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 $\mathcal{X}_k$
 1: $u_0^k \leftarrow 0$; $v_0^k \leftarrow 0$
 2: $w_0^k \leftarrow w_0$
 3: for $t = 0, 1, 2, ...$ do
 4:     $b \leftarrow$ (sample $B$ data samples from $\mathcal{X}_k$)
 5:     $u_{t+1}^k \leftarrow m \cdot u_t^k - \eta \cdot \bigtriangledown f(w_t^k, b)$
 6:     $w_{t+1}^k \leftarrow w_t^k + u_{t+1}^k$
 7:     $v_{t+1}^k \leftarrow v_t^k + u_{t+1}^k$                         ▷ Accumulate weight updates
 8:     for $j = 0, 1, ...M$ do
 9:         $S \leftarrow ||\frac{v_{t+1}^k}{w_{t+1}^k}|| > T_t$           ▷ Check if accumulated updates are significant
10:         $\widetilde{v}_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot S$      ▷ Share significant updates with other $P_k$
11:         $v_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot \neg S$            ▷ Clear significant updates locally
12:     end for
13:     for $i = 0, 1, ...K; i \neq k$ do
14:         $w_{t+1}^k \leftarrow w_{t+1}^k + \widetilde{v}_{t+1}^i$        ▷ Apply significant updates from other $P_k$
15:     end for
16:     $T_{t+1} \leftarrow$ update_threshold($T_t$)      ▷ Decrease threshold whenever the learning rate decreases
17: 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 $\mathcal{X}_k$
 1: $u^k \leftarrow 0$
 2: for $t = 0, 1, 2, ...$ do
 3:     $w_t^k \leftarrow w_t$                                         ▷ Get the latest weights from the server
 4:     for $i = 0, ...Iter_{Local}$ do
 5:         $b \leftarrow$ (sample $B$ data samples from $\mathcal{X}_k$)
 6:         $u^k \leftarrow m \cdot u^k - \eta \cdot \bigtriangledown f(w_t^k, b)$
 7:         $w_t^k \leftarrow w_t^k + u^k$
 8:     end for
 9:     all_reduce: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{1}{K} w_t^k$       ▷ Average weights from all partitions
10: 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 $\mathcal{X}_k$
 1: $u_0^k \leftarrow 0$; $v_0^k \leftarrow 0$
 2: for $t = 0, 1, 2, ...$ do
 3:     $b \leftarrow$ (sample $B$ data samples from $\mathcal{X}_k$)
 4:     $g_{t+1}^k \leftarrow -\eta \cdot \bigtriangledown f(w_t, b)$
 5:     $g_{t+1}^k \leftarrow$ gradient_clipping($g_{t+1}^k$)                ▷ Clip gradients
 6:     $u_{t+1}^k \leftarrow m \cdot u_t^k + g_{t+1}^k$
 7:     $v_{t+1}^k \leftarrow v_t^k + u_{t+1}^k$                              ▷ Accumulate weight updates
 8:     $T \leftarrow s\%$ of $||v_{t+1}^k||$                                 ▷ Determine the threshold for sparsified updates
 9:     for $j = 0, 1, ...M$ do
10:         $S \leftarrow ||v_{t+1}^k|| > T$                                  ▷ Check if accumulated updates are top $s\%$
11:         $\widetilde{v}_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot S$        ▷ Share top updates with other $P_k$
12:         $v_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot \neg S$               ▷ Clear top updates locally
13:         $u_{t+1}^k[j] \leftarrow u_{t+1}^k[j] \odot \neg S$               ▷ Clear the history of top updates (momentum correction)
14:     end for
15:     $w_{t+1} = w_t + \sum_{k=1}^{K} \widetilde{v}_{t+1}^k$                ▷ Apply top updates from all $P_k$
16: 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.

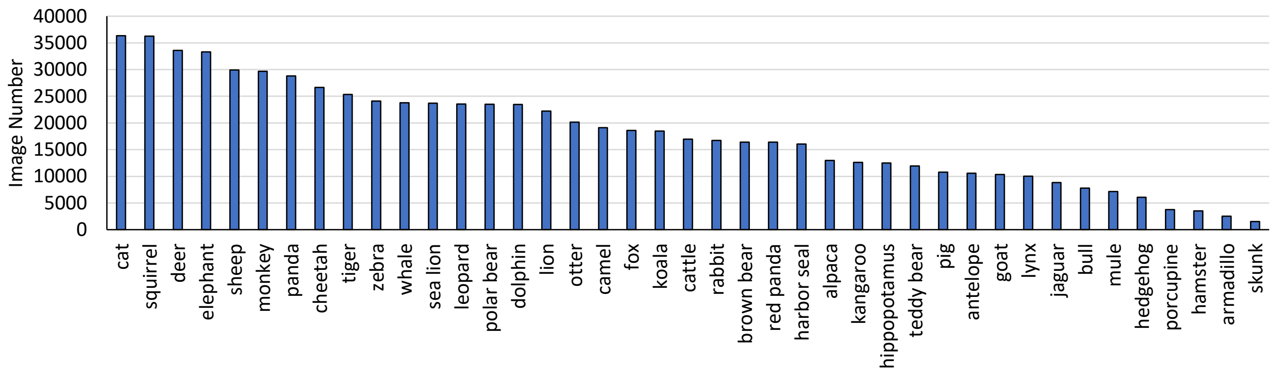


Figure 9. Flickr-Mammal dataset: The number of images in each mammal class.

