Internet Learning: Preliminary Steps Towards Highly Fault-Tolerant Learning on Device Networks

Surojit Ganguli [1] **Avi Amalanshu** [2] **Amritanshu Ranjan** [1] **David Inouye** [1]

Abstract

Distributed machine learning has grown in popularity due to data privacy, edge computing, and large model training. A subset of this class, Vertical Federated learning (VFL), aims to provide privacy guarantees in the scenario where each party shares the same sample space but only holds a subset of features. While VFL tackles key privacy challenges, it often assumes perfect hardware or communication (and may perform poorly under other conditions). This assumption hinders the broad deployment of VFL, particularly on edge devices, which may need to conserve power and may connect or disconnect at any time. To address this gap, we define the paradigm of *Internet Learning* (IL), which defines a context, of which VFL is a subset, and puts good performance under extreme dynamic condition of data entities as the primary goal. As IL represents a fundamentally different paradigm, it will likely require novel learning algorithms beyond end-to-end backpropagation, which requires careful synchronization across devices. In light of this, we provide some potential approaches for IL context and present preliminary analysis and experimental results on a toy problem.

1. Introduction and Motivation

The foundation of the Internet is the development of packet switching to replace circuit switching for communication (Baran, 1964; Davies, 1966). Indeed, Baran (1964) motivated packet switching almost entirely by the concept of *survivability* or *reliability* of the communication network under near catastrophic and adversarial faults (e.g., a nuclear bomb or enemy raid). The communication network would need to autonomously and seamlessly adapt to stations being destroyed (node removal) or being repaired/added (node addition) with (ideally) no noticeable communication loss from the user's perspective. Like circuit switching, current ML algorithms (particularly standard end-to-end backpropagation) require careful synchronization and would likely degrade significantly under benign failures. Yet, highly fault-tolerant learning algorithms could have wide applicability.

We highlight some real-world use cases where the data is distributed across clients but the system's ability to handle dynamic changes to the network of clients is critical (e.g., performance under near catastrophic faults).

Use-case 1: Precision Agriculture In recent times, modernization of agriculture has been aided by the adoption of sensor networks to aid with high crop yield (Thakur et al., 2019). Within individual farms, the sensors may become unreliable given harsh outdoor conditions. Thus, sensors may leave or join the network arbitrarily. System performance must be reliable even in such a scenario.

Use-Case 2: Smart Cities Cities are using edge devices to gather data to enhance quality of life. These devices form networks that enable real-time prediction of various aspects, e.g. traffic conditions (Sharma et al., 2021). However, there is concern that these devices may malfunction, negatively impacting traffic predictions. Therefore, it is important to ensure robust system performance even in the event of device failures.

In recent times, Federated Learning (FL) has emerged as a popular approach to distributed machine learning when data is distributed across clients. FL was primarily developed with the goal of data privacy and efficient communication (McMahan et al., 2017). FL has two standard approaches, horizontal and vertical, based on the method of partitioning of data among parties. The data context of Vertical FL (VFL) involves each party sharing the same set of samples while only possessing a portion of the features, whereas parties in standard horizontal FL possess identical features for unique samples (Wu et al., 2022). While there have been studies in the realm of horizontal FL on dealing with

[*]Equal contribution [1]Department of Electrical and Computer Engineering, Purdue University, West Lafeyette, USA [2]Electronics and Electrical Communication Engineering, IIT Kharagpur , Kharagpur, India. Correspondence to: Surojit Ganguli <sganguli@purdue.edu>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023.

arbitrary joining and dropping of parties (Ruan et al., 2021; Gu et al., 2021; Wang & Ji, 2022), no investigation into dynamic network changes exists for VFL to the best of the authors' knowledge.

To fill this gap, **we introduce the Internet Learning (IL) paradigm that aims to achieve strong performance even in the face of dynamic network conditions, particularly extreme changes, where the features of the data samples are distributed across clients**. Based on the decentralized and online setup envisioned for Internet Learning, we expect a majority of the data used will be unlabeled. Thus, we expect that the majority of learning under this paradigm will likely be unsupervised and self-supervised.

Owing to its widespread adoption and desirable characteristics such as data privacy, we suggest VFL and its variants be used as a baseline for comparing with IL. We also propose that performance in dynamic scenarios, including near catastrophic faults and nodes joining, be used as the primary metric for assessing algorithms developed for IL setup.

We summarize our contributions as follows:

- We define Internet Learning and compare it to similar learning paradigms.
- We review potential solutions via distributed backpropagation and localized learning methods.
- We implement preliminary baselines and evaluate network performance under ideal conditions and under device faults.

2. Internet Learning Problem Formulation

2.1. Internet Learning Definition

IL specifies both an operating context and the desired properties of a learning system. The context is comprised of two entities: data and network. We define the following terms to elucidate IL.

Notation Let $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ denote a dataset of with n samples and d features. Let $\boldsymbol{x}_{\mathcal{S}}$ denote the subvector associated with the indices in $\mathcal{S} \subseteq \{1, 2, \cdots, d\}$, e.g., if $\mathcal{S} = \{1, 5, 8\}$, then $\boldsymbol{x}_{\mathcal{S}} = [x_1, x_5, x_8]^T$. For C clients, the dataset at each client $c \in \{1, 2, \cdots, C\}$ will be denoted by $\mathcal{X}_c$. Let $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ denote a network (or graph) of clients, where $\mathcal{C} \subseteq \{1, 2, \cdots, C\}$ denotes the set of clients (or nodes) and $\mathcal{E}$ denotes the set of communication edges.

Definition 1 (Partial Features Data Context)**.** A partial features data context means that each client has access to a subset of the features, i.e.,

$$\mathcal{X}_c = \{\boldsymbol{x}_{i,\mathcal{S}_c}\}_{i=1}^n \tag{1}$$

where $\mathcal{S}_c \subseteq \{1, 2, \cdots, d\}$ for each client c.

Informally, the data context for IL is similar to that of VFL in the sense that the clients share the same set of samples but have access to different sets of features. One example of this VFL data context is the setup wherein data for the same patient is distributed across multiple hospitals and there is little to no overlap in the data among the hospitals. Another key example is a sensor network where each sample is based on timestamps, i.e., each sensor observes a partial part of the environment but the same time as other sensors.

Definition 2 (Dynamic Network Context)**.** A dynamic network means that the communication graph can change across time indexed by t, i.e., $\mathcal{G}(t) = (\mathcal{C}(t), \mathcal{E}(t))$, where the time dependency can be either deterministic or stochastic.

This dynamic network context includes many possible scenarios during both training and testing including:

1. *Clients leaving or joining the network* - The set of clients can change over time either due to clients failing, clients being repaired, new clients being added, or clients voluntarily leaving or joining. A node joining at time t' can be formalized as:

$$\mathcal{C}(t) = \begin{cases} \{1, \cdots, C-1\}, & t < t' \\ \{1, \cdots, C\}, & t \geq t' \end{cases}, \tag{2}$$

   where clients leaving is similar.

2. *Loss of Communication* - A communication link may be completely lost. This can be formalized as an communication edge between clients being removed at time t':

$$\mathcal{E}(t) = \begin{cases} \mathcal{E}, & t < t' \\ \mathcal{E} \setminus \{(c, c')\}, & t \geq t' \end{cases}, \tag{3}$$

   where $\mathcal{E}$ is the edges before the edge (c, c') is removed.

3. *Intermittent Communication* - A communication link may not always be reliable and data set may be lost. This can be formalized as the probability that a specific communication edge between clients c and c' is in the current edge set:

$$\Pr((c, c') \in \mathcal{E}(t)) = p\,, \tag{4}$$

   where p is the probability that the edge is included.

