The Non-IID Data Quagmire of Decentralized Machine Learning

Kevin Hsieh [1 2] Amar Phanishayee [1] Onur Mutlu [3 2] Phillip B. Gibbons [2]

Abstract

Many large-scale machine learning (ML) applications need to perform *decentralized* learning over datasets generated at different devices and locations. Such datasets pose a significant challenge to decentralized learning because their different contexts result in significant data distribution skew across devices/locations. In this paper, we take a step toward better understanding this challenge by presenting a detailed experimental study of decentralized DNN training on a common type of data skew: skewed distribution of data labels across devices/locations. Our study shows that: (i) skewed data labels are a fundamental and pervasive problem for decentralized learning, causing significant accuracy loss across many ML applications, DNN models, training datasets, and decentralized learning algorithms; (ii) the problem is particularly challenging for DNN models with batch normalization; and (iii) the degree of data skew is a key determinant of the difficulty of the problem. Based on these findings, we present SkewScout, a system-level approach that adapts the communication frequency of decentralized learning algorithms to the (skew-induced) accuracy loss between data partitions. We also show that group normalization can recover much of the accuracy loss of batch normalization.

1. Introduction

The advancement of machine learning (ML) is heavily dependent on the processing of massive amounts of data. The most timely and relevant data are often generated at different devices all over the world, e.g., data collected by mobile phones and video cameras. Because of communication and privacy constraints, gathering all such data for centralized processing can be impractical/infeasible. For example, moving raw data across national borders is subject to data sovereignty law constraints (e.g., GDPR (European Parliament, 2016)). Similar constraints apply to centralizing private data from phones. These constraints motivate the need for ML training over widely distributed data (***decentralized learning***). For example, *geo-distributed learning* (Hsieh et al., 2017) trains a global ML model over data spread across geo-distributed data centers. Similarly, *federated learning* (McMahan et al., 2017) trains a centralized model over data from a large number of mobile devices. Federated learning has been an important topic both in academia (140+ papers in 2019) and industry (500+ million installations on Android devices).

Key Challenges in Decentralized Learning. There are two key challenges in decentralized learning. First, training a model over decentralized data using traditional training approaches (i.e., those designed for centralized data, often using a bulk synchronous parallel (BSP) approach (Valiant, 1990)) requires massive amounts of communication. Doing so drastically slows down the training process because the communication is bottlenecked by the limited wide-area or mobile network bandwidth (Hsieh et al., 2017; McMahan et al., 2017). Second, decentralized data is typically generated at different contexts, which can lead to significant differences in the *distribution* of data across data partitions. For example, facial images collected by cameras would reflect the demographics of each camera's location, and images of kangaroos can be collected only from cameras in Australia or zoos. Unfortunately, existing decentralized learning algorithms (e.g., (Hsieh et al., 2017; McMahan et al., 2017; Smith et al., 2017; Lin et al., 2018; Tang et al., 2018)) mostly focus on reducing communication, as they either (i) assume the data partitions are independent and identically distributed (IID) or (ii) conduct only very limited studies on non-IID data partitions. This leaves a key question mostly unanswered: *What happens to ML applications and decentralized learning algorithms when their data partitions are not IID?*

Our Goal and Key Findings. We aim to take a step to further the understanding of the above key question. In this work, we focus on a common type of non-IID data, widely used in prior work (e.g., (McMahan et al., 2017; Tang et al., 2018; Zhao et al., 2018)): skewed distribution of data labels across devices/locations. Such *skewed label partitions* arise frequently in the real world (see §2.2 for examples). Our study covers various DNN applications, DNNs, training datasets, decentralized learning algorithms, and degrees of label skew. Our study reveals three key findings:

[1]Microsoft Research [2]Carnegie Mellon University [3]ETH Zürich. Correspondence to: Kevin Hsieh <kevin.hsieh@microsoft.com> and Phillip B. Gibbons <gibbons@cs.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

- Training over skewed label partitions is a fundamental and pervasive problem for decentralized learning. Three decentralized learning algorithms (Hsieh et al., 2017; McMahan et al., 2017; Lin et al., 2018) suffer from major model quality loss when run to convergence on skewed label partitions, across the applications, models, and training datasets in our study.
- DNNs with *batch normalization* (Ioffe & Szegedy, 2015) are particularly vulnerable to skewed label partitions, suffering significant model quality loss even under BSP, the most communication-heavy approach.
- The degree of skew is a key determinant of the difficulty level of the problem.

These findings reveal that non-IID data is an important yet heavily understudied challenge in decentralized learning, worthy of extensive study. To facilitate further study on skewed label partitions, we release a real-world, geo-tagged dataset of common mammals on Flickr (Flickr), which is openly available at https://doi.org/10.5281/zenodo.3676081 (§2.2).

Solutions. As two initial steps towards addressing the vast challenge of non-IID data, we first show that among the many proposed alternatives to batch normalization, *group normalization* (Wu & He, 2018) avoids the skew-induced accuracy loss of batch normalization under BSP. With this fix, all models in our study achieve high accuracy on skewed label partitions under (communication-heavy) BSP, and the problem can be viewed as a trade-off between accuracy and the amount of communication. Intuitively, there is a tug-of-war among different data partitions, with each partition pulling the model to reflect its data, and only close communication, tuned to the skew-induced accuracy loss, can save the overall model accuracy of the algorithms in our study. Accordingly, we present SkewScout, which periodically sends local models to remote data partitions and compares the model performance (e.g., validation accuracy) between local and remote partitions. Based on the accuracy loss, SkewScout adjusts the amount of communication among data partitions by controlling how *relaxed* the decentralized learning algorithms should be, such as controlling the threshold that determines which parameters are worthy of communication. Thus, SkewScout can seamlessly integrate with decentralized learning algorithms that provide such communication control. Our experimental results show that SkewScout's adaptive approach automatically reduces communication by 9.6× (under high skew) to 34.1× (under mild skew) while retaining the accuracy of BSP.

Contributions. We make the following contributions. First, we conduct a detailed empirical study on the problem of skewed label partitions. We show that this problem is a fundamental and pervasive challenge for decentralized learning. Second, we build and release a large real-world dataset to facilitate future study on this challenge. Third, we make a new observation showing that this challenge is particularly problematic for DNNs with batch normalization, even under BSP. We discuss the root cause of this problem and we find that it can be addressed by using an alternative normalization technique. Fourth, we show that the difficulty level of this problem varies with the data skew. Finally, we design and evaluate SkewScout, a system-level approach that adapts the communication frequency among data partitions to reflect the skewness in the data, seeking to maximize communication savings while preserving model accuracy.

2. Background and Motivation

We first provide background on popular decentralized learning algorithms (§2.1). We then highlight a real-world example of skewed label partitions: geographical distribution of mammal pictures on Flickr, among other examples (§2.2).

2.1. Decentralized Learning

In a decentralized learning setting, we aim to train an ML model w based on all the training data samples (x_i, y_i) that are generated and stored in one of the K partitions (denoted as P_k). The goal of the training is to fit w to all data samples. Typically, most decentralized learning algorithms assume the data samples are independent and identically distributed (IID) among different P_k, and we refer to such a setting as the *IID setting*. Conversely, we call it the *Non-IID setting* if such an assumption does not hold.

We evaluate three popular decentralized learning algorithms to see how they perform on different applications over the IID and Non-IID settings, using skewed label partitions. These algorithms can be used with a variety of stochastic gradient descent (SGD) (Robbins & Monro, 1951) approaches, and aim to reduce communication, either among data partitions (P_k) or between the data partitions and a centralized server.

- `Gaia` (Hsieh et al., 2017), a geo-distributed learning algorithm that dynamically eliminates insignificant communication among data partitions. Each partition P_k accumulates updates Δw_j to each model weight w_j locally, and communicates Δw_j to all other data partitions only when its relative magnitude exceeds a predefined threshold (Algorithm 1 in Appendix A[1]).
- `FederatedAveraging` (McMahan et al., 2017), a popular algorithm for federated learning that combines local SGD on each client with model averaging. The algorithm selects a subset of the partitions P_k in each epoch, runs a pre-specified number of local SGD steps on each selected P_k, and communicates the resulting models back to a centralized server. The server averages all these models and uses the averaged w as the starting

[1] All Appendices are in the supplemental material.

point for the next epoch. (Algorithm 2 in Appendix A).

- `DeepGradientCompression` (Lin et al., 2018), a popular algorithm that communicates only a prespecified amount of gradients each training step, with various techniques to retain model quality such as momentum correction, gradient clipping (Pascanu et al., 2013), momentum factor masking, and warm-up training (Goyal et al., 2017) (Algorithm 3 in Appendix A).

In addition to these decentralized learning algorithms, we show the results of using BSP (Valiant, 1990) over the IID and Non-IID settings. BSP is significantly slower than the above algorithms because it does not seek to reduce communication: all updates from each P_k are shared among all data partitions after each training step. As noted earlier, for decentralized learning, there is a natural tension between the amount of communication and the quality of the resulting model. Different data distributions among the P_k pull the model in different directions—more communication helps mitigate this "tug-of-war" so that the model well-represents all the data. Thus, BSP, with its full communication at every step, is used to establish a quality target for trained models.

2.2. Real-World Examples of Skewed Label Partitions

Non-IID data among devices/locations encompass many different forms. There can be skewed distribution of features (probability $\mathcal{P}(x)$), labels (probability $\mathcal{P}(y)$), or the relationship between features and labels (e.g., varying $\mathcal{P}(y|x)$ or $\mathcal{P}(x|y)$) among devices/locations (Kairouz et al., 2019) (see more discussion in Appendix K). In this work, we focus on skewed distribution of labels ($\mathcal{P}_{P_i}(y) \not\sim \mathcal{P}_{P_j}(y)$ for different data partitions P_i and P_j), which is also the setting considered by most prior work in this domain (e.g., (McMahan et al., 2017; Tang et al., 2018; Zhao et al., 2018)).

Skewed distribution of labels is common whenever data are generated from heterogeneous users or locations. For example, pedestrians and bicycles are more common in street cameras than in highway cameras (Luo et al., 2019). In facial recognition tasks, most individuals appear in only a few locations around the world. Certain types of clothing (mittens, cowboy boots, kimonos, etc.) are nearly non-existent in many parts of the world. Similarly, certain mammals (e.g., kangaroos) are far more likely to show up in certain locations (Australia). In the rest of this section, we highlight this phenomenon with a study of the geographical distribution of mammal pictures on Flickr (Flickr).

Dataset Creation. We start with the 48 classes in the *mammal* subcategory of the 600 most common classes for bounding boxes in Open Images V4 (Kuznetsova et al., 2018). For each class label, we use Flickr's API to search for relevant pictures. Due to noise in Flickr search results (e.g., "jaguar" returns the mammal or the car), we clean the data with a state-of-the-art DNN, PNAS (Liu et al., 2018), which is pre-trained on ImageNet. As we can only clean classes that exist in both Open Images and ImageNet, we end up with 41 mammal classes and 736,005 total pictures. We call the resulting dataset the *Flickr-Mammal* dataset (see Appendix B for more details).

Geographical Analysis. We map each Flickr picture's geotag to its corresponding geographic regions based on the M49 Standard (United Nation Statistics Division, 2019). As we are mostly interested in the distribution of labels ($\mathcal{P}(y)$) among different regions, we normalize the number of samples across region (non-normalized results are similar (Appendix B)). Table 1 illustrates the top-5 classes among first-level regions (continents) and their normalized share of samples in the world.

Region	Top 1	Top 2	Top 3	Top 4	Top 5
Africa	zebra (72%)	antelope (71%)	lion (68%)	cheetah (62%)	hippopotamus (59%)
Americas	mule (84%)	skunk (82%)	armadillo (73%)	harbor seal (65%)	squirrel (61%)
Asia	panda (64%)	hamster (59%)	monkey (58%)	camel (51%)	red panda (42%)
Europe	lynx (72%)	hedgehog (56%)	sheep (56%)	deer (43%)	otter (43%)
Oceania	kangaroo (92%)	koala (92%)	whale (44%)	sea lion (34%)	alpaca (32%)

Table 1. Top-5 mammals in each continent and their share of samples worldwide (e.g., 72% of zebra images are from Africa).

Skewed distribution of labels is a natural phenomenon. As Table 1 shows, the top-5 mammals in each continent constitute 32%–92% of the normalized sample share in the world (compared to 20% if the distribution were IID). As expected, the top mammals in each region reflect their population share in the world (e.g., kangaroos/koalas in Oceania and zebras/antelopes in Africa). Furthermore, there is *no overlap* for the top-5 classes among different continents, which suggests drastically different label distributions ($\mathcal{P}(y)$) among continents. We observe a similar phenomenon when the analysis is done based on second-level geographical regions (Appendix B). Our observations show that in a decentralized learning setting, where such images would be collected and stored in their native regions, the distribution of labels across partitions would be highly skewed.

