, D., Teng, J., and Yu, S., 'Pdgan: A Novel Poisoning Defense Method in Federated Learning Using Generative Adversarial Network', in, *International Conference on Algorithms and Architectures for Parallel Processing*, (Springer, 2019)
148. Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., and Qi, H., 'Beyond Inferring Class Representatives: User-Level Privacy Leakage from Federated Learning', in, *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, (IEEE, 2019)
149. Mothukuri, V., Parizi, R.M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G., 'A Survey on Security and Privacy of Federated Learning', *Future Generation Computer Systems*, 2020.
150. Liu, K., Dolan-Gavitt, B., and Garg, S., 'Fine-Pruning: Defending against Backdooring Attacks on Deep Neural Networks', in, *International Symposium on Research in Attacks, Intrusions, and Defenses*, (Springer, 2018)
151. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B.Y., 'Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks', in, *2019 IEEE Symposium on Security and Privacy (SP)*, (IEEE, 2019)
152. Jiang, Y., Wang, S., Ko, B.J., Lee, W.-H., and Tassiulas, L., 'Model Pruning Enables Efficient Federated Learning on Edge Devices', *arXiv preprint arXiv:1909.12326*, 2019.
153. Koh, P.W., Steinhardt, J., and Liang, P., 'Stronger Data Poisoning Attacks Break Data Sanitization Defenses', *arXiv preprint arXiv:1811.00741*, 2018.
154. Steinhardt, J., Koh, P.W.W., and Liang, P.S., 'Certified Defenses for Data Poisoning Attacks', in, *Advances in neural information processing systems*, (2017)
155. Qiao, M. and Valiant, G., 'Learning Discrete Distributions from Untrusted Batches', *arXiv preprint arXiv:1711.08113*, 2017.
156. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B., 'Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering', *arXiv preprint arXiv:1811.03728*, 2018.
157. Chou, E., Tramèr, F., Pellegrino, G., and Boneh, D., 'Sentinet: Detecting Physical Attacks against Deep Learning Systems', *arXiv preprint arXiv:1812.00292*, 2018.
158. Shen, Y. and Sanghavi, S., 'Learning with Bad Training Data Via Iterative Trimmed Loss Minimization', in, *International Conference on Machine Learning*, (PMLR, 2019)
159. Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., and Stewart, A., 'Sever: A Robust Meta-Algorithm for Stochastic Optimization', in, *International Conference on Machine Learning*, (2019)
160. Regatti, J. and Gupta, A., 'Befriending the Byzantines through Reputation Scores', *arXiv preprint arXiv:2006.13421*, 2020.
161. Azulay, S., Raz, L., Globerson, A., Koren, T., and Afek, Y., 'Holdout Sgd: Byzantine Tolerant Federated Learning', *arXiv preprint arXiv:2008.04612*, 2020.
162. Schmid, R., Pfitzner, B., Beilharz, J., Arnrich, B., and Polze, A., 'Tangle Ledger for Decentralized Learning', in, *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, (IEEE, 2020)
163. Kim, H., Kim, S.-H., Hwang, J.Y., and Seo, C., 'Efficient Privacy-Preserving Machine Learning for Blockchain Network', *IEEE Access*, 2019, 7, pp. 136481-136495.
164. Shayan, M., Fung, C., Yoon, C.J., and Beschastnikh, I., 'Biscotti: A Ledger for Private and Secure Peer-to-Peer Machine Learning', *arXiv preprint arXiv:1811.09904*, 2018.
165. Chen, X., Ji, J., Luo, C., Liao, W., and Li, P., 'When Machine Learning Meets Blockchain: A Decentralized, Privacy-Preserving and Secure Design', in, *2018 IEEE International Conference on Big Data (Big Data)*, (IEEE, 2018)
166. Kim, H., Park, J., Bennis, M., and Kim, S.-L., 'Blockchained on-Device Federated Learning', *IEEE Communications Letters*, 2019, 24, (6), pp. 1279-1283.

167. Kim, Y.J. and Hong, C.S., 'Blockchain-Based Node-Aware Dynamic Weighting Methods for Improving Federated Learning Performance', in, *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, (IEEE, 2019)
168. Weng, J., Weng, J., Zhang, J., Li, M., Zhang, Y., and Luo, W., 'Deepchain: Auditable and Privacy-Preserving Deep Learning with Blockchain-Based Incentive', *IEEE Transactions on Dependable and Secure Computing*, 2019.