Note that these changes to the network could be caused by benign failures or voluntary changes to the network (e.g., weather or power limits) or they could be adversarially caused by bad actors to jam specific communications or destroy specific clients.

Given these context definitions, we now define the goal of internet learning in terms of a risk function we define next.

Definition 3 (Internet Learning Risk)**.** Given a dynamic network $\mathcal{G}(t)$, the *internet learning risk* of the all clients' parameters θ and a distributed inference algorithm $\Psi(\boldsymbol{x}; \theta, \mathcal{G}(t))$ is defined as the expected loss, denoted by ℓ, under a test distribution $p_{\text{test}}(\boldsymbol{x})$:

$$R_{\mathcal{G}(t)}(\theta, \Psi) \triangleq \mathbb{E}_{\boldsymbol{x} \sim p_{\text{test}}}[\ell(\Psi(\boldsymbol{x}; \theta, \mathcal{G}(t)))] \,. \tag{5}$$

Note that the parameters θ represent the concatenation of the parameters at every client. Each client could have different parameters and even different models. Some clients could even compute deterministic functions with no parameters though we will generally assume each client has a trainable model. Additionally, the inference function $\Psi(\boldsymbol{x}; \theta, \mathcal{G}(t))$ must compute the inference in a distributed manner under the dynamic conditions given by $\mathcal{G}(t)$. This is more complex than simply evaluating a model prediction at a single client because the models at every client may be different and the joint computation across the whole network may be used—this is why we use the terms "system" or "network" instead of "model" as all computation must be computed in a distributed manner. The distributed inference algorithm Ψ could also be trainable with parameters or be a fixed distributed algorithm. Given this risk definition, we can now fully define the internet learning problem.

Definition 4 (Internet Learning Problem)**.** Given partial features client datasets $\{\mathcal{X}_c\}_{c=1}^{C}$ (Definition 1) and a dynamic network $\mathcal{G}(t)$ (Definition 2), the *internet learning problem* aims to find the optimal client parameters θ and distributed inference algorithm Ψ that minimize the IL risk $R_{\mathcal{G}(t)}$ under the constraint that θ and Ψ are the outputs of some distributed learning algorithm $\Omega(\{\mathcal{X}_c\}_{c=1}^{C}, \mathcal{G}(t))$, i.e.,

$$\begin{aligned} \min_{\theta, \Psi} \quad & R_{\mathcal{G}(t)}(\theta, \Psi) \\ \text{s.t.} \quad & (\theta, \Psi) = \Omega(\{\mathcal{X}_c\}_{c=1}^{C}, \mathcal{G}'(t)) \end{aligned} \tag{6}$$

where $\mathcal{G}'(t)$ is the dynamic (possibly stochastic) network during training.

As with the motivation of the internet, we are particularly interested in good performance under extreme conditions such as catastrophic failures of many nodes or quickly changing client networks. While these extreme situations may not be encountered in practice, designing for the extreme situations will enable performance under less extreme situations. Like the significant change from circuit switching to packet switching, we expect that algorithms may have to be completely redesigned to work under these extreme adverse conditions that are not present in standard centralized learning paradigms.

We envision a successful strategy for IL will be to pursue durable representation learning, and share model parameters and representations among participants, relying on peer-to-peer communication to facilitate learning and inference.

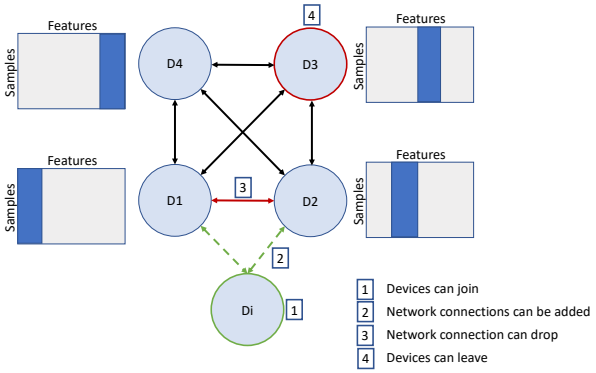


Figure 1. IL paradigm consists of a data and a network context. In the sample setup, there are 4 devices (labelled D1,D2,D3,D4) and each is networked to one another. The devices are free to communicate among themselves. Each device has the same sample space but only consist a portion of the feature space. The primary goal in the IL context is to enable graceful performance degradation in the event of dynamic changes, such as network connection getting added or dropped or devices joining or leaving

This will enable highly decentralized, large scale, and fault-tolerant distributed machine learning that is suited for the IL context.

2.2. Metrics for Internet Learning

In a distributed learning setting, there are various metrics that may be used for evaluating the learning strategy. For IL, we propose IL risk as the primary metric for comparing different approaches and algorithms. To provide a basis for comparison, we will compare the IL risk with various dynamic networks $\mathcal{G}(t)$ to the IL risk with an ideal and complete static network $G^*(t) = (\{1, \cdots, C\}, \{(c, c') : c, c' \in \{1, \cdots, C\}\})$, which does not depend on t and where all edges are included. The gap between these two IL risks is the key metric for understanding an IL system. Beyond IL risk, we will also consider secondary system metrics such as network latency, communication, and computation efficiency.

3. Preliminary IL Baseline: Distributed Backpropagation

We propose a simple framework for IL using backpropagation. While this may not be a practical solution, it is a natural naïve baseline. The network parameters are distributed amongst the devices in a "width-wise" manner, viz. the perceptrons (i.e., artificial neurons) in each layer are divided amongst participants. Input to a perceptron could be the output of another perceptron from the same device (via a standard feed-forward or residual connection) or from another device (via communication). All of these connections

are trained using backpropagation– gradients are communicated if need be.

The permitted communication is defined by two graphs: a "direct" and a "virtual" communication graph. The former describes which devices are connected directly to one another and the latter captures the indirect, multi-hop *virtual* connections between devices, where data must be relayed by intermediate to other devices across the direct communication graph. This is illustrated in the Appendix C. A complete virtual graph would allow any communication at each round, but would increase latency and communication cost. A virtual graph that matches the direct graph may significantly reduce latency and overall communication cost but will constrain architecture possibilities.

3.1. IL with Full Connectivity using Distributed Backpropagation (DB-IL-FC)

When the virtual graph is complete viz. all devices are allowed to communicate with each other, IL is compatible with any standard neural network architecture.

In simulations with no faults and no communication latency (or situations where the communication latency is similar to that of computation), the network behaves identically to a neural network with the same global architecture trained on a single device. However, their behavior under faults is different. Centrally training the network implies catastrophic failure if *the* device fails, whereas other participants in IL can continue learning when *a* device fails. Similarly, data-distributed learning, which generally requires a central aggregator or an elected leader, is contingent on the availability and sanctity of the server or leader.

3.2. Distributed Backpropagation with Partial Connectivity for IL (DB-IL-Part)

To reduce latency and communication costs, we consider a model where we force the virtual graph to match the direct graph so that every layer can be computed with only one communication round. When the virtual graph is incomplete and the device graph is not fully connected, the resulting neural network is sparsely connected. The analogous centralized neural network could have learned connections which are not permitted here due to lack of communication in the IL network.

Effectively, some weights of the centralized neural network are frozen at zero. Alternatively, the number of parameters in the network is limited, as weights between perceptrons on different devices are invalid. This harms the generalization ability given a neural network design, but reduces communication cost (under the assumption that not all pairs of devices on the network are directly connected).