3. Experimental Setup

Our study consists of three dimensions: *(i)* ML applications/models, *(ii)* decentralized learning algorithms, and *(iii)* degree of data skew. We explore all three dimensions with rigorous experimental methodologies. In particular, we make sure the accuracy of our trained ML models on IID data matches the reported accuracy in corresponding papers. All source code and settings are available at `https://github.com/kevinhsieh/non_iid_dml`.

Applications. We evaluate different deep learning applications, DNN model structures, and training datasets:

- IMAGE CLASSIFICATION with four DNN models: AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015), LeNet (LeCun et al., 1998), and ResNet (He et al., 2016). We use two datasets, CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015). We use the default validation set of each of the two datasets to quantify the validation accuracy as our model quality metric. We use popular datasets in order to compare model accuracy with existing work, and we also report results with our *Flickr-Mammal* dataset.
- FACE RECOGNITION with the center-loss face model (Wen et al., 2016) over the CASIA-WebFace (Yi et al., 2014) dataset. We use verification accuracy on the LFW dataset (Huang et al., 2007) as our model quality metric.

For all applications, we tune the training parameters (e.g., learning rate, minibatch size, number of epochs, etc.) such that the baseline model (BSP in the IID setting) achieves the model quality of the corresponding original paper. We then use these training parameters in all other settings. We further ensure that training/validation accuracy has stopped improving by the end of all our experiments. Appendix C lists all major training parameters in our study.

Non-IID Data Partitions. In addition to studying *Flickr-Mammal*, we create non-IID data partitions by partitioning datasets using the data labels, i.e., using image classes for image classification and person identities for face recognition. We control the *skewness* by controlling the fraction of data that are non-IID. For example, 20% non-IID indicates 20% of the dataset is partitioned by labels, while the remaining 80% is partitioned uniformly at random. §4 and §5 focus on the 100% non-IID setting in which the entire dataset is partitioned using labels, while §6 studies the effect of varying the skewness. As our goal is to train a global model, the model is tested on the entire validation set.

Hyper-Parameters Selection. The algorithms we study provide the following hyper-parameters (see Appendix A for details of these algorithms) to control the amount of communication (and hence the training time):

- `Gaia` uses T_0, the initial threshold to determine if an update (Δw_j) is significant. The significance threshold decreases whenever the learning rate decreases.
- `FederatedAveraging` uses $Iter_{Local}$ to control the number of local SGD steps on each selected P_k.
- `DeepGradientCompression` uses s to control the sparsity of updates (update magnitudes in top s percentile are exchanged). Following the original paper (Lin et al., 2018), s follows a warm-up schedule: 75%, 93.75%, 98.4375%, 99.6%, 99.9%. We use a hyper-parameter E_{warm}, the number of epochs for each warm-up sparsity, to control the duration of the warm-up. For example, if $E_{warm} = 4$, s is 75% in epochs 1–4, 93.75% in epochs 5–8, and so on.

We select a hyper-parameter θ of each decentralized learning algorithm (T_0, $Iter_{Local}$, E_{warm}) so that *(i)* θ achieves the same model quality as BSP in the IID setting and *(ii)* θ achieves similar communication savings across the three decentralized learning algorithms. We study the sensitivity of our findings to the choice of θ in §4.4.

4. Non-IID Study: Results Overview

This paper seeks to answer the question of what happens to ML applications, ML models, and decentralized learning algorithms when their data label partitions are not IID. In this section, we provide an overview of our findings, showing that skewed label partitions cause *major model quality loss*, across many applications, models, and algorithms.

4.1. Image Classification

We first present the model quality with different decentralized learning algorithms in the IID and Non-IID settings for IMAGE CLASSIFICATION using the CIFAR-10 dataset. We use five partitions ($K=5$) in this evaluation, and we discuss results with more partitions in Appendix F. As the CIFAR-10 dataset consists of ten object classes, each data partition has two object classes in the Non-IID setting. Figure 1 shows the results with four popular DNNs (AlexNet, GoogLeNet, LeNet, and ResNet). (Convergence curves for AlexNet and ResNet are shown in Appendix D.) According to the hyper-parameter criteria in §3, we select $T_0 = 10\%$ for `Gaia`, $Iter_{Local} = 20$ for `FederatedAveraging`, and $E_{warm} = 8$ for `DeepGradientCompression`. We make two major observations.

1) Non-IID data is a pervasive problem. *All* three decentralized learning algorithms lose significant model quality for *all* four DNNs in the Non-IID setting. We see that while these algorithms retain the validation accuracy of BSP in the *IID setting* with 15×–20× communication savings (agreeing with the results from the original papers for these algorithms), they lose 3% to 74% validation accuracy in the *Non-IID setting*. Simply running these algorithms for more epochs would not help because the training/validation accuracy has already stopped improving. Furthermore, the training completely diverges in some cases, such as `DeepGradientCompression` with GoogLeNet and ResNet20 (`DeepGradientCompression` with ResNet20 also diverges in the IID setting). The pervasiveness of the problem is quite surprising, as we have a diverse set of decentralized learning algorithms and DNNs. This result shows that Non-IID data is a pervasive and challenging problem for decentralized learning, and this problem has been heavily understudied. §4.3 discusses potential causes of this problem.

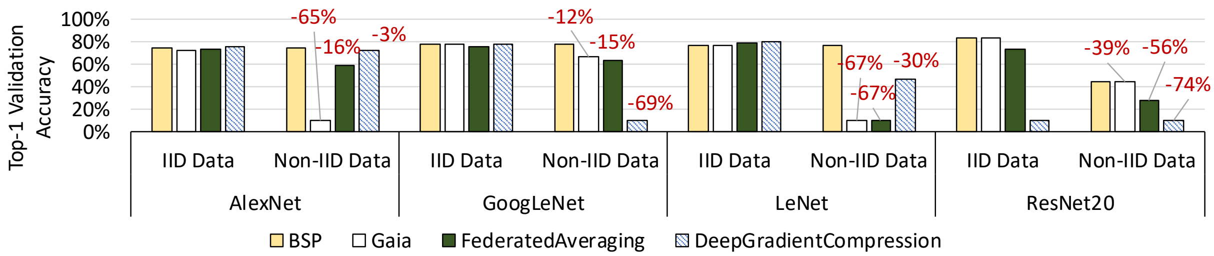


Figure 1. Top-1 validation accuracy for IMAGE CLASSIFICATION over the CIFAR-10 dataset. A "-x%" label above a bar indicates the accuracy loss relative to BSP in the IID setting.

2) Even BSP cannot completely solve this problem. We see that even BSP, with its full communication at every step, cannot retain model quality for some DNNs in the Non-IID setting. The validation accuracy of ResNet20 in the Non-IID setting is 39% lower than that in the IID setting. This finding suggests that, for some DNNs, it *may not be possible* to solve the Non-IID data challenge by increasing communication between data partitions. We find that this problem exists not only in ResNet20, but also in all other DNNs we study with batch normalization (ResNet10, BN-LeNet (Ioffe & Szegedy, 2015) and Inception-v3 (Szegedy et al., 2016)). We discuss this problem and potential solutions in §5.

The same trend in a larger dataset. We conduct a similar study using the ImageNet dataset (Russakovsky et al., 2015) (1,000 image classes). We observe the same problems in the ImageNet dataset (e.g., an 8.1% to 61.7% accuracy loss on ResNet10), whose number of classes is two orders of magnitude more than the CIFAR-10 dataset. Appendix E discusses the experiment in detail.

The same problem in real-world datasets. We run similar experiments on our Flickr-Mammal dataset. We use five partitions (K=5) in this experiment, one for each continent, where each partition has as its local training data precisely the images from its corresponding continent. Thus, we capture the real-world non-IID setting present in Flickr-Mammal. For comparison, we also consider an artificial IID setting, in which all the Flickr-Mammal images are randomly distributed among the five partitions. Figure 2 shows the results. We use GoogLeNet in this experiment, and we select $T_0 = 10\%$ for Gaia and $Iter_{Local} = 20$ for FederatedAveraging based on the criteria in §3. We observe the same problems for decentralized learning algorithms on this real-world dataset. Specifically, Gaia and FederatedAveraging are able to retain the model quality in the (artificial) IID setting, but they lose 3.7% and 3.2% accuracy in the (real-world) Non-IID setting, respectively. The loss is smaller compared to Figure 1 in part because most data labels still exist in all data partitions in the (real-world) Non-IID setting, which makes the problem easier than the 100% non-IID setting. This loss arises even with modest hyper-parameter settings, and is expected to be larger with settings that more greatly reduce communication. We also show that the loss increases to 5.2% and 5.5%, respectively, when Flickr-Mammal is partitioned at the subcontinent level (Appendix F). This is significant as the result suggests that skewed labels arising in real-world settings are a major problem for decentralized learning.

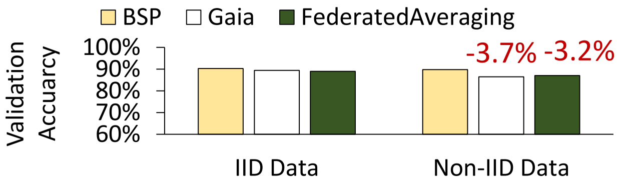


Figure 2. GoogLeNet's Top-1 validation accuracy for IMAGE CLASSIFICATION over the Flickr-Mammal dataset, where 5% data are randomly selected as the validation set. Non-IID Data is based on real-world data distribution among continents, and IID Data is the artificial setting in which images are randomly assigned to partitions. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting. Note: The y-axis starts at 60% accuracy.

4.2. Face Recognition

We further examine another popular ML application, FACE RECOGNITION, to see if the Non-IID data problem is a challenge across different applications. We use two partitions in this evaluation. According to the hyper-parameter criteria in §3, we select T_0=20% for Gaia and $Iter_{Local}$=50 for FederatedAveraging. It is worth noting that the verification process of FACE RECOGNITION is fundamentally different from IMAGE CLASSIFICATION, as FACE RECOGNITION does *not* use the classification layer (and thus the training labels) at all in the verification process. Instead, for each pair of verification images, the DNN uses the distance between the feature vectors of these images to determine whether the two images are of the same person.

The same problem in different applications. Figure 3 shows the LFW verification accuracy. Again, the same problem happens: the decentralized learning algorithms work

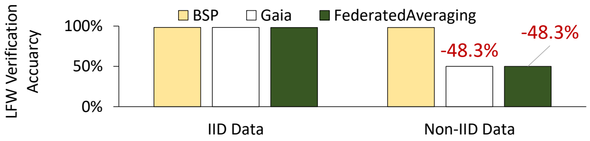


Figure 3. LFW verification accuracy for FACE RECOGNITION. Labels show the accuracy loss relative to BSP in the IID setting.

well in the IID setting, but they lose significant accuracy in the Non-IID setting. In fact, both `Gaia` and `FederatedAveraging` cannot converge to a useful model in the Non-IID setting: their 50% accuracy is no better than random guessing for the binary questions. This result is interesting because the labels of the validation dataset are completely different from the labels of the training dataset, but the validation accuracy is still severely impacted by the non-IID data label partitions in the training set.

4.3. The Problem of Decentralized Algorithms

The above results show that three diverse decentralized learning algorithms all experience drastic accuracy losses in the Non-IID setting. We find two reasons for the accuracy loss. First, for algorithms such as `Gaia` that save communication by allowing small model differences in each partition P_k, the Non-IID setting results in *completely different models among* P_k. The small differences give local models room for specializing to local data. Second, for algorithms that save communication by synchronizing sparsely (e.g., `FederatedAveraging` and `DeepGradientCompression`), each P_k generates more diverged gradients in the Non-IID setting, which is not surprising as each P_k sees vastly different training data. When they are finally synchronized, they may have diverged so much from the global model that they push the global model the wrong direction. See Appendix G for further details.

4.4. Algorithm Hyper-Parameters

We also study the sensitivity of the Non-IID problem to hyper-parameter choice among decentralized learning algorithms. We find that even relatively conservative hyper-parameter settings, which incur high communication costs, still suffer major accuracy loss in the Non-IID setting. In the IID setting, on the other hand, the *same* hyper-parameter achieves similar high accuracy as BSP. In other words, the the Non-IID problem is not specific to particular hyper-parameter choices. Appendix H shows supporting results.