169. Zhou, S., Huang, H., Chen, W., Zhou, P., Zheng, Z., and Guo, S., 'Pirate: A Blockchain-Based Secure Framework of Distributed Machine Learning in 5g Networks', *IEEE Network*, 2020.
170. Toyoda, K. and Zhang, A.N., 'Mechanism Design for an Incentive-Aware Blockchain-Enabled Federated Learning Platform', in, *2019 IEEE International Conference on Big Data (Big Data)*, (IEEE, 2019)
171. Majeed, U. and Hong, C.S., 'Flchain: Federated Learning Via Mec-Enabled Blockchain Network', in, *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, (IEEE, 2019)
172. Salah, K., Rehman, M.H.U., Nizamuddin, N., and Al-Fuqaha, A., 'Blockchain for Ai: Review and Open Research Challenges', *IEEE Access*, 2019, 7, pp. 10127-10149.
173. Bao, X., Su, C., Xiong, Y., Huang, W., and Hu, Y., 'Flchain: A Blockchain for Auditable Federated Learning with Trust and Incentive', in, *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, (IEEE, 2019)
174. TOYODA, K., MATHIOPOULOS, P.T., and ZHANG, A.N., 'Novel Blockchain-Based Incentive-Aware Federated Learning Platform with Mechanism Design'.
175. Zhao, Y., Zhao, J., Jiang, L., Tan, R., and Niyato, D., 'Mobile Edge Computing, Blockchain and Reputation-Based Crowdsourcing Iot Federated Learning: A Secure, Decentralized and Privacy-Preserving System', *arXiv preprint arXiv:1906.10893*, 2019.
176. Kang, J., Xiong, Z., Niyato, D., Yu, H., Liang, Y.-C., and Kim, D.I., 'Incentive Design for Efficient Federated Learning in Mobile Networks: A Contract Theory Approach', in, *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, (IEEE, 2019)
177. Kang, J., Xiong, Z., Niyato, D., Xie, S., and Zhang, J., 'Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory', *IEEE Internet of Things Journal*, 2019, 6, (6), pp. 10700-10714.
178. Zhao, Y., Zhao, J., Jiang, L., Tan, R., Niyato, D., Li, Z., Lyu, L., and Liu, Y., 'Privacy-Preserving Blockchain-Based Federated Learning for Iot Devices', *IEEE Internet of Things Journal*, 2020.
179. Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W., and Ilie-Zudor, E., 'Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study', *Applied Sciences*, 2018, 8, (12), p. 2663.
180. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N., 'Algorand: Scaling Byzantine Agreements for Cryptocurrencies', in, *Proceedings of the 26th Symposium on Operating Systems Principles*, (2017)
181. Zhan, Y., Li, P., Qu, Z., Zeng, D., and Guo, S., 'A Learning-Based Incentive Mechanism for Federated Learning', *IEEE Internet of Things Journal*, 2020.
182. Khan, L.U., Tran, N.H., Pandey, S.R., Saad, W., Han, Z., Nguyen, M.N., and Hong, C.S., 'Federated Learning for Edge Networks: Resource Optimization and Incentive Mechanism', *arXiv preprint arXiv:1911.05642*, 2019.
183. Yu, H., Liu, Z., Liu, Y., Chen, T., Cong, M., Weng, X., Niyato, D., and Yang, Q., 'A Fairness-Aware Incentive Scheme for Federated Learning', in, *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, (2020)
184. Hu, R. and Gong, Y., 'Trading Data for Learning: Incentive Mechanism for on-Device Federated Learning', *arXiv preprint arXiv:2009.05604*, 2020.
185. Zeng, R., Zhang, S., Wang, J., and Chu, X., 'Fmore: An Incentive Scheme of Multi-Dimensional Auction for Federated Learning in Mec', *arXiv preprint arXiv:2002.09699*, 2020.
186. Yu, H., Liu, Z., Liu, Y., Chen, T., Cong, M., Weng, X., Niyato, D., and Yang, Q., 'A Sustainable Incentive Scheme for Federated Learning', *IEEE Intelligent Systems*, 2020.