4. Experiments

In our experiments, we simulate a network of four devices. We train our distributed backpropagation baselines for IL and show the effects of incomplete communication, random communication faults, and device faults. We first train and freeze an autoencoder, assumed to be shared across all devices. We then train an IL network on the latent representations produced by the encoder (see Section 4.1). Effectively, we use IL to train auxiliary deeper layers of the encoder.

4.1. Experimental setup

We simulate feature-parallel learning. Each device obtains different features for a given sample, then collaborates with the others to learn a joint representation. To simulate this, we first split MNIST images into a 2x2 square grid of patches. This splits the 28x28 image into 4 patches sized 14x14 each. Each patch is then passed as input to an encoder, which massively reduces the dimensionality of the patch to a single dimension (i.e., a 1x1 patch). At this point, there are 4 one-dimensional values—one to represent each patch of the original MNIST image. These 4 values are passed as input to the simple MLP network shown in Figure 2. Autoencoder architecture and implementation details are discussed in Appendix B.

The distributed MLP model contains 4 input nodes, one on each "device". We term the set of perceptrons of a single layer on a single device a *computational unit*. Here, each computational unit is one linear perceptron. Together, the device network resembles a neural network with 4 linear layers. The output of the final layer receives a residual connection from the input and is activated by a Leaky ReLU with a slope of 0.01 in the left half-plane. In the final step, the latent output from the IL Network is passed into a decoder which generates a reconstruction of the patch. We use reconstruction error as the loss function for this experiment.

We simulate the distribution of layers across multiple devices by using a separate PyTorch `nn.Linear` object for each computational unit. Partial connection is realized using PyTorch pre-forward hooks to drop inputs to each unit of a device from devices which are not virtually connected to it. The outputs from each device are concatenated to form the input features for the next "layer" (set of computational units).

With full connectivity, all devices can communicate with each other in a P2P manner. To study the effects of sparse connectivity, we employ a virtual graph that is equal to the direct communication graph. In our case, each device can only communicate with two "adjacent" devices, viz. there is ring topology. We also use this framework to simulate device and communication faults. Outputs from a device which has "failed" are masked with zeros. Similarly, gradi-

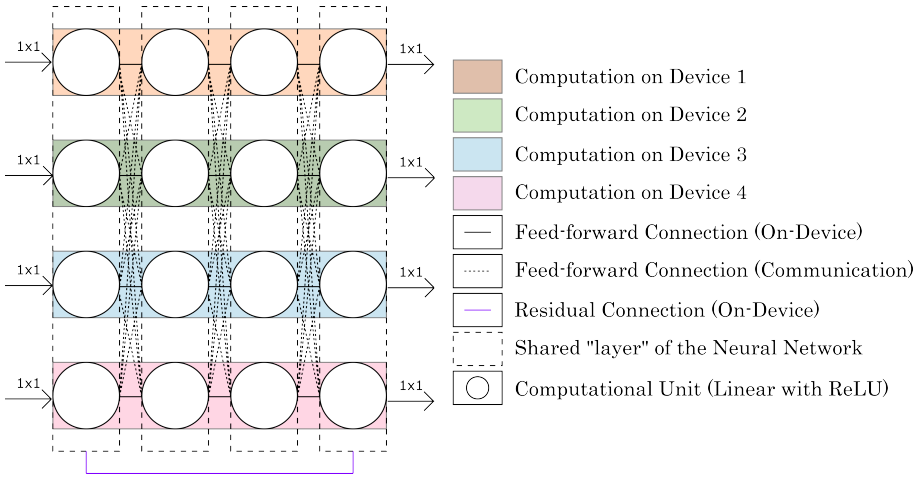


Figure 2. DB-IL-FC Network design used to simulate four devices connected to one another. The parameters of the network are split across the devices as shown.

ents (using backward hooks) and inputs to a device from a failed communication are masked with zeros. More elaborate details are discussed along with a quantitative analysis of the constraints on model architectures in Appendices C and D.

4.2. Results

A sample reconstruction for each experiment is presented in Figure 3, and average results are summarized in Table 1. With no communication faults during training, we observe a normalized MSE of 0.0540. Injecting random faults at a rate of 0.1 increases the error to 0.649. Injecting faults at rates 0.3 and 0.6 leads to a further increase of MSE to 0.1079 and 0.1161 respectively. We observe further degradation in performance when simulating device faults during inference at test time, i.e., when computing IL risk. Disabling one device during inference leads to an increase in MSE to 0.0650, 0.0756, 0.1320 and 0.1328 when the fault rates during training were 0.0, 0.1, 0.3 and 0.6 respectively. Training with sparse connectivity between devices also leads to a decrease in performance, with an MSE of 0.1100 when all devices are functioning during inference and 0.1317 with one faulty device.

5. Discussion and the Road to IL

The results of our experiments enforce the importance of communication in distributed learning and motivate a closer look at methods to improve fault tolerance. Our DB-IL baselines show a quantitative and qualitative decline in performance with an increase in faults, but DB-IL does not suffer from a single point of failure as would most server or leader-based distributed learning algorithms. We discuss future areas of work within IL below.

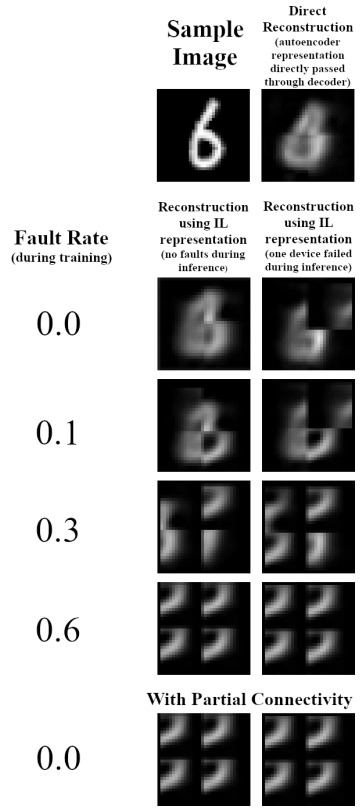


Figure 3. A sample of reconstructions from all our experiments. Qualitatively, the results for fault rate ≥ 0.3 are catastrophic. The advantage of splitting the network is also observable– even when one device fails, the network is able to produce a reconstruction. Similarly, the advantage of connectivity is demonstrated, since the result for partial connectivity is unacceptable. Further, for the case of inference with one failure, although the device responsible for the top right patch is dead, the decoder receives some contextual information due to connectivity from other devices.

Table 1. Results of our experiments. We observe that full communication between disjoint devices (DB-IL-FC) can be used to learn meaningful global representations, and that performance degrades with an increase in communication failure rate. Disabling a device during inference leads to decreased performance. The results highlight the significant impact faults have on network performance. Removing some connections during training altogether also increases the observed error (DB-IL-Part).

REGIME	FAULT RATE (*Training-time*)	INFERENCE ERROR (MSE) (*No device fault*)	(*With device fault*)
No IL net.	N/A	0.0551	N/A
DB-IL-FC	0.0	**0.0540**	0.0650
	0.1	0.0649	0.0756
	0.3	0.1079	0.1320
	0.6	0.1161	0.1328
DB-IL-Part	0.0	0.1100	0.1317

5.1. Relation to Vertical Federated Learning

Vertical Federated Learning (VFL) is a distributed learning regime for feature-split data. Similarly to IL, VFL is intended for systems where individual participants alone have incomplete information about the target. In VFL, participants first agree upon a sample to process. Each "guest" participant has a subset of features of the target sample. The guests use that data to train their own "lower" models and communicate their activations to a "host" participant, which owns labels and an "upper" model which aggregates activations and computes global losses. Based on the global loss, the host updates the upper model updates and communicates gradients to the lower model.