5. Batch Normalization: Problem and Solution

5.1. Batch Normalization in the Non-IID Setting

How BatchNorm works. Batch normalization (BatchNorm) (Ioffe & Szegedy, 2015) is one of the most popular mechanisms in deep learning (20,000+ citations as of August 2020). BatchNorm aims to stabilize a DNN by normalizing the input distribution to zero mean and unit variance. Because the *global* mean and variance are unattainable with stochastic training, BatchNorm uses *minibatch mean and variance* as an estimate of the global mean and variance. Specifically, for each minibatch $\mathcal{B}$, BatchNorm calculates the minibatch mean $\mu_\mathcal{B}$ and variance $\sigma_\mathcal{B}$, and then uses $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ to normalize each input in $\mathcal{B}$. BatchNorm enables faster and more stable training because it enables larger learning rates (Bjorck et al., 2018; Santurkar et al., 2018).

BatchNorm and the Non-IID setting. While BatchNorm is effective in practice, its dependence on minibatch mean and variance ($\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$) is known to be problematic in certain settings. This is because BatchNorm uses $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ for training, but it typically uses an estimated global mean and variance (μ and σ) for validation. If there is a major mismatch between these means and variances, the validation accuracy is going to be low. This can happen if the minibatch size is small or the sampling of minibatches is not IID (Ioffe, 2017). The Non-IID setting in our study exacerbates this problem because each data partition P_k sees very different training samples. Hence, the $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ in each partition can vary significantly across the partitions, and the synchronized global model may not work for *any* set of data. Worse still, we cannot simply increase the minibatch size or do better minibatch sampling to solve this problem, because in the Non-IID setting the underlying dataset in each P_k does not represent the global dataset.

We validate if there is indeed major divergence in $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ among different P_k in the Non-IID setting. We calculate the divergence of $\mu_\mathcal{B}$ as the difference between $\mu_\mathcal{B}$ in different P_k over the average $\mu_\mathcal{B}$ (i.e., it is $\frac{||\mu_{\mathcal{B},P_0} - \mu_{\mathcal{B},P_1}||}{||AVG(\mu_{\mathcal{B},P_0},\, \mu_{\mathcal{B},P_1})||}$ for two partitions P_0 and P_1). We use the average $\mu_\mathcal{B}$ over every 100 minibatches in each P_k so that we get better estimation. Figure 4 depicts the divergence of $\mu_\mathcal{B}$ for each channel of the first layer of BN-LeNet, which is constructed by inserting BatchNorm to LeNet after each convolutional layer. As we see, the divergence of $\mu_\mathcal{B}$ is significantly larger in the Non-IID setting (between 6% to 61%) than in the IID setting (between 1% to 5%). We also observe the same trend in minibatch variances $\sigma_\mathcal{B}$ (not shown). As this problem has nothing to do with the amount of communication among P_k, it explains why even BSP cannot retain model accuracy for BatchNorm in the Non-IID setting.

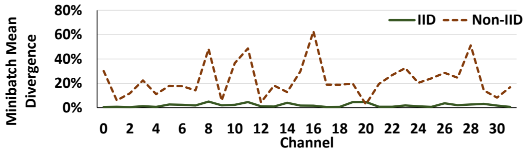


Figure 4. Minibatch mean divergence for the first layer of BN-LeNet over CIFAR-10 using two P_k.

5.2. Alternatives to Batch Normalization

As the problem of BatchNorm in the Non-IID setting is due to its dependence on minibatches, the natural solution is to replace BatchNorm with alternative normalization mechanisms that are *not* dependent on minibatches. Unfortunately, most existing alternative normalization mechanisms (Weight Normalization (Salimans & Kingma, 2016), Layer Normalization (Ba et al., 2016), Batch Renormalization (Ioffe, 2017)) have their own drawbacks (see Appendix I). Here, we discuss a particular mechanism that may be used instead.

Group Normalization. Group Normalization (GroupNorm) (Wu & He, 2018) is an alternative normalization mechanism that aims to overcome the shortcomings of BatchNorm and Layer Normalization (LayerNorm). GroupNorm divides adjacent channels into groups of a prespecified size $\mathcal{G}_{size}$, and computes the per-group mean and variance for *each input sample*. Hence, GroupNorm does not depend on minibatches for normalization (the shortcoming of BatchNorm), and GroupNorm does not assume all channels make equal contributions (the shortcoming of LayerNorm).

We evaluate GroupNorm with BN-LeNet over CIFAR-10. We carefully select $\mathcal{G}_{size} = 2$, which works best with this DNN. Figure 5 shows the Top-1 validation accuracy with GroupNorm and BatchNorm across decentralized learning algorithms. We make two major observations.

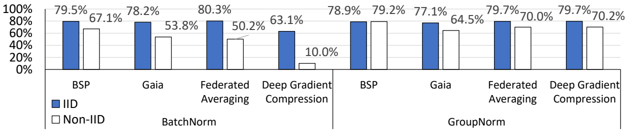


Figure 5. Top-1 validation accuracy with BatchNorm and GroupNorm for BN-LeNet over CIFAR-10 with 5 partitions.

First, GroupNorm successfully recovers the accuracy loss of BatchNorm with BSP in the Non-IID setting. As the figure shows, GroupNorm with BSP achieves 79.2% validation accuracy in the Non-IID setting, which is as good as the accuracy in the IID setting. This shows GroupNorm can be used as an alternative to BatchNorm to overcome the Non-IID data challenge for BSP. Second, GroupNorm dramatically helps the decentralized learning algorithms in the Non-IID setting as well. With GroupNorm, there is 14.4%, 8.9% and 8.7% accuracy loss for `Gaia`, `FederatedAveraging` and `DeepGradientCompression`, respectively. While the accuracy losses are still significant, they are better than their BatchNorm counterparts by an additive 10.7%, 19.8% and 60.2%, respectively.

Discussion. While our study shows that GroupNorm can be a good alternative to BatchNorm in the Non-IID setting, it is worth noting that BatchNorm is widely adopted in many DNNs. Hence, more study is needed to see if GroupNorm can replace BatchNorm for different applications and DNN models. As for other tasks such as recurrent (e.g., LSTM (Hochreiter & Schmidhuber, 1997)) and generative (e.g., GAN (Goodfellow et al., 2014)) models, other normalization techniques such as LayerNorm can be good options because *(i)* they are shown to be effective in these tasks and *(ii)* they are not dependent on minibatches.

6. Degree of Data Skew

In §4–§5, we studied a strict case of skewed label partitions, where each label only exists in a single data partition, *exclusively* (the one exception being our experiments with Flickr-Mammal). While this case may be a reasonable approximation for some applications (e.g., for FACE RECOGNITION, a person's face image may exist only in one data partition), it could be an extreme case for other applications (e.g., IMAGE CLASSIFICATION, as §2.2 shows). Here, we study how the problem changes with the degree of skew by controlling the fraction of the dataset that is non-IID (i.e., partitioned using labels, §3). Figure 6 shows the CIFAR-10 validation accuracy of GN-LeNet (our name for BN-LeNet with GroupNorm replacing BatchNorm) in the 20%, 40%, 60% and 80% non-IID setting. We make two observations.

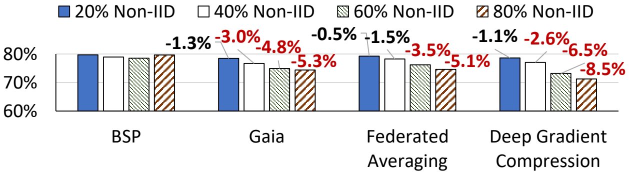


Figure 6. Top-1 validation accuracy for GN-LeNet over CIFAR-10, varying the degree of skew. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting. Note: The y-axis starts at 60% accuracy.

1) Partial non-IID data is also problematic. We see that for all three decentralized learning algorithms, partial non-IID data can still cause major accuracy loss. Even with a small degree of non-IID data such as 40%, we still see 1.5%–3.0% accuracy loss. Thus, the problem of non-IID data does not occur only with exclusive label partitioning, and the problem exists in the vast majority of practical settings.

2) Degree of skew often determines the difficulty level of the problem. The model accuracy gets worse with higher degrees of skew, and the accuracy gap between 80% and 20% non-IID data can be as large as 7.4% (`DeepGradientCompression`). In general, we see that the problem becomes more difficult with higher degree of skew.

7. Our Approach: SkewScout

To address the problem of skewed label partitions, we introduce SkewScout, a general approach that enables communication-efficient decentralized learning over *arbitrarily* skewed label partitions.

7.1. Overview of SkewScout

We design SkewScout as a general module that can be seamlessly integrated with different decentralized learning algorithms, ML training frameworks, and ML applications. Figure 7 provides an overview of the SkewScout design.

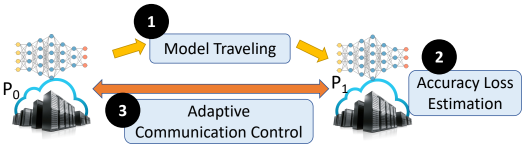


Figure 7. Overview of SkewScout

1. **Estimating the degree of skew.** As §6 shows, knowing the degree of skew is very useful to determine an appropriate solution. To learn this information, SkewScout periodically moves the ML model from one data partition (P_k) to another during training (*model traveling*, ❶ in Figure 7). SkewScout then evaluates how well a model performs on a remote data partition by evaluating the model accuracy with a subset of training data on the remote node. As we already know the training accuracy of this model in its original data partition, we can infer the *accuracy loss* in the remote data partition (❷).
2. **Adaptive communication control (❸).** Based on the estimated accuracy loss, SkewScout controls the amount of communication among data partitions to retain model quality. SkewScout controls the amount of communication by automatically tuning the hyper-parameters of the decentralized learning algorithm (§4.4). This tuning process essentially solves an optimization problem that aims to minimize communication among data partitions while keeping accuracy loss within a reasonable threshold (further details below).

In essence, SkewScout handles non-IID data partitions by controlling communication based on accuracy loss. SkewScout is agnostic to the source of the loss, which may be due to skewed label partitions or other forms of non-IID data (Appendix K). As long as increasing communication improves accuracy for the data skew, SkewScout should be effective in retaining model quality while minimizing communication.

7.2. Mechanism Details

We discuss the mechanisms of SkewScout in detail.

Accuracy Loss. The accuracy loss between data partitions represents the degree of model divergence. As §4.3 discusses, ML models in different data partitions tend to specialize for their training data, especially when we use decentralized learning algorithms to reduce communication.

We study accuracy loss under Gaia, for hyper-parameter choices T_0=$2\%, 5\%, 10\%, 20\%$, in the IID and non-IID settings. We find that accuracy loss changes drastically from the IID setting (0.4% on average) to the Non-IID setting (39.6% on average), and that lower T_0 results in smaller accuracy loss in the non-IID setting. See Appendix J for further details. Accordingly, we can use accuracy loss (i) to estimate how much the models diverge from each other (reflecting training data differences); and (ii) to serve as an objective function for communication control. The computation overhead to evaluate accuracy loss is quite small because we run inference with only a small fraction of training data, and we only do so once in a while (we empirically find that once every 500 mini-batches is frequent enough).

Communication Control. The goal of communication control is to retain model quality while minimizing communication among data partitions. We achieve this by solving an optimization problem, which aims to minimize communication while keeping the *accuracy loss* below a small threshold σ_{AL} so that we can control model divergence caused by non-IID data partitions. We solve this optimization problem periodically after we estimate the accuracy loss with model traveling. Specifically, our target function is:

$$\underset{\theta}{\operatorname{argmin}} \left(\lambda_{AL} \left(\mathtt{max}(0, AL(\theta) - \sigma_{AL}) \right) + \lambda_C \frac{C(\theta)}{CM} \right) \quad (1)$$

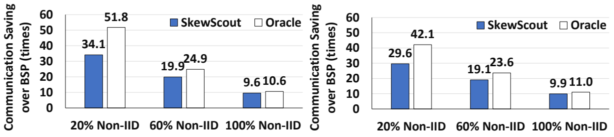
where $AL(\theta)$ is the accuracy loss based on the previously selected hyper-parameter θ (we memoize the most recent value for each θ that has been explored), $C(\theta)$ is the amount of communication given θ, CM is the communication cost for the whole ML model, and λ_{AL}, λ_C are given parameters to determine the weights of accuracy loss and communication, respectively. We can employ various algorithms with Equation 1 to select θ, such as hill climbing, stochastic hill climbing (Russell & Norvig, 2020), and simulated annealing (Van Laarhoven & Aarts, 1987).

7.3. Evaluation Results

We implement and evaluate SkewScout in a GPU parameter server system (Cui et al., 2016) based on Caffe (Jia et al., 2014). We evaluate several aforementioned auto-tuning algorithms and we find that hill climbing provides the best results. As our primary goal is to minimize accuracy loss, we set $\lambda_{AL} = 50$ and $\lambda_{AC} = 1$. We set $\sigma_{AL} = 5\%$ to tolerate an acceptable accuracy variation during training, which does not reduce the final validation accuracy.