187. Le, T.H.T., Tran, N.H., Tun, Y.K., Nguyen, M.N., Pandey, S.R., Han, Z., and Hong, C.S., 'An Incentive Mechanism for Federated Learning in Wireless Cellular Network: An Auction Approach', *arXiv preprint arXiv:2009.10269*, 2020.
188. Cong, M., Yu, H., Weng, X., Qu, J., Liu, Y., and Yiu, S.M., 'A Vcg-Based Fair Incentive Mechanism for Federated Learning', *arXiv preprint arXiv:2008.06680*, 2020.
189. Ding, N., Fang, Z., and Huang, J., 'Incentive Mechanism Design for Federated Learning with Multi-Dimensional Private Information', in, *2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, (IEEE, 2020)
190. Lim, W.Y.B., Xiong, Z., Kang, J., Niyato, D., Zhang, Y., Leung, C., and Miao, C., 'An Incentive Scheme for Federated Learning in the Sky', in, *Proceedings of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*, (2020)
191. Lim, W.Y.B., Xiong, Z., Miao, C., Niyato, D., Yang, Q., Leung, C., and Poor, H.V., 'Hierarchical Incentive Mechanism Design for Federated Machine Learning in Mobile Networks', *IEEE Internet of Things Journal*, 2020.
192. Ng, K.L., Chen, Z., Zelei Liu, H.Y., Liu, Y., and Yang, Q., 'A Multi-Player Game for Studying Federated Learning Incentive Schemes'.
193. Pandey, S.R., Suhail, S., Tun, Y.K., Alsenwi, M., and Hong, C.S., 'An Incentive Design to Perform Federated Learning'.
194. Feng, S., Niyato, D., Wang, P., Kim, D.I., and Liang, Y.-C., 'Joint Service Pricing and Cooperative Relay Communication for Federated Learning', in, *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, (IEEE, 2019)
195. Sarikaya, Y. and Ercetin, O., 'Motivating Workers in Federated Learning: A Stackelberg Game Perspective', *IEEE Networking Letters*, 2019, 2, (1), pp. 23-27.
196. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., and Seth, K., 'Practical Secure Aggregation for Federated Learning on User-Held Data', *arXiv preprint arXiv:1611.04482*, 2016.

197. Sabt, M., Achemlal, M., and Bouabdallah, A., 'Trusted Execution Environment: What It Is, and What It Is Not', in, *2015 IEEE Trustcom/BigDataSE/ISPA*, (IEEE, 2015)
198. Mo, F. and Haddadi, H., 'Efficient and Private Federated Learning Using Tee', in, *EuroSys*, (2019)
199. Chen, Y., Luo, F., Li, T., Xiang, T., Liu, Z., and Li, J., 'A Training-Integrity Privacy-Preserving Federated Learning Scheme with Trusted Execution Environment', *Information Sciences*, 2020, 522, pp. 69-79.
200. Ji, J., Chen, X., Wang, Q., Yu, L., and Li, P., 'Learning to Learn Gradient Aggregation by Gradient Descent', in, *IJCAI*, (2019)
201. Li, S., Cheng, Y., Wang, W., Liu, Y., and Chen, T., 'Learning to Detect Malicious Clients for Robust Federated Learning', *arXiv preprint arXiv:2002.00211*, 2020.
202. Rajput, S., Wang, H., Charles, Z., and Papailiopoulos, D., 'Detox: A Redundancy-Based Framework for Faster and More Robust Gradient Aggregation', in, *Advances in neural information processing systems*, (2019)
203. Data, D., Song, L., and Diggavi, S., 'Data Encoding for Byzantine-Resilient Distributed Optimization', *arXiv preprint arXiv:1907.02664*, 2019.
204. Wang, S., Tuor, T., Salonidis, T., Leung, K.K., Makaya, C., He, T., and Chan, K., 'Adaptive Federated Learning in Resource Constrained Edge Computing Systems', *IEEE Journal on Selected Areas in Communications*, 2019, 37, (6), pp. 1205-1221.
205. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V., 'Federated Learning with Non-Iid Data', *arXiv preprint arXiv:1806.00582*, 2018.