Like IL, VFL depends on communication to learn meaningful joint representations. Unlike IL, standard VFL algorithms only allow communication at one *cut* layer (between the upper and lower models). Further, VFL is generally a supervised paradigm, whereas IL is intended as a paradigm for unsupervised and self-supervised representation learning. Moreover, VFL depends on a central host client to perform aggregation over all local models. Like other such distributed learning algorithms, the aggregator acts as a single point of failure and an extra synchronization bottleneck. Thus, a failure of the aggregator creates a complete system failure. The IL paradigm is designed to be robust against *any* single client failures and even more extreme situations.

VFL is practically implemented in cross-silo scenarios, where there are a small number of participants (possibly within the same organization) with meaningfully intersecting sample spaces. That is to say, VFL participants must have a substantial number of samples in common, and they must also have a salient label feature. On the other hand, IL can be extended to large-scale networks of edge devices in the wild, since it is unsupervised and does not rely on aligning all participants by a specific target feature.

5.2. Potential alternative training algorithms

While distributed backpropagation is a natural naïve baseline for IL, local computation is preferable for IL, as backpropagation suffers from heavy computational overhead which is unlikely to be suitable for asynchronous or low-power applications. We review a variety of localized learning algorithms in the context of IL below.

Forward-forward Learning In distributed learning settings using modern hardware, backpropagation sometimes produces communication overhead, scalability limitations, or privacy concerns due to the heavy exchange or synchronization of model parameters. With faulty devices, this can lead to slower training time and poorer accuracy. These limitations can potentially be overcome by the Forward-Forward (FF) algorithm (Hinton, 2022). FF is a greedy learning procedure that replaces the forward and backward passes of backpropagation with two forward passes that operate on opposite data and objectives. Not only can FF efficiently train devices with power limitations because it can learn in a continuous stream without having to stop to compute gradients, but it also cultivates decentralization (and thereby privacy) because its localized layer-wise learning can be simulated in devices that individually learn towards a global objective. This localized learning approach is inherently fault-tolerant as the failure of one device should not significantly impact the learning process of other devices.

Hebbian-based Learning Journé et al. (2022); Moraitis et al. (2022) present SoftHebb, a biologically inspired training algorithm that implements a probabilistic understanding of local plasticity to learn unsupervised representations with local plasticity, which is a radically different approach to backpropagation which uses global supervisory signals. It addresses drawbacks such as non-bio plausibility, weight transport, non-local learning and update locking, which are associated with backpropgation. The authors have demonstrated that learning representations in a deep network using SoftHebb achieves accuracies on MNIST, CIFAR-10, STL-10 and ImageNet, respectively to 97.4%, 80.3%, 76.2% and 27.3%. Energy-based models using Hebbian-like contrastive divergence are also relevant (Høier et al., 2023; Detorakis et al., 2018; Movellan, 1991). Since all these methods update individual neural weights independently of each other, each virtual connection could communicate independently as opposed to synchronously with its layer.

Blockwise Learning Algorithms which train in a blockwise manner have recently seen renewed interest. Due to their local computation, they alleviate some of the "locking" synchronization constraints of end-to-end training with backpropagation and allow for a stronger form of parallelism and pipelining (Czarnecki et al., 2017). Belilovsky

et al. (2019; 2020) show that greedy layer-wise training scales to difficult datasets such as ImageNet, and that not freezing lower layers during training leads to performance comparable to backpropagation on shallow networks. The authors claim that each layer or module can be trained independently of those before it, using a replay buffer to store activations for the next layer. They use such "forward unlocking" to realize a fault-tolerant asynchronous model-distributed training regime, and show its ability to train and infer even with unstable workers. Such model parallelism could lead to an unprecedented level of decentralization if combined with width-wise feature parallelism based on IL's feature-split data context. Similarly, Siddiqui et al. (2023); Löwe et al. (2019); Xiong et al. (2020); Wang et al. (2021) present information theoretic objectives to achieve block-wise unsupervised representation learning without the use of auxiliary networks. Since these methods relax synchronization constraints and do not rely on supervision, they are candidates for an IL standard.

5.3. Security and Privacy

One of the reasons FL has gained prominence is due to the algorithms ability to ensure that clients or devices data remains private. Furthermore, recent research advances have also pushed the frontiers of making FL secure and be able to successfully resist threats and attacks. In a similar manner, there can be further research in making IL secure and privacy preserving.

Provided the distributed nature and the ability of the devices/clients in IL to interact with each other, we are aware that this setup is vulnerable to threats of attack and information leakage. Thus, moving forward, we believe that research in areas of security and privacy for IL will be a critical topic and inspiration can be derived from related research areas in FL.

5.4. Better evaluation metrics

It is also important to test IL on more meaningful problems. The distributed backpropagation algorithm could be used to train a classifier instead of an autoencoder, for example. This would also allow information-efficient unsupervised classification algorithms such as Forward-Forward.

Further, it is necessary to investigate training objectives which take better advantage of communication. In our example, strong and stable connectivity is critical to performance. However, even with no faults, the reconstructions from the split images are not qualitatively impressive, perhaps since the full linear connections do not provide any semantic structure with their context. Moreover, they are bottlenecked by the lack of communication in the encoder training.

5.5. Partial Timing Protocol

Our experiments were simulated with no communication latency. However, communication latency is a salient practical consideration for IL. Since there is no aggregation in IL-based representation learning, synchronization requirements are not strict. However, being able to handle communication and device failures requires the ability to distinguish between missed communication and late communication. Therefore, it is practically necessary to implement a partial synchronization protocol. Perhaps a "best-effort" strategy could be employed, where a crash is assumed to have occurred if there is no communication within a timeout period or number of repeated polls. Simulating experiments with finitesimal communication latency is possible within our framework. Combined with a semi-synchronous protocol, this would allow for a study of the effect of variable and increasing latency on model performance under IL.

5.6. Fault Tolerance Strategy

By not having a centralized aggregation procedure or any other inherent synchronization requirements, our IL baseline does not suffer from a single-point of failure. We observe that performance instead degrades gradually as the fault rate is increased. However, the degradation is eventually catastrophic.

A fault tolerance strategy could lead to a more graceful degradation in performance. The direct reliance on communication for model execution implies a passive strategy via packet switching is possible, e.g. one based on communication only with neighbors as opposed to a global filter. IL on a vast network with such a strategy would be highly durable without requiring heavy computation or extra synchronization. We present a study of fault tolerance strategies for existing distributed learning algorithms in the Appendix A.

6. Conclusion

In this paper, we presented some ideas towards Internet Learning. IL systems are intended to achieve a high level of fault tolerance and survivability without synchronization or aggregation overhead. Through our experiments, we showed that distributing the parameters of a neural network width-wise across participants is one feasible approach that serves as a simple baseline.

Localized learning algorithms are a promising tool that may improve the fault tolerance of IL by relaxing some synchronization bottlenecks. The absence of backpropagation would completely eliminate a communication requirement and therefore a failure point. Another interesting direction to investigate is extending the activation replay buffer system of (Belilovsky et al., 2020) to account for feature parallelism. This may achieve a higher level of fault tolerance and also

decentralization by jointly implementing both feature and model parallelism.

IL would enable local networks of sensing clients to asynchronously collaborate and learn meaningful representations without needing to communicate with a central server or bear the cost of broadcasting updates to each other. Further, it would enable a world-wide web of learning. Instead of solely passing data between clients, participants in this web would process it to learn robust representations.