We compare SkewScout with two other baselines: (1) BSP: the most communication-heavy approach that retains model accuracy in all Non-IID settings; and (2) Oracle: the ideal, yet unrealistic, approach that selects the most communication-efficient θ that retains model accuracy, by *running all possible* θ in each setting prior to measured execution. Figure 8 shows the communication savings over BSP for both SkewScout and Oracle when training with Gaia. Note that all results achieve *the same* validation accuracy as BSP. We make two observations.

First, SkewScout is much more effective than BSP in handling Non-IID settings. Overall, SkewScout achieves 9.6–34.1× communication savings over BSP in various Non-IID settings without sacrificing model accuracy. As expected, SkewScout saves more communication with less skewed data because SkewScout can safely loosen communication.



(a) AlexNet **(b)** GoogLeNet

Figure 8. Communication savings over BSP with SkewScout and `Oracle` for training with CIFAR-10. All results achieve the same accuracy as BSP in the IID setting.

Second, SkewScout is not far from the ideal `Oracle` baseline. Overall, SkewScout requires only 1.1–1.5× more communication than `Oracle` to achieve the same model accuracy. SkewScout cannot match the communication savings of `Oracle` because: (i) SkewScout does model traveling periodically, which leads to some overhead; and (ii) for some θ, high accuracy loss at the beginning can still lead to a high accuracy model, which SkewScout cannot foresee. As `Oracle` requires *many* runs in practice, we conclude that SkewScout is an effective, realistic one-pass solution for decentralized learning over non-IID data partitions.

8. Related Work

To our knowledge, this is the first study to show that skewed label partitions across devices/locations is a fundamental and pervasive problem for decentralized learning. Our study investigates various aspects of this problem, such as a real-world dataset, decentralized learning algorithms, batch normalization, and data skew, as well as presenting our SkewScout approach. Here, we discuss related work.

Large-scale systems for centralized learning. There are many large-scale ML systems that aim to enable efficient ML training over *centralized* datasets using communication-efficient designs, such as relaxing synchronization requirements (Recht et al., 2011; Ho et al., 2013; Goyal et al., 2017) or sending fewer updates to parameter servers (Li et al., 2014a;b). These works assume the training data are centralized so they can be easily partitioned among the machines performing the training in an IID manner (e.g., by random shuffling). Hence, they are neither designed for nor validated on non-IID data partitions.

Decentralized learning. Recent prior work proposes communication-efficient algorithms (e.g., (Hsieh et al., 2017; McMahan et al., 2017; Shokri & Shmatikov, 2015; Lin et al., 2018; Tang et al., 2018)) for ML training over *decentralized* datasets. However, as our study shows, these decentralized learning algorithms lose significant model accuracy in the Non-IID setting (§4). Some recent work studies the problem of non-IID data partitions. For example, instead of training a global model to fit non-IID data partitions, federated multi-task learning (Smith et al., 2017) trains local models for each data partition while leveraging other data partitions to improve model accuracy. However, this approach sidesteps the problem for global models, which are essential when a local model is unavailable (e.g., a brand new partition without training data) or ineffective (e.g., a partition with too few training examples for a class). Several recent works show significant accuracy loss for `FederatedAveraging` over non-IID data, and some propose algorithms to improve `FederatedAveraging` over non-IID data (Zhao et al., 2018; Li et al., 2019; Shoham et al., 2019; Karimireddy et al., 2019; Liang et al., 2019; Li et al., 2020a; Wang et al., 2020; Khaled et al., 2020). While the result of these works aligns with our observations, our study *(i)* broadens the problem scope to a variety of decentralized learning algorithms, ML applications, DNN models, and datasets, *(ii)* explores the problem of batch normalization and possible solutions, and *(iii)* designs and evaluates SkewScout, which can also complement the aforementioned algorithms by controlling their hyper-parameters over arbitrarily skewed data partitions.

Non-IID dataset. Recent work offers non-IID datasets to facilitate the study of federated learning. For example, LEAF (Caldas et al., 2018) provides datasets that are partitioned in various ways. Luo et al. release 900 images collected from cameras in different locations, and they show severe skewed label distribution across cameras (Luo et al., 2019). Our study on geo-tagged mammals on Flickr shows the same problem at a much larger scale, and our dataset broadens the scope to include geo-distributed learning.

9. Conclusion

As most timely and relevant ML data are generated at different physical locations, and often infeasible/impractical to collect centrally, decentralized learning provides an important path for ML applications to leverage such data. However, decentralized data is often generated at different contexts, which leads to a heavily understudied problem: *non-IID training data partitions*. We conduct a detailed empirical study of this problem for skewed label partitions, revealing three key findings. First, we show that training over skewed label partitions is a fundamental and pervasive problem for decentralized learning, as all decentralized learning algorithms in our study suffer major accuracy loss. Second, we find that DNNs with batch normalization are particularly vulnerable in the Non-IID setting, with even the most communication-heavy approach being unable to retain model quality. We further discuss the cause and a potential solution to this problem. Third, we show that the difficulty level of this problem varies greatly with the degree of skew. Based on these findings, we present SkewScout, a general approach to minimizing communication while retaining model quality even for non-IID data. We hope that the findings and insights in this paper, as well as our open source code and dataset, will spur further research into the fundamental and important problem of non-IID data in decentralized learning.

Acknowledgments

We thank H. Brendan McMahan, Gregory R. Ganger, and the anonymous ICML reviewers for their valuable and constructive suggestions. Special thanks to Chenghao Zhang for his help in setting up the face recognition experiments. We also thank the members and companies of the PDL Consortium (Alibaba, Amazon, Datrium, Facebook, Google, Hewlett-Packard Enterprise, Hitachi, IBM, Intel, Microsoft, NetApp, Oracle, Salesforce, Samsung, Seagate, and Two Sigma) for their interest, insights, feedback, and support. We acknowledge the support of the SAFARI Research Group's industrial partners: Google, Huawei, Intel, Microsoft, and VMware. This work is supported in part by NSF. Most of the work was done when Kevin Hsieh was at Carnegie Mellon University.

References

Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. Understanding batch normalization. In *NeurIPS*, 2018.

Briggs, C., Fan, Z., and Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. *CoRR*, abs/2004.11791, 2020.

Caldas, S., Wu, P., Li, T., Konecný, J., McMahan, H. B., Smith, V., and Talwalkar, A. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

Cui, H., Zhang, H., Ganger, G. R., Gibbons, P. B., and Xing, E. P. GeePS: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server. In *EuroSys*, 2016. Software available at https://github.com/cuihenggang/geeps.

European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46 (general data protection regulation). *Official Journal of the European Union (OJ)*, 59, 2016.

Flickr. `https://www.flickr.com/`.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch SGD: training ImageNet in 1 hour. *CoRR*, abs/1706.02677, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J. K., Gibbons, P. B., Gibson, G. A., Ganger, G. R., and Xing, E. P. More effective distributed ML via a stale synchronous parallel parameter server. In *NIPS*, 2013.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8), 1997.

Hsieh, K., Harlap, A., Vijaykumar, N., Konomis, D., Ganger, G. R., Gibbons, P. B., and Mutlu, O. Gaia: Geo-distributed machine learning approaching LAN speeds. In *NSDI*, 2017.

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

Ioffe, S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NIPS*, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

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. *CoRR*, abs/1910.06378, 2019.

Khaled, A., Mishchenko, K., and Richtárik, P. Tighter theory for local SGD on identical and heterogeneous data. In *AISTATS*, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J. R. R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., and Ferrari, V. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018.

Laguel, Y., Pillutla, K., Malick, J., and Harchaoui, Z. Device heterogeneity in federated learning: A superquantile approach. *CoRR*, abs/2002.11223, 2020.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B. Scaling distributed machine learning with the parameter server. In *OSDI*, 2014a.

Li, M., Andersen, D. G., Smola, A. J., and Yu, K. Communication efficient distributed machine learning with the parameter server. In *NIPS*, 2014b.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *MLSys*, 2020a.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *ICLR*, 2020b.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of FedAvg on non-IID data. *CoRR*, abs/1907.02189, 2019.

Liang, X., Shen, S., Liu, J., Pan, Z., Chen, E., and Cheng, Y. Variance reduced local SGD with lower communication complexity. *CoRR*, abs/1912.12844, 2019.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*, 2018.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A. L., Huang, J., and Murphy, K. Progressive neural architecture search. In *ECCV*, 2018.

Luo, J., Wu, X., Luo, Y., Huang, A., Huang, Y., Liu, Y., and Yang, Q. Real-world image datasets for federated learning. *CoRR*, abs/1910.11089, 2019.

Mansour, Y., Mohri, M., Ro, J., and Suresh, A. T. Three approaches for personalization with applications to federated learning. *CoRR*, abs/2002.10619, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

Recht, B., Ré, C., Wright, S. J., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3), 1951.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2020.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *NeurIPS*, 2018.

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. *CoRR*, abs/1910.07796, 2019.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *CCS*, 2015.

Smith, V., Chiang, C., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *NIPS*, 2017.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D^2: Decentralized training over decentralized data. In *ICML*, 2018.

United Nation Statistics Division. Methodology: Standard country or area codes for statistical use (m49), 2019.

Valiant, L. G. A bridging model for parallel computation. *Communications of the ACM*, 33(8), 1990.

Van Laarhoven, P. J. and Aarts, E. H. Simulated annealing. In *Simulated Annealing: Theory and Applications*. Springer, 1987.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D. S., and Khazaeni, Y. Federated learning with matched averaging. In *ICLR*, 2020.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016.

Wu, Y. and He, K. Group normalization. In *ECCV*, 2018.

Yi, D., Lei, Z., Liao, S., and Li, S. Z. Learning face representation from scratch. *CoRR*, abs/1411.7923, 2014.

Yu, T., Bagdasaryan, E., and Shmatikov, V. Salvaging federated learning by local adaptation. *CoRR*, abs/2002.04758, 2020.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-IID data. *CoRR*, abs/1806.00582, 2018.

Appendix

A. Details of Decentralized Learning Algorithms

This section presents the pseudocode for `Gaia`, `FederatedAveraging`, and `DeepGradientCompression`.

Algorithm 1 `Gaia` (Hsieh et al., 2017) on node k for vanilla momentum SGD

```
Input: initial weights $w_0 = \{w_0[0], ..., w_0[M]\}$
Input: $K$ data partitions (or data centers); initial significance threshold $T_0$
Input: local minibatch size $B$; momentum $m$; learning rate $\eta$; local dataset $\mathcal{X}_k$
$u^k_0 \leftarrow 0$; $v^k_0 \leftarrow 0$
$w^k_0 \leftarrow w_0$
for $t = 0, 1, 2, ...$ do
    $b \leftarrow$ (sample $B$ data samples from $\mathcal{X}_k$)
    $u^k_{t+1} \leftarrow m \cdot u^k_t - \eta \cdot \bigtriangledown f(w^k_t, b)$
    $w^k_{t+1} \leftarrow w^k_t + u^k_{t+1}$
    $v^k_{t+1} \leftarrow v^k_t + u^k_{t+1}$                         ▷ Accumulate weight updates
    for $j = 0, 1, ...M$ do
        $S \leftarrow ||\frac{v^k_{t+1}}{w^k_{t+1}}|| > T_t$             ▷ Check if accumulated updates are significant
        $\widetilde{v}^k_{t+1}[j] \leftarrow v^k_{t+1}[j] \odot S$        ▷ Share significant updates with other $P_k$
        $v^k_{t+1}[j] \leftarrow v^k_{t+1}[j] \odot \neg S$              ▷ Clear significant updates locally
    end for
    for $i = 0, 1, ...K; i \neq k$ do
        $w^k_{t+1} \leftarrow w^k_{t+1} + \widetilde{v}^i_{t+1}$          ▷ Apply significant updates from other $P_k$
    end for
    $T_{t+1} \leftarrow$ update_threshold($T_t$)     ▷ Decrease threshold whenever the learning rate decreases
end for
```

Algorithm 2 `FederatedAveraging` (McMahan et al., 2017) on node k for vanilla momentum SGD

```
Input: initial weights $w_0$; $K$ data partitions (or clients)
Input: local minibatch size $B$; local iteration number $Iter_{Local}$
Input: momentum $m$; learning rate $\eta$; local dataset $\mathcal{X}_k$
$u^k \leftarrow 0$
for $t = 0, 1, 2, ...$ do
    $w^k_t \leftarrow w_t$                                   ▷ Get the latest weights from the server
    for $i = 0, ...Iter_{Local}$ do
        $b \leftarrow$ (sample $B$ data samples from $\mathcal{X}_k$)
        $u^k \leftarrow m \cdot u^k - \eta \cdot \bigtriangledown f(w^k_t, b)$
        $w^k_t \leftarrow w^k_t + u^k$
    end for
    all_reduce: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{1}{K} w^k_t$          ▷ Average weights from all partitions
end for
```

In order make our experiments deterministic and simpler, we use all data partitions (or clients) in every epoch for `FederatedAveraging`.