206. Zhang, Y. and Yang, Q., 'A Survey on Multi-Task Learning', *arXiv preprint arXiv:1707.08114*, 2017.
207. Hertz, J.A., *Introduction to the Theory of Neural Computation*, (CRC Press, 2018)
208. Parisi, G.I., Tani, J., Weber, C., and Wermter, S., 'Lifelong Learning of Human Actions with Deep Neural Network Self-Organization', *Neural Networks*, 2017, 96, pp. 137-149.
209. Parisi, G.I., Tani, J., Weber, C., and Wermter, S., 'Lifelong Learning of Spatiotemporal Representations with Dual-Memory Recurrent Self-Organization', *Frontiers in neurorobotics*, 2018, 12, p. 78.
210. Rabinowitz, N.C., Desjardins, G., Rusu, A.-A., Kavukcuoglu, K., Hadsell, R.T., Pascanu, R., Kirkpatrick, J., and Soyer, H.J., Progressive Neural Networks', (Google Patents, 2017)
211. Li, Z. and Hoiem, D., 'Learning without Forgetting', *IEEE transactions on pattern analysis and machine intelligence*, 2017, 40, (12), pp. 2935-2947.
212. Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C., 'Measuring Catastrophic Forgetting in Neural Networks', in, *Proceedings of the AAAI Conference on Artificial Intelligence*, (2018)
213. Liu, X., Masana, M., Herranz, L., Van de Weijer, J., Lopez, A.M., and Bagdanov, A.D., 'Rotate Your Networks: Better Weight Consolidation and Less Catastrophic Forgetting', in, *2018 24th International Conference on Pattern Recognition (ICPR)*, (IEEE, 2018)
214. Ritter, H., Botev, A., and Barber, D., 'Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting', *arXiv preprint arXiv:1805.07810*, 2018.
215. Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T., 'Overcoming Catastrophic Forgetting by Incremental Moment Matching', *arXiv preprint arXiv:1703.08475*, 2017.
216. Zenke, F., Poole, B., and Ganguli, S., 'Continual Learning through Synaptic Intelligence', in, *International Conference on Machine Learning*, (PMLR, 2017)
217. Robins, A., 'Catastrophic Forgetting in Neural Networks: The Role of Rehearsal Mechanisms', in, *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, (IEEE, 1993)
218. Robins, A., 'Catastrophic Forgetting, Rehearsal and Pseudorehearsal', *Connection Science*, 1995, 7, (2), pp. 123-146.
219. Gepperth, A. and Karaoguz, C., 'A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems', *Cognitive Computation*, 2016, 8, (5), pp. 924-934.
220. Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C.H., 'Icarl: Incremental Classifier and Representation Learning', in, *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, (2017)
221. Ma, C., Konečný, J., Jaggi, M., Smith, V., Jordan, M.I., Richtárik, P., and Takáč, M., 'Distributed Optimization with Arbitrary Local Solvers', *Optimization Methods and Software*, 2017, 32, (4), pp. 813-848.
222. Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hofmann, T., and Jordan, M.I., 'Communication-Efficient Distributed Dual Coordinate Ascent', *Advances in neural information processing systems*, 2014, 27, pp. 3068-3076.
223. Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A.S., 'Federated Multi-Task Learning', in, *Advances in neural information processing systems*, (2017)
224. Kumar, S., Dutta, S., Chatturvedi, S., and Bhatia, M., 'Strategies for Enhancing Training and Privacy in Blockchain Enabled Federated Learning', in, *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, (IEEE, 2020)
225. Kopparapu, K. and Lin, E., 'Fedfmc: Sequential Efficient Federated Learning on Non-Iid Data', *arXiv preprint arXiv:2006.10937*, 2020.
226. Yao, X. and Sun, L., 'Continual Local Training for Better Initialization of Federated Models', in, *2020 IEEE International Conference on Image Processing (ICIP)*, (IEEE, 2020)
227. Gonzalez, C., Sakas, G., and Mukhopadhyay, A., 'What Is Wrong with Continual Learning in Medical Image Segmentation?', *arXiv preprint arXiv:2010.11008*, 2020.
228. Ling, C.X. and Bohn, T., 'A Conceptual Framework for Lifelong Learning', *arXiv preprint arXiv:1911.09704*, 2019.
229. Lomonaco, V., 'Continual Learning with Deep Architectures', 2019.
230. Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L., and Zeitak, I., 'Overcoming Forgetting in Federated Learning on Non-Iid Data', *arXiv preprint arXiv:1910.07796*, 2019.