The decentralized nature of IL combined with model and feature parallelism could be used to leverage large scale, low power edge computation for deep learning. Therein, sensing devices can process their collected data and forward their output to intermediate clients training deeper layers. Similarly, an edge device may extract features for inference by communicating with any other client. Since each client is only responsible for a small subset of parameters of the global model, participants do not take on substantial computational workloads.

We believe IL may not only enable a much larger scale of collaboration in distributed ML, but by practically achieving a high level of durability and efficiency, also extend ML applications to systems in highly decentralized and dynamic environments where they are currently infeasible.

References

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a07c2f3b3b907aaf8436a26c6d77f0a2-Paper.pdf.

Baran, P. *On Distributed Communications: I. Introduction to Distributed Communications Networks*. RAND Corporation, Santa Monica, CA, 1964. doi: 10.7249/RM3420.

Barrak, A., Petrillo, F., and Jaafar, F. Architecting Peer-to-Peer Serverless Distributed Machine Learning Training for Improved Fault Tolerance. *arXiv e-prints*, art. arXiv:2302.13995, February 2023. doi: 10.48550/arXiv.2302.13995.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pp. 583–593. PMLR, 2019.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pp. 736–745. PMLR, 2020.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.

Bouhata, D., Moumen, H., Mazari, J. A., and Bounceur, A. Byzantine fault tolerance in distributed machine learning : a survey, 2022.

Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D. DRACO: Byzantine-resilient distributed training via redundant gradients. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 903–912. PMLR, 10–15 Jul 2018a. URL https://proceedings.mlr.press/v80/chen18l.html.

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)*, pp. 1178–1187, 2018b. doi: 10.1109/BigData.2018.8622598.

Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1 (2):1–25, December 2017. doi: 10.1145/3154503. URL https://doi.org/10.1145/3154503.

Chen, Y., Yang, Q., He, S., Shi, Z., and Chen, J. Ftpipehd: A fault-tolerant pipeline-parallel distributed training framework for heterogeneous edge devices, 2021.

Czarnecki, W. M., Świrszcz, G., Jaderberg, M., Osindero, S., Vinyals, O., and Kavukcuoglu, K. Understanding synthetic gradients and decoupled neural interfaces. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 904–912. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/czarnecki17a.html.

Damaskinos, G., El Mhamdi, E. M., Guerraoui, R., Patra, R., and Taziki, M. Asynchronous Byzantine machine learning (the case of SGD). In Dy, J. and Krause, A. (eds.),

Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pp. 1145–1154. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/damaskinos18a.html.

Data, D., Song, L., and Diggavi, S. Data encoding methods for byzantine-resilient distributed optimization. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 2719–2723, 2019. doi: 10.1109/ISIT.2019.8849857.

Davies, D. W. Proposal for a digital communication network. *Unpublished memo*, 1966.

Detorakis, G., Bartley, T., and Neftci, E. Contrastive hebbian learning with random feedback weights. *CoRR*, abs/1806.07406, 2018. URL http://arxiv.org/abs/1806.07406.

El Mhamdi, E. M., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in Byzantium. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3521–3530. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/mhamdi18a.html.

El-Mhamdi, E.-M., Guerraoui, R., Guirguis, A., Hoang, L. N., and Rouault, S. Genuinely distributed byzantine machine learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. ACM, July 2020. doi: 10.1145/3382734.3405695. URL https://doi.org/10.1145/3382734.3405695.

Fan, X., Ma, Y., Dai, Z., Jing, W., Tan, C., and Low, B. K. H. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in Neural Information Processing Systems*, 34:1007–1021, 2021.

Fawzi, H., Tabuada, P., and Diggavi, S. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic Control*, 59(6): 1454–1467, 2014. doi: 10.1109/TAC.2014.2303233.

Gu, X., Huang, K., Zhang, J., and Huang, L. Fast federated learning in the presence of arbitrary device unavailability. *Advances in Neural Information Processing Systems*, 34: 12052–12064, 2021.

Guo, S., Zhang, T., Yu, H., Xie, X., Ma, L., Xiang, T., and Liu, Y. Byzantine-resilient decentralized stochastic gradient descent, 2021.

Gupta, N. and Vaidya, N. H. Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 415–420, 2019a. doi: 10.1109/ALLERTON.2019.8919735.

Gupta, N. and Vaidya, N. H. Randomized reactive redundancy for byzantine fault-tolerance in parallelized learning, 2019b.

Gupta, N. and Vaidya, N. H. Fault-tolerance in distributed optimization: The case of redundancy. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. ACM, July 2020a. doi: 10.1145/3382734.3405748. URL https://doi.org/10.1145/3382734.3405748.

Gupta, N. and Vaidya, N. H. Fault-tolerance in distributed optimization: The case of redundancy. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pp. 365–374, 2020b.

Gupta, N., Doan, T. T., and Vaidya, N. H. Byzantine fault-tolerance in decentralized optimization under 2f-redundancy. In *2021 American Control Conference (ACC)*, pp. 3632–3637, 2021a. doi: 10.23919/ACC50511.2021.9483067.

Gupta, N., Liu, S., and Vaidya, N. Byzantine fault-tolerant distributed machine learning with norm-based comparative gradient elimination. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 175–181, 2021b. doi: 10.1109/DSN-W52860.2021.00037.

He, L., Karimireddy, S. P., and Jaggi, M. Secure byzantine-robust machine learning, 2021. URL https://openreview.net/forum?id=69EFStdgTD2.

Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Høier, R., Staudt, D., and Zach, C. Dual propagation: Accelerating contrastive hebbian learning with dyadic neurons, 2023.

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)*, pp. 1–6, 2019. doi: 10.1109/ICC.2019.8761674.

Jin, R., Huang, Y., He, X., Dai, H., and Wu, T. Stochastic-sign SGD for federated learning with theoretical guarantees. *CoRR*, abs/2002.10940, 2020. URL https://arxiv.org/abs/2002.10940.

Journé, A., Rodriguez, H. G., Guo, Q., and Moraitis, T. Hebbian deep learning without feedback. *arXiv preprint arXiv:2209.11883*, 2022.

Li, L., Xu, W., Chen, T., Giannakis, G. B., and Ling, Q. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1544–1551, July 2019. doi: 10.1609/aaai.v33i01.33011544. URL https://doi.org/10.1609/aaai.v33i01.33011544.

Liu, S. A survey on fault-tolerance in distributed optimization and machine learning, 2021.

Liu, S., Gupta, N., and Vaidya, N. H. Approximate byzantine fault-tolerance in distributed optimization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pp. 379–389, 2021.

Löwe, S., O'Connor, P., and Veeling, B. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems*, 32, 2019.

Lugan, S., Desbordes, P., Brion, E., Ramos Tormo, L. X., Legay, A., and Macq, B. Secure architectures implementing trusted coalitions for blockchained distributed learning (tclearn). *IEEE Access*, 7:181789–181799, 2019. doi: 10.1109/ACCESS.2019.2959220.

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*, pp. 1273–1282. PMLR, 2017.

Moraitis, T., Toichkin, D., Journé, A., Chua, Y., and Guo, Q. Softhebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks. *Neuromorphic Computing and Engineering*, 2(4):044017, 2022.

Movellan, J. R. Contrastive hebbian learning in the continuous hopfield model. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E. (eds.), *Connectionist Models*, pp. 10–17. Morgan Kaufmann, 1991. ISBN 978-1-4832-1448-1. doi: https://doi.org/10.1016/B978-1-4832-1448-1.50007-X. URL https://www.sciencedirect.com/science/article/pii/B978148321448150007X.