Algorithm 3 `DeepGradientCompression` (Lin et al., 2018) on node k for vanilla momentum SGD

```
Input: initial weights $w_0 = \{w_0[0], ..., w_0[M]\}$
Input: $K$ data partitions (or data centers); $s\%$ update sparsity
Input: local minibatch size $B$; momentum $m$; learning rate $\eta$; local dataset $\mathcal{X}_k$
$u_0^k \leftarrow 0$; $v_0^k \leftarrow 0$
for $t = 0, 1, 2, ...$ do
    $b \leftarrow$ (sample $B$ data samples from $\mathcal{X}_k$)
    $g_{t+1}^k \leftarrow -\eta \cdot \bigtriangledown f(w_t, b)$
    $g_{t+1}^k \leftarrow \texttt{gradient\_clipping}(g_{t+1}^k)$                ▷ Clip gradients
    $u_{t+1}^k \leftarrow m \cdot u_t^k + g_{t+1}^k$
    $v_{t+1}^k \leftarrow v_t^k + u_{t+1}^k$                                      ▷ Accumulate weight updates
    $T \leftarrow s\%$ of $||v_{t+1}^k||$                                          ▷ Determine the threshold for sparsified updates
    for $j = 0, 1, ...M$ do
        $S \leftarrow ||v_{t+1}^k|| > T$                                           ▷ Check if accumulated updates are top $s\%$
        $\widetilde{v}_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot S$                 ▷ Share top updates with other $P_k$
        $v_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot \neg S$                        ▷ Clear top updates locally
        $u_{t+1}^k[j] \leftarrow u_{t+1}^k[j] \odot \neg S$                        ▷ Clear the history of top updates (momentum correction)
    end for
    $w_{t+1} = w_t + \sum_{k=1}^{K} \widetilde{v}_{t+1}^k$                         ▷ Apply top updates from all $P_k$
end for
```

B. Details of Geographical Distribution of Mammal Pictures on Flickr

B.1. Dataset Details

We query Flickr for the top 40,000 images (4000 images from each of 10 years) for each of the 48 mammal classes in Open Images V4 (Kuznetsova et al., 2018). We then use PNAS (Liu et al., 2018) to clean the search results. As PNAS is pre-trained on ImageNet, we can only consider classes that exist both in Open Image and ImageNet. As a result, we remove 7 classes from our dataset (Bat, Dog, Raccoon, Giraffe, Rhinoceros, Horse, Mouse). Note that while ImageNet has many dogs, they are categorized into hundreds of classes. Hence, we remove dogs in our dataset for simplicity. We run all the images through PNAS, and keep all the images with a matching class result in the top-5 predictions.

Figure 9 shows the number of images in each class of our Flickr-Mammal dataset. As expected, popular mammals (e.g., cat and squirrel) have a lot more images than less popular mammals (e.g., armadillo and skunk). The gap between different classes is large: the most popular mammal (cat) has 23× more images than the least popular mammal (skunk). Nonetheless, the vast majority of classes have at least 10,000 images. Even the least popular mammal has 1,531 images, which is a reasonable number for DNN training. In comparison, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 has around 1,200 images for each class.

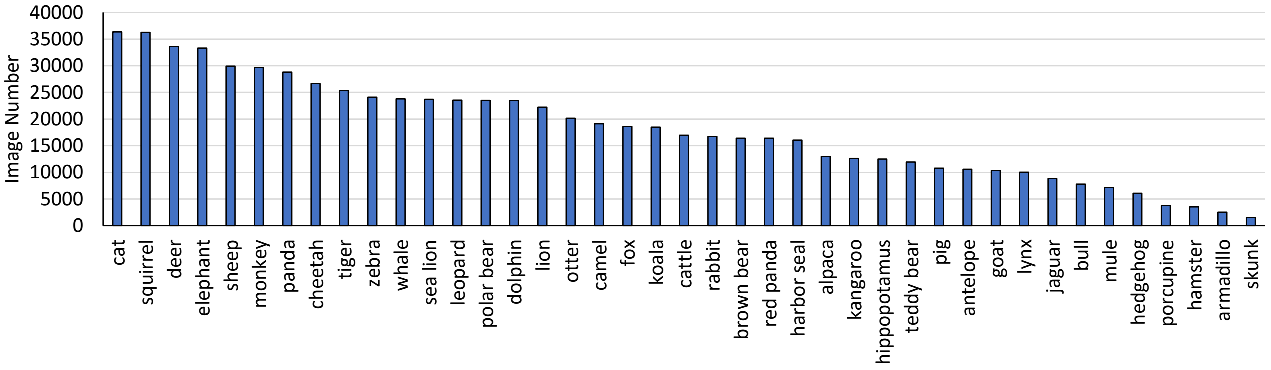


Figure 9. Flickr-Mammal dataset: The number of images in each mammal class.

B.2. First-Level Geographical Region Analysis

As §2.2 mentions, we use the M49 Standard (United Nation Statistics Division, 2019) to map the geotag of each image to different regions. The first-level regions in the M49 Standard are the continents. Figure 10 shows the number of images in each continent (our analysis omits the 53 images that were not from any one of these five continents). There is an inherent skew in the number of images in each continent: Americas and Europe have significantly more pictures than the other continents, probably because these two continents have more people who use Flickr to upload pictures.

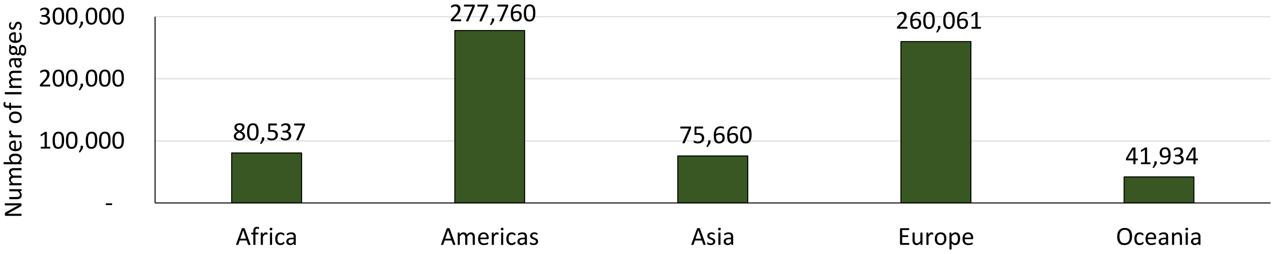


Figure 10. Flickr-Mammal dataset: The number of images in each continent.

Share of raw samples across continents. Figure 11 depicts the share of samples across continents for each mammal class. As expected, Americas and Europe dominate the share of images for many mammals as they have more images than other continents (Figure 10). However, the geographical distribution of mammals is the main reason for the skew in the share distribution. For example, Oceania has more than 70% of Kangaroo and Koala images even though it only has 6% of the total images. Similarly, Africa has more than 40% of Antelope, Cheetah, Elephant, Hippopotamus, Lion, and Zebra images while it has only 11% of the total images. Overall, we see that the vast majority of mammals are dominated by two or three continents, leaving the other continents with a small number of image samples for these mammal classes.

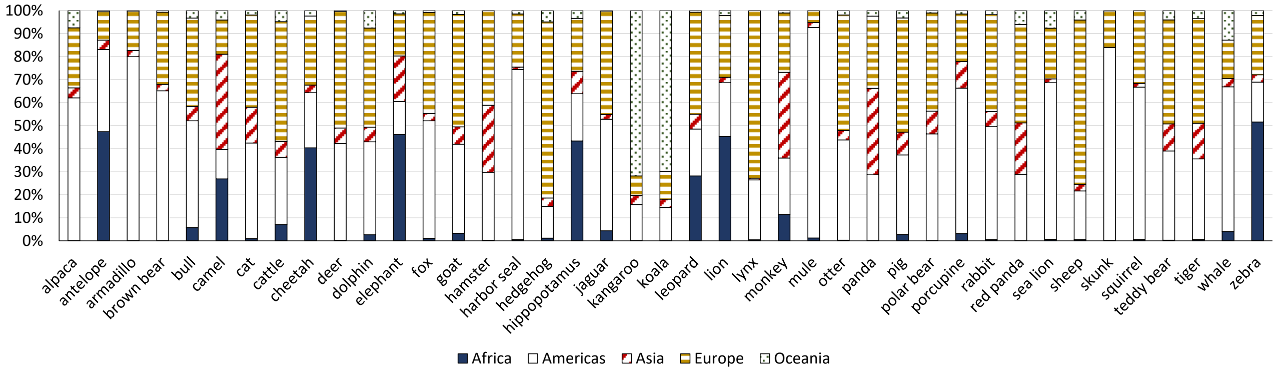


Figure 11. Flickr-Mammal dataset: The share of images in each continent based on raw samples.

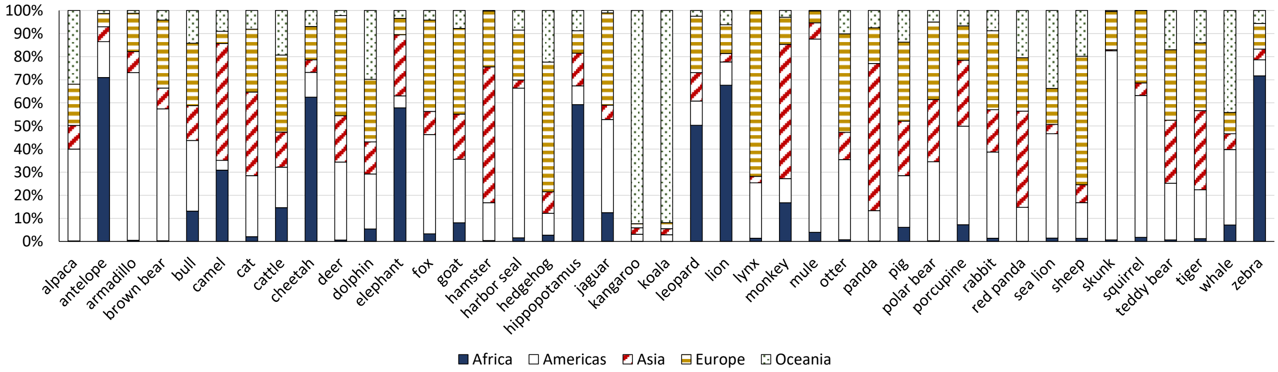


Figure 12. Flickr-Mammal dataset: The share of images in each continent based on normalized samples.

Share of normalized samples across continents. As we are mostly interested in the distribution of labels ($\mathcal{P}(y)$) among different continents, we normalize the number of images so that each continent has *the same number of total images*. Table 1 in §2.2 shows the top-5 mammals in each continent based on these normalized samples. Here, Figure 12 illustrates the normalized sample share for all mammals across continents. As we see, the overall label distribution is similar between normalized samples (Figure 12) and raw samples (Figure 11). The continent that dominates a mammal class in the raw sample distribution tends to be even more dominant in the normalized sample distribution. For example, Africa consists of 50% to 70% of the African mammals (e.g., Antelope, Cheetah, Elephant, etc.) in the normalized sample distribution, compared to 40% in the raw sample distribution. We conclude that skewed distribution of labels is a natural phenomenon, and both raw samples and normalized samples exhibit very significant skew across common mammals.

B.3. Second-Level Geographical Region Analysis

We also analyze our dataset using the second-level regions (subcontinents) in the M49 Standard. We remove the second-level regions that have fewer than 1,000 images in our analysis (Central Asia, Melanesia, Micronesia, and Polynesia), resulting in 13 subcontinents and 735,071 images. Figure 13 shows the number of images in each subcontinent. Similar to Figure 10, we see that Northern America and Northern Europe have significantly more images than other subcontinents.