231. Abbasi, M., Rajabi, A., Gagné, C., and Bobba, R.B., 'Towards Dependable Deep Convolutional Neural Networks (Cnns) with out-Distribution Learning', *arXiv preprint arXiv:1804.08794*, 2018.
232. https://www.cis.fordham.edu/wisdm/dataset.php, accessed Date Accessed
233. Saeed, A., Ozcelebi, T., and Lukkien, J., 'Multi-Task Self-Supervised Learning for Human Activity Detection', *Proceedings of*

the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2019, 3, (2), pp. 1-30.
234. Stokkink, Q., Epema, D., and Pouwelse, J., 'A Truly Self-Sovereign Identity System', *arXiv preprint arXiv:2007.00415*, 2020.
235. Stokkink, Q. and Pouwelse, J., 'Deployment of a Blockchain-Based Self-Sovereign Identity', in, *2018 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, (IEEE, 2018)
236. Pouwelse, J.A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D.H., Reinders, M., Van Steen, M.R., and Sips, H.J., 'Tribler: A Social-Based Peer-to-Peer System', *Concurrency and computation: Practice and experience*, 2008, 20, (2), pp. 127-138.
237. Zeilemaker, N., Capotă, M., Bakker, A., and Pouwelse, J., 'Tribler: P2p Media Search and Sharing', in, *Proceedings of the 19th ACM international conference on Multimedia*, (2011)
238. Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D., 'Randomized Gossip Algorithms', *IEEE transactions on information theory*, 2006, 52, (6), pp. 2508-2530.
239. Jin, P.H., Yuan, Q., Iandola, F., and Keutzer, K., 'How to Scale Distributed Deep Learning?', *arXiv preprint arXiv:1611.04581*, 2016.
240. Chang, H., Shejwalkar, V., Shokri, R., and Houmansadr, A., 'Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer', *arXiv preprint arXiv:1912.11279*, 2019.
241. Xie, C., Koyejo, O., and Gupta, I., 'Fall of Empires: Breaking Byzantine-Tolerant Sgd by Inner Product Manipulation', in, *Uncertainty in Artificial Intelligence*, (PMLR, 2020)
242. Cao, X. and Lai, L., 'Distributed Gradient Descent Algorithm Robust to an Arbitrary Number of Byzantine Attackers', *IEEE Transactions on Signal Processing*, 2019, 67, (22), pp. 5850-5864.
243. Alistarh, D., Allen-Zhu, Z., and Li, J., 'Byzantine Stochastic Gradient Descent', in, *Advances in neural information processing systems*, (2018)
244. Chen, C., Zhang, J., Tung, A.K., Kankanhalli, M., and Chen, G., 'Robust Federated Recommendation System', *arXiv preprint arXiv:2006.08259*, 2020.
245. Lv, S., Ye, J., Yin, S., Cheng, X., Feng, C., Liu, X., Li, R., Li, Z., Liu, Z., and Zhou, L., 'Unbalanced Private Set Intersection Cardinality Protocol with Low Communication Cost', *Future Generation Computer Systems*, 2020, 102, pp. 1054-1061.
246. De Cristofaro, E., Gasti, P., and Tsudik, G., 'Fast and Private Computation of Cardinality of Set Intersection and Union', in, *International Conference on Cryptology and Network Security*, (Springer, 2012)
247. Holzapfel, K., Karl, M., Lotz, L., Carle, G., Djeffal, C., Fruck, C., Haack, C., Heckmann, D., Kindt, P.H., and Köppl, M., 'Digital Contact Tracing Service: An Improved Decentralized Design for Privacy and Effectiveness', *arXiv preprint arXiv:2006.16960*, 2020.
248. Kales, D., Rechberger, C., Schneider, T., Senker, M., and Weinert, C., 'Mobile Private Contact Discovery at Scale', in, *28th {USENIX} Security Symposium ({USENIX} Security 19)*, (2019)
249. Pohlig, S. and Hellman, M., 'An Improved Algorithm for Computing Logarithms over Gf (P) and Its Cryptographic Significance (Corresp.)', *IEEE transactions on information theory*, 1978, 24, (1), pp. 106-110.
250. Shamir, A., Rivest, R.L., and Adleman, L.M., Mental Poker', *The Mathematical Gardner*, (Springer, 1981)