Rabbat, M. and Nowak, R. Distributed optimization in sensor networks. In *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, pp. 20–27, 2004. doi: 10.1145/984622.984626.

Rajput, S., Wang, H., Charles, Z., and Papailiopoulos, D. Detox: A redundancy-based framework for faster and more robust gradient aggregation. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/415185ea244ea2b2bedeb0449b926802-Paper.pdf.

Rathore, S., Pan, Y., and Park, J. H. Blockdeepnet: A blockchain-based secure deep learning for iot network. *Sustainability*, 2019.

Ruan, Y., Zhang, X., Liang, S.-C., and Joe-Wong, C. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3403–3411. PMLR, 2021.

Sharma, H., Haque, A., and Blaabjerg, F. Machine learning in wireless sensor networks for smart cities: a survey. *Electronics*, 10(9):1012, 2021.

Siddiqui, S. A., Krueger, D., LeCun, Y., and Deny, S. Blockwise self-supervised learning at scale, 2023.

Su, L. and Xu, J. Securing distributed gradient descent in high dimensional statistical learning, 2019.

Thakur, D., Kumar, Y., Kumar, A., and Singh, P. K. Applicability of wireless sensor networks in precision agriculture: A review. *Wireless Personal Communications*, 107:471–512, 2019.

Tuli, S., Casale, G., and Jennings, N. R. Dragon: Decentralized fault tolerance in edge federations. *IEEE Transactions on Network and Service Management*, 20 (1):276–291, 2023. doi: 10.1109/TNSM.2022.3199886.

Wang, S. and Ji, M. A unified analysis of federated learning with arbitrary client participation. *Advances in Neural Information Processing Systems*, 35:19124–19137, 2022.

Wang, Y., Ni, Z., Song, S., Yang, L., and Huang, G. Revisiting locally supervised learning: an alternative to end-to-end training. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=fAbkE6ant2.

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*, 18(5):2438–2455, 2021. doi: 10.1109/TDSC.2019.2952332.

Wu, Z., Li, Q., and He, B. Practical vertical federated learning with unsupervised representation learning. *IEEE Transactions on Big Data*, 2022.

Xia, Q., Tao, Z., Hao, Z., and Li, Q. Faba: An algorithm for fast aggregation against byzantine attacks in distributed neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 4824–4830. International Joint

Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/670. URL https://doi.org/10.24963/ijcai.2019/670.

Xie, C., Koyejo, O., and Gupta, I. Generalized byzantine-tolerant sgd, 2018a.

Xie, C., Koyejo, O., and Gupta, I. Phocas: dimensional byzantine-resilient stochastic gradient descent. *CoRR*, abs/1805.09682, 2018b. URL http://arxiv.org/abs/1805.09682.

Xie, C., Koyejo, S., and Gupta, I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6893–6901. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/xie19b.html.

Xie, C., Koyejo, S., and Gupta, I. Zeno++: Robust fully asynchronous SGD. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10495–10503. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/xie20c.html.

Xiong, Y., Ren, M., and Urtasun, R. Loco: Local contrastive representation learning, 2020.

Yang, H., Zhang, X., Fang, M., and Liu, J. Byzantine-resilient stochastic gradient descent for distributed learning: A lipschitz-inspired coordinate-wise median approach, 2019.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5650–5659. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/yin18a.html.

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, 2021.

Ángel Morell, J. and Alba, E. Dynamic and adaptive fault-tolerant asynchronous federated learning using volunteer edge devices. *Future Generation Computer Systems*, 133:53–67, 2022. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2022.02.024. URL https://www.sciencedirect.com/science/article/pii/S0167739X22000735.

A. Fault Tolerance in Distributed Learning

There is a growing body of work on fault tolerance in distributed learning. Methods to mitigate the effects of Byzantine clients have seen particular interest. Most state-of-the-art methods involve gradient filters. Recently, techniques based on error correcting codes and blockchain have also been proposed. We summarize some broad strategies here. (Liu et al., 2021; Liu, 2021; Bouhata et al., 2022; Gupta & Vaidya, 2020b) present detailed reviews.

Gradient Filters Methods such as (Barrak et al., 2023; He et al., 2021; Fan et al., 2021; Gupta et al., 2021a; Gupta & Vaidya, 2020a; Xie et al., 2019; Alistarh et al., 2018) employ gradient (or model) filters. The goal therein is to aggregate n gradients or local models assuming up to f may be from Byzantine agents. (Liu et al., 2021) show that when the system is $2f$-redundant viz.

$$\arg\min_x \sum_{i\in\hat{S}} Q_i(x) = \arg\min_x \sum_{i\in S} Q_i(x)$$

(where S is any subset of the n distributed agents s.t. $|S| = n - f$, $\hat{S} \subseteq S$ s.t. $|\hat{S}| \geq n - 2f$, and Q_i is the local objective of the i^{th} agent), it is possible to calculate the optimal x of $n - f$ good agents without knowing which f are faulty. The authors relax this condition to define $f - \epsilon$ resilience (effectively, the sets of state x converging to within ϵ under f faults) and $(2f, \epsilon)$ redundancy (effectively, relaxing exact equality in $2f$-redundancy). The authors use these to provide convergence guarantees for Comparative Gradient Elimination (CGE) (Gupta et al., 2021b) and Coordinate-wise Trimmed Mean (CwTM) (Yin et al., 2018).

Other coordinate-wise methods include (El-Mhamdi et al., 2020; Yang et al., 2019; Xie et al., 2018b; Chen et al., 2017) and MarMed and MeaMed from (Xie et al., 2018a). Therein, instead of generating an aggregate based on global characteristics, each index of an incoming gradient is compared to the corresponding index of all others. The f farthest outliers for every coordinate are suppressed. Each coordinate of the final gradient is therefore a coordinate-wise aggregation.

On the other hand, methods such as (Guo et al., 2021; Jin et al., 2020; Li et al., 2019; Xia et al., 2019) involve dampening or dropping entire gradients that are expected to be outliers. Usually, this is based on a geometric metric such as norm (Gupta et al., 2021b; Gupta & Vaidya, 2019a; El Mhamdi et al., 2018) or distance from mean/median such as GeoMed from (Xie et al., 2018a). It may also be based on statistical features such as Lipschitz characteristics.

Gradient Codes There is also some recent work based on error-correcting codes (Data et al., 2019; Rajput et al., 2019; Gupta & Vaidya, 2019b; Chen et al., 2018a). (Chen et al., 2018a) present an aggregation technique based on coding theory, using redundancy as a repetition code. (Gupta & Vaidya, 2019b) add a reactive redundancy mechanism to identify and suppress Byzantine workers. Being linear codes, the compuatational overhead is lower than that of many gradient filters. Further, they are compatible with non-convex losses, which is not generally true for gradient filters.

Blockchain Methods (Weng et al., 2021; Zhao et al., 2021; Rathore et al., 2019; Lugan et al., 2019; Chen et al., 2018b) propose methods using blockchain technology. Various validation mechanisms exist. These are used to add aggregated models as new blocks. Consensus-based validation enables peer-to-peer computation, and mining rewards discourage adversarial faults.

State Filters The problem of state estimation is related to distributed machine learning (being a case of distributed optimization, if the state is linear), and moreso to IL (being a case of partial observability). The objective is to construct an estimate of the state of a system given incomplete observations from a network of sensors. Distributed gradient descent is one way to achieve this. As such, gradient filters are compatible (Rabbat & Nowak, 2004). Some works instead use state filters. (Fawzi et al., 2014) propose an algorithm based on the Kalmann filter to detect adversarial attacks and produce a robust state estimate. Similar to distributed learning, the partial states must show a form of $2f$-redundancy called $2k$-sparse observability.