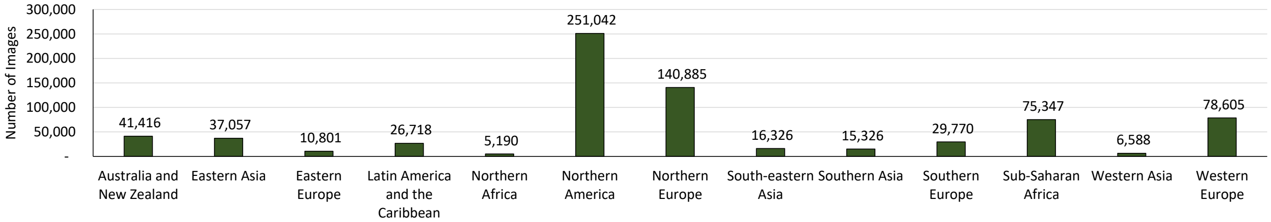


Figure 13. Flickr-Mammal dataset: The number of images in each subcontinent.

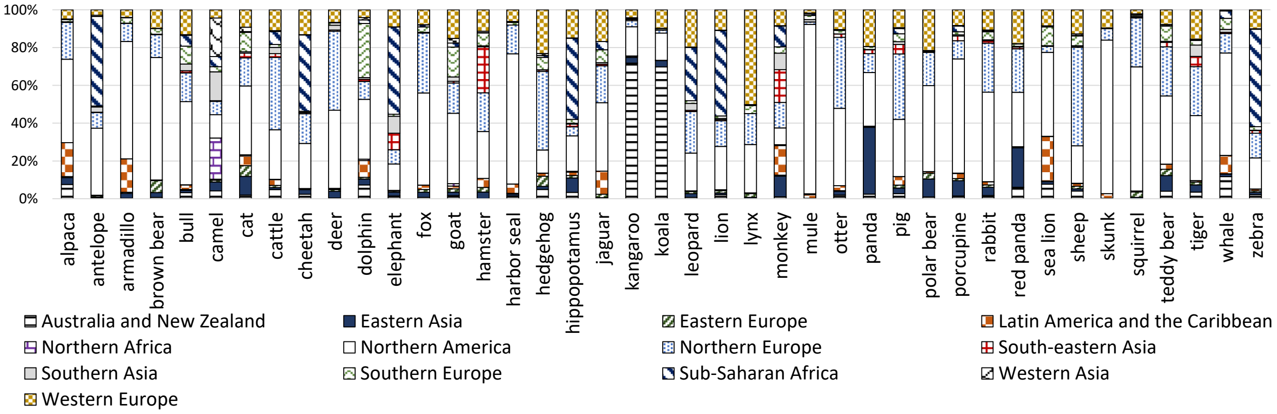


Figure 14. Flickr-Mammal dataset: The share of images in each subcontinent based on raw samples.

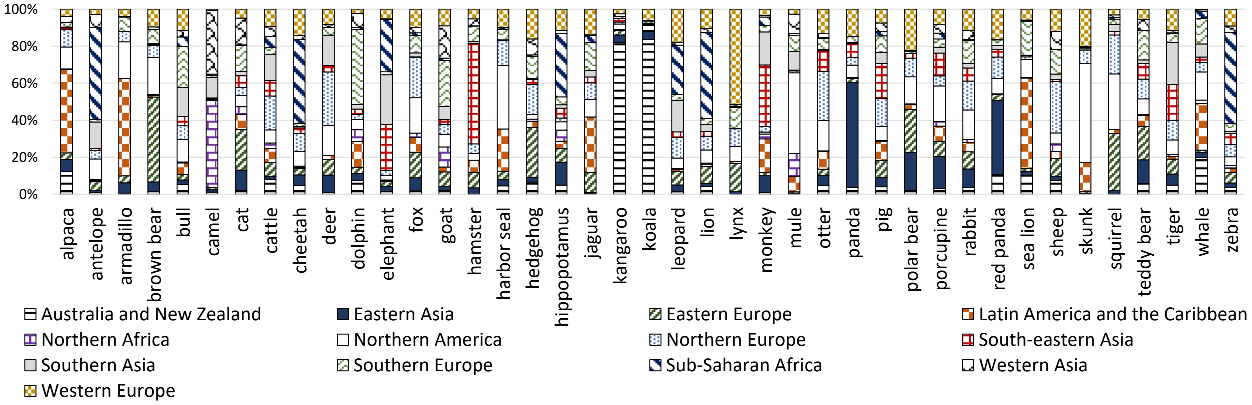


Figure 15. Flickr-Mammal dataset: The share of images in each subcontinent based on normalized samples.

Share of samples across subcontinents. Figure 14 illustrates the share of samples across subcontinents for each mammal class. Again, we observe that the label distribution is highly skewed. Among the 13 subcontinents, the vast majority of mammal classes mostly exist in 3-5 subcontinents. Furthermore, the sample concentration pattern varies greatly among mammal classes. For example, Kangaroo and Koala are mostly in Australia and New Zealand, Antelope and Zebra are mostly in Sub-Saharan Africa, and Mule and Skunk are mostly in Northern America. On average, 5 of the 13 subcontinents contain less than 1% of the images for each mammal class. We also show the normalized sample share across subcontinents (Figure 15), and we can see the difference of $\mathcal{P}(y)$ among subcontinents. Overall, our analysis shows that skewed label distribution is also very common at the subcontinent-level.

C. Training Parameters

Tables 2–5 list the major training parameters for all the applications, models, and datasets in our study.

Model	**Minibatch size per node (5 nodes)**	**Momentum**	**Weight decay**	**Learning rate**	**Total epochs**
AlexNet	20	0.9	0.0005	$\eta_0 = 0.0002$, divides by 10 at epoch 64 and 96	128
GoogLeNet	20	0.9	0.0005	$\eta_0 = 0.002$, divides by 10 at epoch 64 and 96	128
LeNet, BN-LeNet, GN-LeNet	20	0.9	0.0005	$\eta_0 = 0.002$, divides by 10 at epoch 64 and 96	128
ResNet-20	20	0.9	0.0005	$\eta_0 = 0.002$, divides by 10 at epoch 64 and 96	128

Table 2. Major training parameters for IMAGE CLASSIFICATION over CIFAR-10

Model	**Minibatch size per node (8 nodes)**	**Momentum**	**Weight decay**	**Learning rate**	**Total epochs**
GoogLeNet	32	0.9	0.0002	$\eta_0 = 0.0025$, polynomial decay, power = 0.5	60
ResNet-10	32	0.9	0.0001	$\eta_0 = 0.00125$, polynomial decay, power = 1	64

Table 3. Major training parameters for IMAGE CLASSIFICATION over ImageNet. Polynomial decay means $\eta = \eta_0 \cdot (1 - \frac{\text{iter}}{\text{max\_iter}})^{\text{power}}$.

Model	**Minibatch size per node (4 nodes)**	**Momentum**	**Weight decay**	**Learning rate**	**Total epochs**
center-loss	64	0.9	0.0005	$\eta_0 = 0.025$, divides by 10 at epoch 4 and 6	7

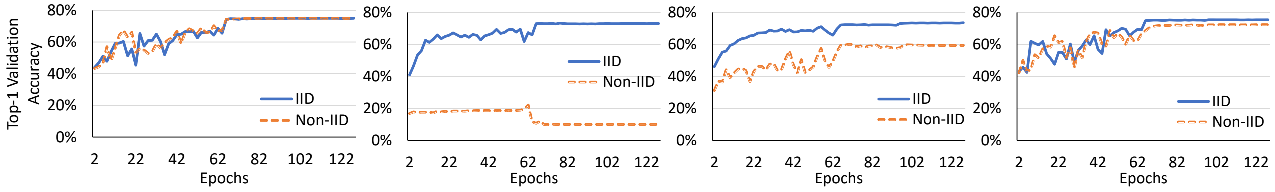
Table 4. Major training parameters for FACE RECOGNITION over CASIA-WebFace.

Model	**Minibatch size per node (5 nodes)**	**Momentum**	**Weight decay**	**Learning rate**	**Total epochs**
GoogLeNet	32	0.9	0.0002	$\eta_0 = 0.004$, polynomial decay, power = 0.5	55

Table 5. Major training parameters for IMAGE CLASSIFICATION over Flickr-Mammal.

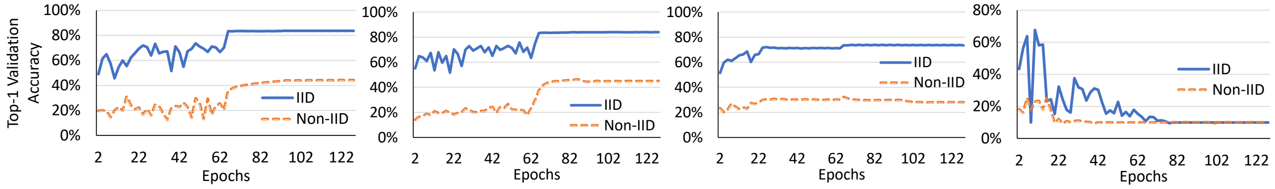
D. Training Convergence Curves

Figures 16 and 17 show the training convergence curves for AlexNet and ResNet20 over the CIFAR-10 dataset. We make two major observations. First, all training processes stop improving long before the end of experiments, which suggest longer training cannot solve the problem of non-IID data. Second, the convergence curves in the Non-IID settings generally follow similar trends to the curves in the IID settings, but the model accuracy is significantly lower. Appendix G discusses the potential reasons behind this phenomenon. As we discuss in §5, even BSP loses significant accuracy for DNN models with BatchNorm, which explains the curves in Figure 17.



(a) BSP **(b)** Gaia **(c)** Federated Averaging **(d)** Deep Gradient Compression

Figure 16. The training convergence curves for AlexNet over the CIFAR-10 dataset.



(a) BSP **(b)** Gaia **(c)** Federated Averaging **(d)** Deep Gradient Compression

Figure 17. The training convergence curves for ResNet20 over the CIFAR-10 dataset.

E. Image Classification with ImageNet

§4.1 summarized our results for Image Classification over the ImageNet dataset (Russakovsky et al., 2015) (1,000 image classes). In this section, we provide the details.

We use two partitions ($K = 2$) in this experiment so each partition contains 500 image classes. According to the hyper-parameter criteria in §3, we select $T_0 = 40\%$ for `Gaia`, $Iter_{Local} = 200$ for `FederatedAveraging`, and $E_{warm} = 4$ for `DeepGradientCompression`. Figure 18 shows the validation accuracy in the IID and Non-IID settings.

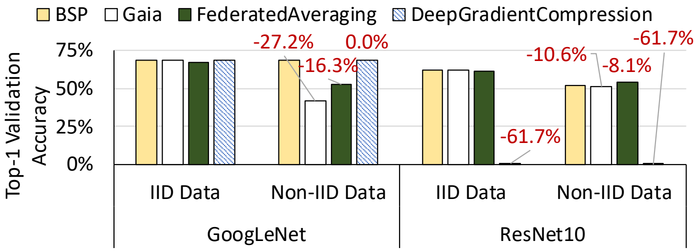


Figure 18. Top-1 validation accuracy for Image Classification over the ImageNet dataset. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting.

Interestingly, we observe the same problems in the ImageNet dataset as in the CIFAR-10 dataset (§4.1), even though the number of classes in ImageNet is two orders of magnitude larger than in CIFAR-10. First, we see that `Gaia` and `FederatedAveraging` lose significant validation accuracy (8.1% to 27.2%) for both DNNs in the Non-IID setting. On the other hand, while `DeepGradientCompression` is able to retain the validation accuracy for GoogLeNet in the Non-IID setting, it cannot converge to a useful model for ResNet10. Second, BSP also cannot retain the validation accuracy for ResNet10 in the Non-IID setting, which concurs with our observation in the CIFAR-10 study. Together with the results in §4.1, these results show that the Non-IID data problem exists not only in various decentralized learning algorithms and DNNs, but also in different image datasets.

F. Effect of Larger Numbers of Data Partitions

So far, we have used a relatively modest number of data partitions ($K = 2$ or $K = 5$) to demonstrate the Non-IID data problem in our study. Here, we study the effect of having a larger number of data partitions.

CIFAR-10. We compare the model accuracy of ResNet20 using the CIFAR-10 dataset with ten data partitions ($K = 10$). We quickly discover that even training with BSP *does not* converge in the 100% Non-IID setting. This is because each partition has only one object class (CIFAR-10 consists of ten object classes), so the gradients from different data partitions diverge too much. Instead, we create a Non-IID setting such that each partition has 80% of one object class and 20% of another object class. Figure 19 shows the results. The hyper-parameters are the same as in §4.1. We observe that decentralized learning algorithms experience similar model accuracy loss with $K = 10$ compared to $K = 5$. This is interesting as we have a relatively easier Non-IID setting for $K = 10$. We also observe that with $K = 10$, decentralized learning algorithms lose more accuracy relative to BSP in the Non-IID setting compared to $K = 5$. When $K = 10$, `Gaia` and `FederatedAveraging` lose 3% and 36% compared to BSP in the Non-IID setting, which is larger than their 0% and 17% losses when $K = 5$. These results suggest that a larger number of data partitions negatively impacts model accuracy in the Non-IID setting.