Crash Tolerance Tolerance to Byzantine faults encompass a wide range of scenarios, from benign communication faults, to adversarial attacks. However, this vast literature does not extend to meta-system faults e.g. communication delay and device failure. There is some work that proposes techniques to handle such faults. Most methods are adaptive algorithms (Ángel Morell & Alba, 2022; He et al., 2021) or based on heartbeat monitoring and checkpointing (Ángel Morell & Alba, 2022; Chen et al., 2021; Tuli et al., 2023).

Drawbacks, and towards an IL solution The fundamental drawback of these methods is their computational overhead. All gradient filters and codes cited are $O(n^2d)$ per round (or worse), with only some also being $O(nd)$ where n is the number of agents and d is model parameter size (Bouhata et al., 2022; Liu, 2021). The methods based on blockchain are substantially costlier in time complexity and energy.

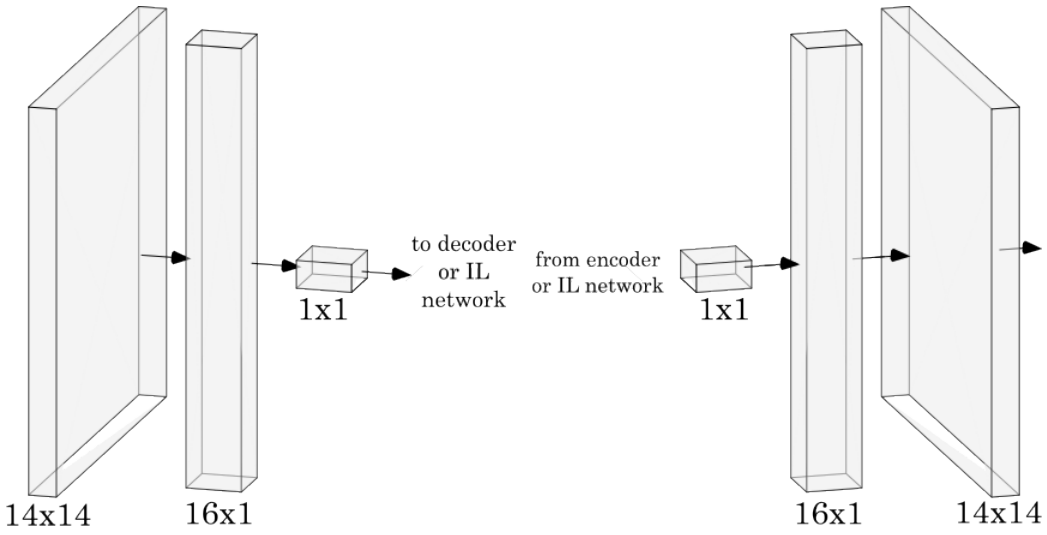
Further, the convergence of many aggregators are based on strong assumptions, for example on convexity (Liu et al., 2021; Gupta & Vaidya, 2019a; Xie et al., 2018b; Blanchard et al., 2017) or i.i.d. data (Xie et al., 2020; 2019; Yang et al., 2019; Su & Xu, 2019) or a combination of the two. These are not necessarily practical assumptions. Many deep models utilize non-convex loss functions, for which some gradient filters may fail to converge altogether. Similarly, while the assumption of i.i.d. data is not unfounded, it is impractical in federated and decentralized scenarios, and it is a threat to data privacy. Some methods (Rajput et al., 2019; Data et al., 2019; Chen et al., 2017) rely on the assumption that all clients will independently learn near-identical models, which is really only feasible if there is data redundancy imposed by a central allocator. Even when all conditions are met, there is a tradeoff between tightness of convergence and rate of convergence.

It is also difficult to implement aggregation in a decentralized environment. Since all gradients must be available to any aggregating participant, there is a high communication overhead. Further, either there must be elected leaders to perform this aggregation (unreliable, since the leaders may be faulty) or all devices should perform aggregation by consensus (further communication cost). Furthermore, maintaining a copy of the model and communicating updates is not feasible for low power edge devices or large foundation models with billions or even trillions of parameters.

Mechanisms such as consensus and aggregation require synchronization between clients. In a server-client setting, the server must wait for all gradients for the current round to arrive. In a P2P setting, clients must wait for communication from all other devices. Asynchronous methods exist (Xie et al., 2020; Jin et al., 2019; Damaskinos et al., 2018) and are based on other statistical properties. However, this is at the expense of even stronger requirements for convergence.

IL, by virtue of splitting the global model itself amongst participants, does not require global aggregation. Instead, identifying adversarial or failed local neighbors should be sufficient to implement a packet switching-like strategy. This would make fault tolerance inherent to the learning process, and mitigate constraints such as synchronization and computational overhead.

B. Autoencoder Architecture



(a) Encoder Architecture (b) Decoder Architecture

Figure 4. The parameters of this encoder-decoder architecture are shared across all devices. We use IL to process the 1×1 latent representations generated by the encoder in our experiments.

The encoder (Figure 4a) receives an input patch of 14x14, viz. the partial input observed by one device. The input is passed to a fully connected 16×1 linear layer activated by ReLU. This is followed by another fully connected linear layer, which further compresses the dimension to 1.

The decoder (Figure 4b) increases the data dimensionality symmetrically to the encoder. The input is passed to a fully connected linear layer with ReLU activation, increasing the dimensionality to 16. Another fully connected linear layer increases the data dimension to 14×14. This network is trained on all patches for 10 epochs using an Adam optimizer with an initial learning rate of 0.1 and a decay of $\gamma = 0.1$ every 3 epochs.

The encoder-decoder model is trained to reconstruct the original image using an MSE loss between the patch and the 14×14 network output. This model is frozen and identically available on all devices. We then use IL with the goal of learning a better encoded representation. We generate new latent representations using the frozen encoder and pass it to an IL network. To evaluate performance, we pass the resultant IL-trained latent representation through the decoder and calculate reconstruction loss, averaged over all four patches.

C. Device and Virtual Graphs

We run our experiments under the assumption of no communication latency. The virtual graph (Figure 5b) describes all possible network connections between devices, including those via intermediate devices. A multiple-hop connection with zero latency is no different from a direct connection with zero latency. Therefore, in our scenario the virtual graph alone is sufficient to describe the network behavior. However, our implementation allows for future simulations with communication delay. In that case, the virtual graph edges are weighed based on the device graph. The virtual graph should be constructed judiciously, since varying communication path lengths might desynchronize and slow down training. This would also make the system less tolerant to communication faults, since intermediate connections may fail before the virtual communication reaches its destination. Since the device graph is universally known, a simple graph traversal algorithm (say, Dijkstra's) on the device graph (Figure 5a) can be used to determine the least-hops path for a virtual connection before training begins. For example (assuming all direct connections are equivalent in latency and neglecting switching latency), in Figure 5, the latency between device 2 and 3 is thrice of that between device 4 and 3. But, they are allowed to communicate gradients and activations to each other.

In the case of incomplete connectivity, we use a ring structure (Figure 6). We do not use any virtual-only connections (Figure 6b), viz. only devices that are adjacent per the device graph can communicate with each other. For example, a pair of fully connected layers in a neural network would have a 4×4 weight parameter, where a similar layer in our IL example with full connectivity has four 4×1 weight parameters (see next para.). Our example with ring connectivity has four 3×1 parameters, since there is no connection with the non-adjacent device. It is obvious the generalization ability will be hampered, as the degree of freedom is lower. In the Figure 6, device 2 cannot send activations and gradients to perceptrons on device 4, for instance.

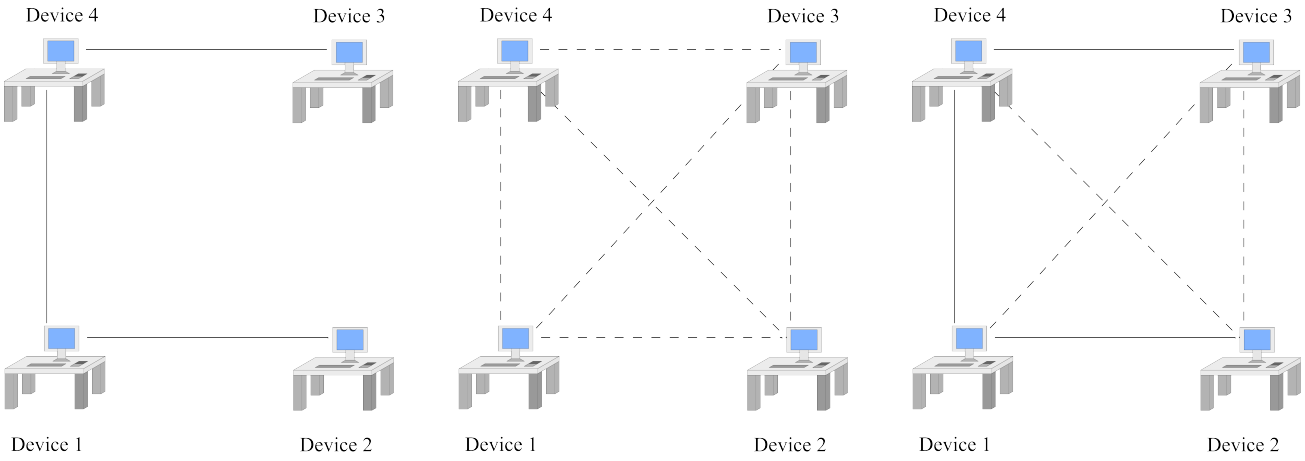
Both the device and the virtual graph are implemented in PyTorch using custom "distributed" layers, which are a list of linear layers corresponding to the "computational units" on each device. A distributed layer accepts a full input vector, and uses the virtual graph to pass the relevant inputs to each computational unit. This is achieved using a PyTorch forward hook which drops indices from the input before passing it to a computational unit. The resulting activations are then concatenated to form the feature vector for the next layer.

D. Fault Injection

We use PyTorch hooks to simulate faults on forward passes and backward passes. We simulate two kinds of faults:

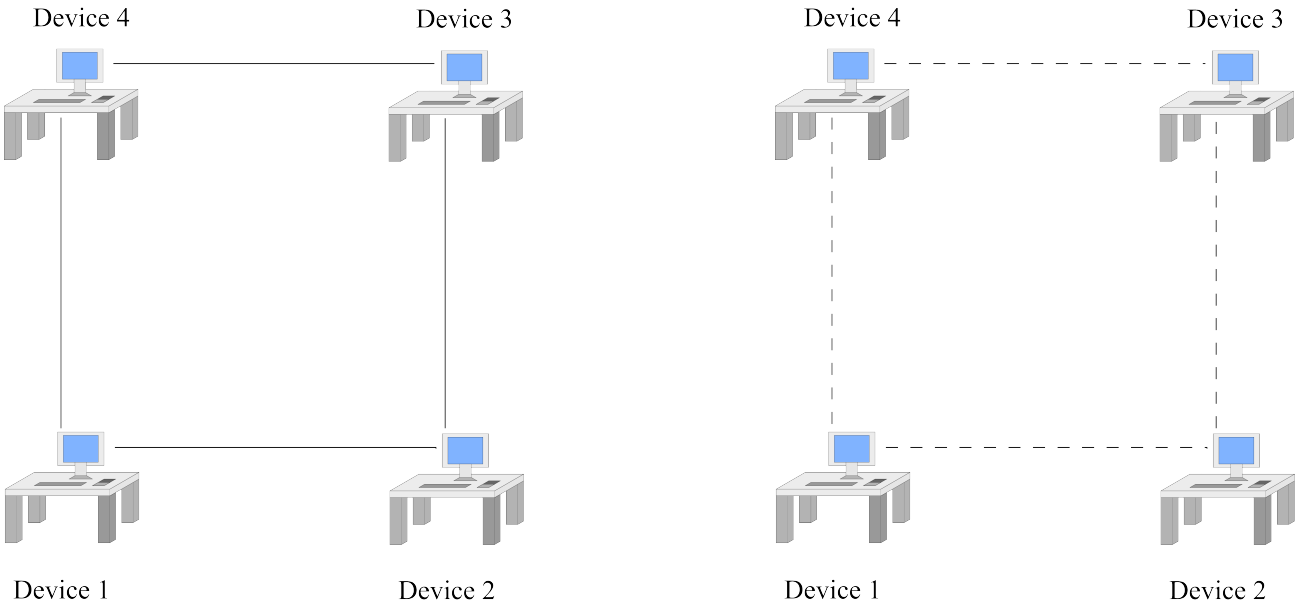
1. **Device Faults:** For the period of a device fault, that device does not communicate with any other devices and stops training its on-device computational units. We implement this by dropping inputs to and outputs from computational units on that device, effectively setting them to zero.
2. **Random Communication Faults:** On any call of `forward()` or `backward()`, each inter-device connection may fail by some fixed probability. When this occurs, the respective features and gradients are masked to zero.

We test both types of faults in our experiments, and note a considerable degradation in performance. This emphasizes the importance of communication in our paradigm and the need for a fault tolerance strategy. It is also important to note that the network we test IL with is very shallow, with only 48 parameters in the fully-connected case. A single device fault could invalidate 25% of all parameters. This is similarly the case for incomplete communication– as mentioned in section D of the appendix, removing only two virtual links reduced our parameter count to 36.



(a) Device Graph (b) Virtual Graph (c) Combined Graph

Figure 5. Communication graphs for our experiments with full connectivity. Solid lines between two devices indicate they are directly connected. Dashed lines indicate they are virtually connected, viz. either directly connected or connected via other devices.



(a) Device Graph (b) Virtual Graph

Figure 6. Communication graphs for our experiments with partial connectivity. Devices 2 and 4 cannot communicate with each other. Device 3 and 1 also cannot communicate with each other.

For the case of random faults, the number of faults in any round is effectively the result of a binomial experiment with $n = \binom{4}{2} = 6$ trials with failure probability $p =$ failure `rate`. With `rate` $= 0.6$, the probability that at least two connections will fail (viz. no more parameters are active than the ring-connected case) $\mathbb{P}(f \geq 2) \approx 0.96$. For `rate` $= 0.3$, $\mathbb{P}(f \geq 2) \approx 0.58$ and for `rate` $= 0.1$ it is ≈ 0.11. This is in line with the performance degradation we observe, and that which we see under incomplete communication and device failure. On the other hand, if there were instead $n = 8$ devices viz. 192 parameters, the binomial experiment has $n = \binom{8}{2} = 28$. Since suppressing one inter-device connection effectively kills $L - 1$ parameters (where $L = 4$ is the number of layers), we need $f = 16$ connection faults to remove $\frac{1}{4}$ths of the parameters. $\mathbb{P}(f \geq 16) \approx 0.69$ for `rate` $= 0.6$ and 0 for `rate` $= 0.1$ and 0.3. Indeed, a single fault will affect a smaller fraction of global model parameters as the number of parameters increases. It is clear that with larger-scale connectivity, fault tolerance will inherently improve.