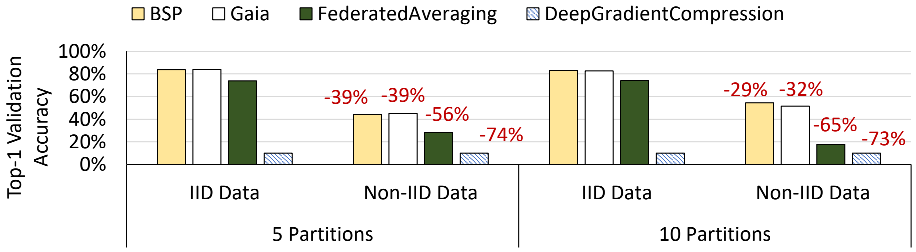


Figure 19. Top-1 validation accuracy for ResNet20 over the CIFAR-10 dataset, with 5 and 10 data partitions. The five partition results are repeated from Figure 1. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting.

Flickr-Mammal. We create a real-world non-IID setting using the locations of mammal images in the second-level (subcontinent) regions. As §B.3 shows, there are thirteen subcontinents in our dataset so we have $K = 13$. For comparison, we create an artificial IID setting, in which all images are randomly distributed among the 13 partitions. Figure 20 shows the results of running BSP, `Gaia`, and `FederatedAveraging` in these settings. We use GoogLeNet in this experiment. We select $T_0 = 10\%$ for `Gaia` and $Iter_{Local} = 20$ for `FederatedAveraging` based on the criteria in §3. We see that both `Gaia` and `FederatedAveraging` lose more accuracy when data are partitioned at the subcontinent level ($K = 13$) than at the continent level ($K = 5$). This is expected because the vast majority of mammals mostly exist in 3-5 subcontinents (Figure 15), so many subcontinents do not have all the mammal labels. In contrast, most continents have all the mammal labels, which reduces the difficulty level of the problem. This result suggests that the non-IID data problem can have a more severe impact with a larger number of data partitions in the real world.

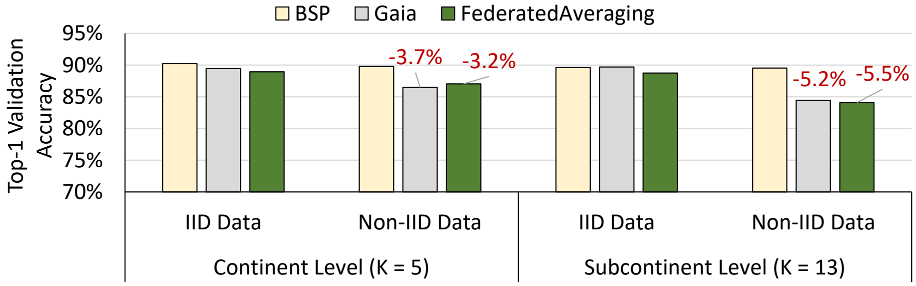


Figure 20. Top-1 validation accuracy for GoogLeNet over the Flickr-Mammal dataset, which is partitioned at the continent level and the subcontinent level. 5% of data are randomly selected as the validation set. The five partition results are repeated from Figure 2. Non-IID Data is based on real-world data distribution among continents or subcontinents, and IID Data is the artificial setting in which training images are randomly assigned to partitions. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting. Note: The y-axis starts at 70% accuracy.

G. Reasons for Model Quality Loss

Gaia. As discussed in §4.3, Gaia saves communication by allowing small model differences in each partition P_k, and this gives each P_k room for specializing to its local data. To demonstrate this, we extract the Gaia-trained models from both partitions (denoted DC-0 and DC-1) for the GoogLeNet experiment in Figure 18, and then evaluate the validation accuracy of each model based on the *image classes* in each partition. As Figure 21 shows, the validation accuracy is very consistent between the two sets of image classes when training the model in the IID setting: the results for IID DC-0 Model are shown, and IID DC-1 Model is the same. However, the validation accuracy varies drastically under the Non-IID setting (Non-IID DC-0 Model and Non-IID DC-1 Model). Specifically, both models perform well for the image classes in their respective partitions, but they perform very poorly for the image classes that are *not* in their respective partitions. This reveals that using Gaia in the Non-IID setting results in *completely different* models among data partitions, and each model is only good for recognizing the image classes in its own data partition.

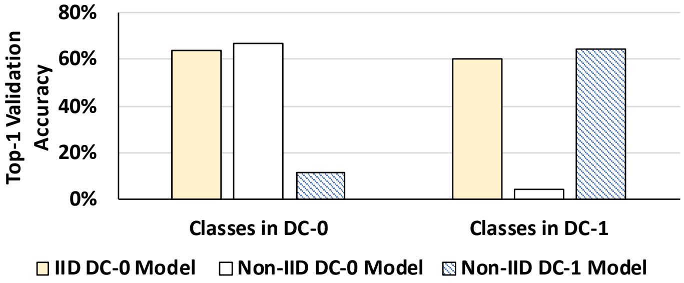


Figure 21. Top-1 validation accuracy (ImageNet) for models in different partitions.

This raises the following question: How does Gaia produce completely different models in the Non-IID setting, given that Gaia synchronizes all significant updates (Δw_j) to ensure that the differences across models in each weight w_j is insignificant (§2)? To answer this, we first compare each weight w_j in the Non-IID DC-0 and DC-1 Models, and find that the average difference among all the weights is only 0.5% (reflecting a 1% threshold for significance in the last epoch). However, we find that given the same input image, the *neuron* values are vastly different (with an average difference of 173%). This finding suggests that small model differences can result in completely different models. Mathematically, this is because weights can be positive or negative: a small percentage difference in individual weights can lead to a large percentage difference in the resulting neuron values, especially for neuron values that have small magnitudes. As Gaia eliminates insignificant communication, it creates an opportunity for models in each data partition to specialize for the image classes in their respective data partition, at the expense of other classes.

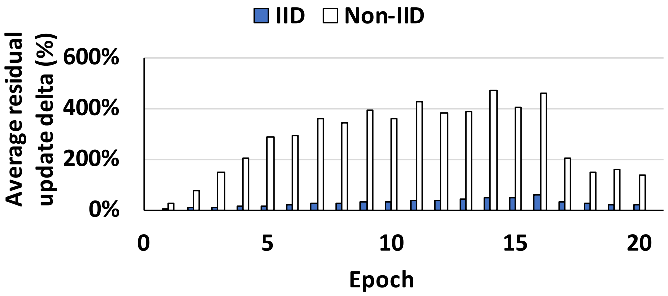


Figure 22. Average residual update delta (%) for DeepGradientCompression over the first 20 epochs.

DeepGradientCompression. DeepGradientCompression and FederatedAveraging always maintain *one* global model, and hence there must be a *different* reason for their model quality loss. For DeepGradientCompression, we examine the average residual update delta ($||\Delta w_i/w_i||$). This number represents the magnitude of the gradients that have *not* yet been exchanged among different P_k, as the algorithm communicates only a fixed number of gradients in each epoch (§2). Thus, it can be viewed as the amount of gradient divergence among different P_k. Figure 22 depicts the average residual update delta for the first 20 training epochs when training ResNet20 over CIFAR-10. (We show only the first 20 epochs because, as shown in Figure 17(d), training diverges after 20 epochs in the Non-IID setting.) As the figure shows, the average residual update delta is an order of magnitude higher in the Non-IID setting (283%) than that in the IID setting (27%). Hence, each P_k generates large gradients in the Non-IID setting, which is not surprising as each P_k sees vastly different training data. However, these large gradients are not synchronized because `DeepGradientCompression` sparsifies the gradients at a fixed rate. When they are finally synchronized, they may have diverged so much from the global model that they lead to the divergence of the whole model, and indeed our experiments often show such divergence.

FederatedAveraging. The analysis for `DeepGradientCompression` can also apply to `FederatedAveraging`, which delays communication from each P_k by a fixed number of local iterations. If the weights in different P_k diverge too much, the synchronized global model can lose accuracy or completely diverge (Zhao et al., 2018). We validate this by plotting the average local weight update delta for `FederatedAveraging` at each global synchronization point ($||\Delta w_i/w_i||$, where w_i is the averaged global model weight). Figure 23 depicts this number for the first 25 training epochs when training AlexNet over the CIFAR-10 dataset (Figure 16(c)). As the figure shows, the average local weight update delta in the Non-IID setting (48.5%) is much higher than that in the IID setting (20.2%), which explains why Non-IID data partitions lead to major accuracy loss for `FederatedAveraging`. The difference is less pronounced than with `DeepGradientCompression`, so the impact on accuracy is smaller with `FederatedAveraging`.

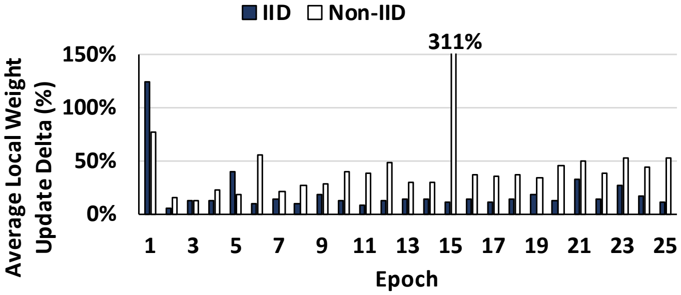


Figure 23. Average local update delta (%) for `FederatedAveraging` over the first 25 epochs.

H. Details on Algorithm Hyper-Parameters

We study the sensitivity of the non-IID problem to hyper-parameter choice. Tables 6, 7 and 8 present the results for `Gaia`, `FederatedAveraging` and `DeepGradientCompression`, respectively, by varying their respective hyper-parameters when training on CIFAR-10. We compare the results with BSP. Two major observations are in order.

First, almost all hyper-parameter settings lead to significant accuracy loss in the Non-IID setting (relative to BSP in the IID setting). Even with a relatively conservative hyper-parameter setting (e.g., $T_0 = 2\%$ for `Gaia` or $Iter_{Local} = 5$ for `FederatedAveraging`, the most communication-intensive of the choices shown), we still observe a 3.3% to 42.3% accuracy loss. On the other hand, the exact same hyper-parameter choice in the IID setting can mostly achieve BSP-level accuracy (except for ResNet20, which is troubled by the batch normalization problem, §5). We see the same trend with much more aggressive hyper-parameter settings as well (e.g., $T_0 = 40\%$ for `Gaia`). This shows that the problem of Non-IID data partitions is not specific to particular hyper-parameter settings, and that hyper-parameter settings that work well in the IID setting may perform poorly in the Non-IID setting.

Second, more conservative hyper-parameter settings (which implies more frequent communication among the P_k) often greatly decrease the accuracy loss in the Non-IID setting. For example, the validation accuracy with $T_0 = 2\%$ is significantly higher than the one with $T_0 = 30\%$ for `Gaia`. This supports SkewScout's approach (§7) that more frequent communication among the P_k leads to higher model quality in the Non-IID setting (mitigating the "tug-of-war" among the P_k (§2.1)).

I. More Alternatives to Batch Normalization

Weight Normalization (Salimans & Kingma, 2016). Weight Normalization (WeightNorm) normalizes the weights in a DNN as opposed to the neurons (which is what BatchNorm and most other normalization techniques do). WeightNorm is not dependent on minibatches as it normalizes the weights. However, while WeightNorm can effectively control the variance of the neurons, it still needs a mean-only BatchNorm in many cases to achieve the model quality and training speeds of BatchNorm (Salimans & Kingma, 2016). This mean-only BatchNorm makes WeightNorm vulnerable to the Non-IID setting again, because there is a large divergence in $\mu_{\mathcal{B}}$ among the P_k in the Non-IID setting (§5.1).

Configuration	AlexNet		GoogLeNet		LeNet		ResNet20	
	IID	Non-IID	IID	Non-IID	IID	Non-IID	IID	Non-IID
BSP	74.9%	75.0%	79.1%	78.9%	77.4%	76.6%	83.7%	**44.3%**
$T_0 = 2\%$	73.8%	**70.5%**	78.4%	**56.5%**	76.9%	**52.6%**	83.1%	**48.0%**
$T_0 = 5\%$	73.2%	**71.4%**	77.6%	**75.6%**	**74.6%**	**10.0%**	83.2%	**43.1%**
$T_0 = 10\%$	73.0%	**10.0%**	78.4%	**68.0%**	76.7%	**10.0%**	84.0%	**45.1%**
$T_0 = 20\%$	**72.5%**	**37.6%**	77.7%	**67.0%**	77.7%	**10.0%**	83.6%	**38.9%**
$T_0 = 30\%$	**72.4%**	**26.0%**	77.5%	**23.9%**	78.6%	**10.0%**	81.3%	**39.4%**
$T_0 = 40\%$	**71.4%**	**20.1%**	77.2%	**33.4%**	78.3%	**10.1%**	82.1%	**28.5%**
$T_0 = 50\%$	**10.0%**	**22.2%**	**76.2%**	**26.7%**	78.0%	**10.0%**	**77.3%**	**28.4%**

Table 6. CIFAR-10 Top-1 validation accuracy varying `Gaia`'s T_0 hyper-parameter. Configurations with more than 2% accuracy loss relative to BSP in the IID setting are highlighted in purple. Note that larger settings for T_0 indicate larger communication savings.

Configuration	AlexNet		GoogLeNet		LeNet		ResNet20	
	IID	Non-IID	IID	Non-IID	IID	Non-IID	IID	Non-IID
BSP	74.9%	75.0%	79.1%	78.9%	77.4%	76.6%	83.7%	**44.3%**
$Iter_{Local} = 5$	73.7%	**62.8%**	**75.8%**	**68.9%**	79.7%	**67.3%**	**73.6%**	**31.3%**
$Iter_{Local} = 10$	73.5%	**60.1%**	**76.4%**	**64.8%**	79.3%	**63.2%**	**73.4%**	**28.0%**
$Iter_{Local} = 20$	73.4%	**59.4%**	**76.3%**	**64.0%**	79.1%	**10.1%**	**73.8%**	**28.1%**
$Iter_{Local} = 50$	73.5%	**56.3%**	**75.9%**	**59.6%**	79.2%	**55.6%**	**74.0%**	**26.3%**
$Iter_{Local} = 200$	73.7%	**53.2%**	**76.8%**	**52.9%**	79.4%	**54.2%**	**75.7%**	**27.3%**
$Iter_{Local} = 500$	73.0%	**24.0%**	**76.8%**	**20.8%**	79.6%	**19.4%**	**74.1%**	**24.0%**
$Iter_{Local} = 1000$	73.4%	**23.9%**	**76.1%**	**20.9%**	78.3%	**19.0%**	**74.3%**	**22.8%**

Table 7. CIFAR-10 Top-1 validation accuracy varying `FederatedAveraging`'s $Iter_{Local}$ hyper-parameter. Configurations with more than 2% accuracy loss relative to BSP in the IID setting are highlighted in purple. Note that larger settings for $Iter_{Local}$ indicate larger communication savings.

Configuration	AlexNet		GoogLeNet		LeNet		ResNet20	
	IID	Non-IID	IID	Non-IID	IID	Non-IID	IID	Non-IID
BSP	74.9%	75.0%	79.1%	78.9%	77.4%	76.6%	83.7%	**44.3%**
$E_{warm} = 8$	75.5%	**72.3%**	78.3%	**10.0%**	80.3%	**47.2%**	**10.0%**	**10.0%**
$E_{warm} = 4$	75.5%	75.7%	79.4%	**61.6%**	**10.0%**	**47.3%**	**10.0%**	**10.0%**
$E_{warm} = 3$	75.9%	74.9%	78.9%	**75.7%**	**64.9%**	**50.5%**	**10.0%**	**10.0%**
$E_{warm} = 2$	75.7%	76.7%	79.0%	**58.7%**	**10.0%**	**47.5%**	**10.0%**	**10.0%**
$E_{warm} = 1$	75.4%	77.9%	78.6%	**74.7%**	**10.0%**	**39.9%**	**10.0%**	**10.0%**

Table 8. CIFAR-10 Top-1 validation accuracy varying `DeepGradientCompression`'s E_{warm} hyper-parameter. Configurations with more than 2% accuracy loss relative to BSP in the IID setting are highlighted in purple. Note that smaller settings for E_{warm} indicate larger communication savings.

Layer Normalization (Ba et al., 2016). Layer Normalization (LayerNorm) is a technique that is inspired by BatchNorm. Instead of computing the mean and variance of a minibatch for each *channel*, LayerNorm computes the mean and variance across all channels for each *sample*. Specifically, if the inputs are four-dimensional vectors $\mathcal{B} \times \mathcal{C} \times \mathcal{W} \times \mathcal{H}$ (batch × channel × width × height), BatchNorm produces $\mathcal{C}$ means and variances along the $\mathcal{B} \times \mathcal{W} \times \mathcal{H}$ dimensions. In contrast, LayerNorm produces $\mathcal{B}$ means and variances along the $\mathcal{C} \times \mathcal{W} \times \mathcal{H}$ dimensions (per-sample mean and variance). As the normalization is done on a per-sample basis, LayerNorm is not dependent on minibatches. However, LayerNorm makes a key assumption that all inputs make similar contributions to the final prediction, but this assumption does not hold for some models such as convolutional neural networks, where the activation of neurons should not be normalized with non-activated neurons. As a result, BatchNorm still outperforms LayerNorm for these models (Ba et al., 2016).

Batch Renormalization (Ioffe, 2017). Batch Renormalization (BatchReNorm) is an extension to BatchNorm that aims to alleviate the problem of small minibatches (or inaccurate minibatch mean, $\mu_{\mathcal{B}}$, and variance, $\sigma_{\mathcal{B}}$). BatchReNorm achieves this by incorporating the estimated global mean (μ) and variance (σ) during *training*, and introducing two hyper-parameters to contain the difference between ($\mu_{\mathcal{B}}$, $\sigma_{\mathcal{B}}$) and (μ, σ). These two hyper-parameters are gradually relaxed such that the earlier training phase is more like BatchNorm, and the later phase is more like BatchReNorm.

We evaluate BatchReNorm with BN-LeNet over CIFAR-10 to see if BatchReNorm can solve the problem of Non-IID data partitions. We replace all BatchNorm layers with BatchReNorm layers, and we carefully select the BatchReNorm hyper-parameters so that BatchReNorm achieves the highest validation accuracy in both the IID and Non-IID settings. Table 9 shows the Top-1 validation accuracy. We observe that while BatchNorm and BatchReNorm achieve similar accuracy in the IID setting, they both perform worse in the Non-IID setting. In particular, while BatchReNorm performs much better than BatchNorm in the Non-IID setting (75.3% vs. 65.4%), BatchReNorm still loses ∼3% accuracy compared to the IID setting. This is not surprising, because BatchReNorm still relies on minibatches to a certain degree, and prior work has shown that BatchReNorm's performance still degrades when the minibatch size is small (Ioffe, 2017). Hence, BatchReNorm cannot completely solve the problem of Non-IID data partitions, which is a more challenging problem than small minibatches.

BatchNorm		BatchReNorm	
IID	Non-IID	IID	Non-IID
78.8%	**65.4%**	78.1%	**75.3%**

Table 9. Top-1 validation accuracy (CIFAR-10) with BatchNorm and BatchReNorm for BN-LeNet, using BSP with $K = 2$ partitions.

J. Accuracy Loss Details

This section presents the full details of the findings summarized in §7.2. Figure 24 plots the accuracy loss between different data partitions when training GoogLeNet over CIFAR-10 with `Gaia`. Two observations are in order. First, the accuracy loss changes drastically from the IID setting (0.4% on average) to the Non-IID setting (39.6% on average). This is expected as each data partition sees very different training data in the Non-IID setting, which leads to very different models in different data partitions. Second, more conservative hyper-parameters can lead to smaller accuracy losses in the Non-IID setting. For example, the accuracy loss for $T_0 = 2\%$ is significantly smaller than those for larger settings of T_0. This is also intuitive as model divergence can be controlled by tightening communication between data partitions.

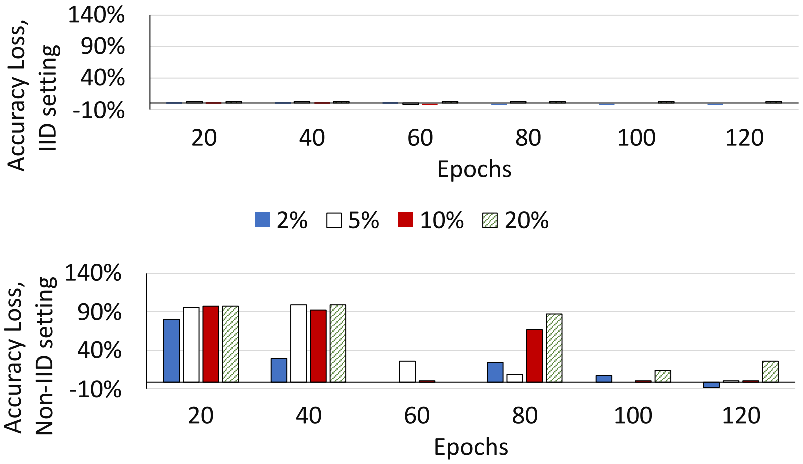


Figure 24. Training accuracy loss over time (epochs) between data partitions when training GoogLeNet over CIFAR-10 with `Gaia`. Each bar represents a T_0 for `Gaia`.

K. Discussion: Regimes of Non-IID Data

Our study has focused on *label-based* partitioning of data, in which the distribution of labels varies across partitions. In this section, we present a broader taxonomy of regimes of non-IID data, as well as various possible strategies for dealing with non-IID data, the study of which we leave to future work. We assume a general setting in which there may be many disjoint partitions, with each partition holding data collected from devices (mobile phones, video cameras, etc.) from a particular geographic region and time window.

Violations of Independence. Common ways in which data tend to deviate from being independently drawn from an overall distribution are:

- *Intra-partition correlation:* If the data within a partition are processed in an insufficiently-random order, e.g., ordered by collection device and/or by time, then independence is violated. For example, consecutive frames in a video are highly correlated, even if the camera is moving.
- *Inter-partition correlation:* Devices sharing a common feature can have correlated data across partitions. For example, neighboring geo-locations have the same diurnal effects (daylight, workday patterns), have correlated weather patterns (major storms), and can witness the same phenomena (eclipses).

Violations of Identicalness. Common ways in which data tend to deviate from being identically distributed are:

- *Quantity skew:* Different partitions can hold vastly different amounts of data. For example, some partitions may collect data from fewer devices or from devices that produce less data.
- *Label distribution skew:* Because partitions are tied to particular geo-regions, the distribution of labels varies across partitions. For example, kangaroos are only in Australia or zoos, and a person's face is only in a small number of locations worldwide. The study in this paper focused on this setting.
- *Same label, different features:* The same label can have very different "feature vectors" in different partitions, e.g., due to cultural differences, weather effects, standards of living, etc. For example, images of homes can vary dramatically around the world and items of clothing vary widely. Even within the U.S., images of parked cars in the winter will be snow-covered only in certain parts of the country. The same label can also look very different at different times, at different time scales: day vs. night, seasonal effects, natural disasters, fashion and design trends, etc.
- *Same features, different label:* Because of personal preferences, the same feature vectors in a training data item can have different labels. For example, labels that reflect sentiment or next word predictors have personal/regional biases.

As noted in some of the above examples, non-IID-ness can occur over both time (often called *concept drift*) and space (geo-location).

Strategies for dealing with non-IID data. The above taxonomy of the many regimes of non-IID data partitions naturally leads to the question of what should the objective function of the DNN model be. In our study, we have focused on obtaining a global model that minimizes an objective function over the union of all the data. An alternative objective function might instead include some notion of "fairness" among the partitions in the final accuracy on their local data (Li et al., 2020b). There could also be different strategies for treating different non-IID regimes.

As noted in Section 8, multi-task learning approaches have been proposed for jointly training local models for each partition, but a global model is essential whenever a local model is unavailable or ineffective. A hybrid approach would be to train a "base" global model that can be quickly "specialized" to local data via a modest amount of further training on local data (Yu et al., 2020). This approach would be useful for differences across space and time. For example, a global model trained under normal circumstances could be quickly adapted to natural disaster settings such as hurricanes, flash floods and forest fires.

As one proceeds down the path towards more local/specialized models, it may make sense to cluster partitions that hold similar data, with one model for each cluster (Mansour et al., 2020; Briggs et al., 2020; Laguel et al., 2020). The goal is to avoid a proliferation of too many models that must be trained, stored, and maintained over time.

Finally, another alternative for handling non-IID data partitions is to use multi-modal training that combines DNNs with key attributes about the data partition pertaining to its geo-location. A challenge with this approach is determining what the attributes should be, in order to have an accurate yet reasonably compact model (otherwise, in the extreme, the model could devolve into local models for each geo-location).