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.

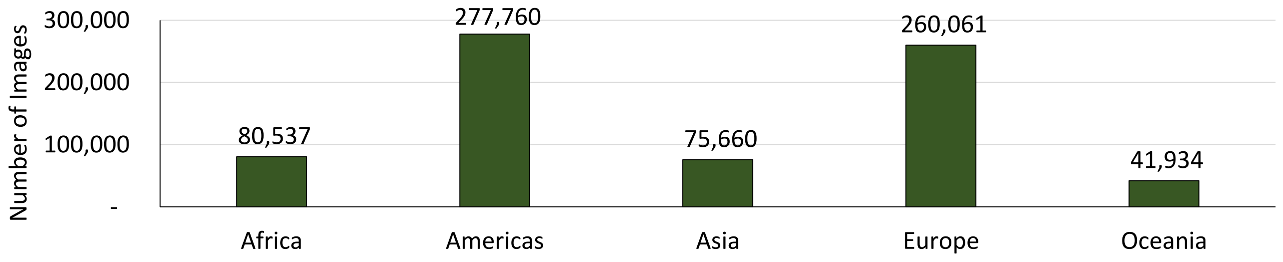


Figure 10. Flickr-Mammal dataset: The number of images in each continent.

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.

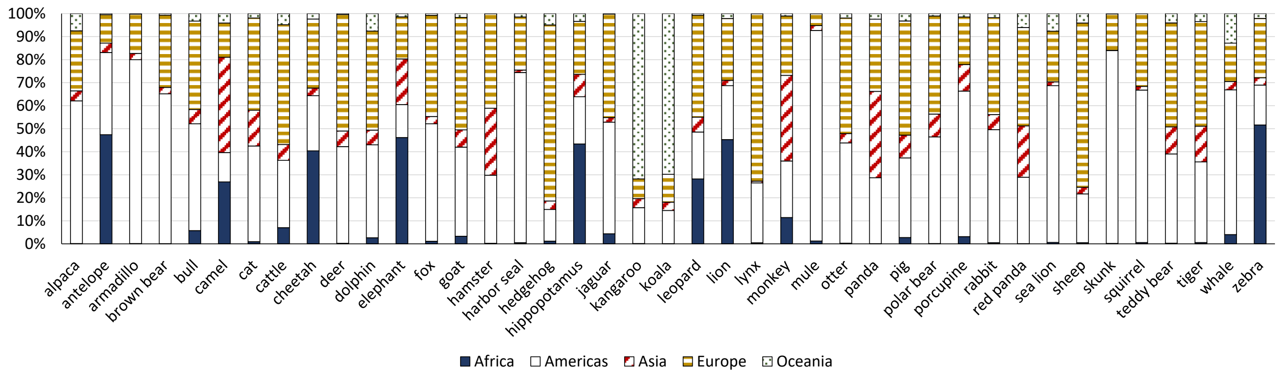


Figure 11. Flickr-Mammal dataset: The share of images in each continent based on raw samples.

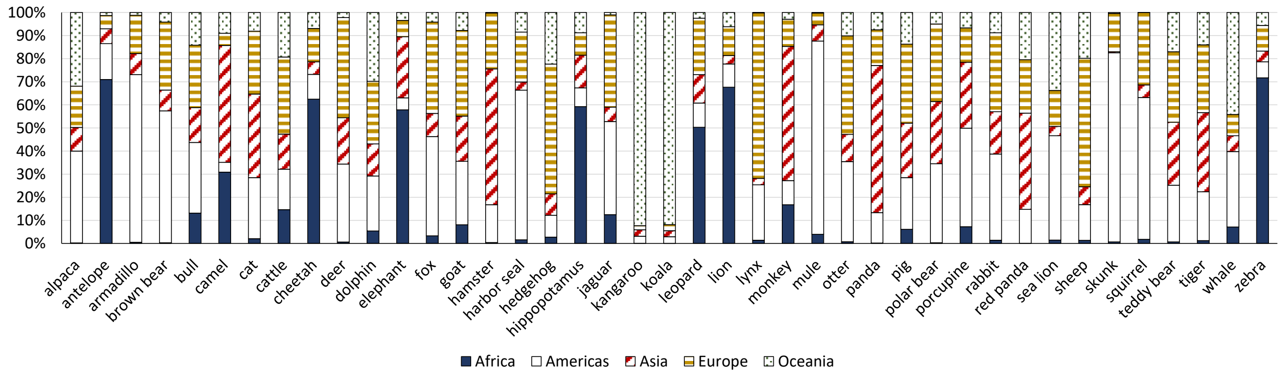


Figure 12. Flickr-Mammal dataset: The share of images in each continent based on normalized samples.

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.

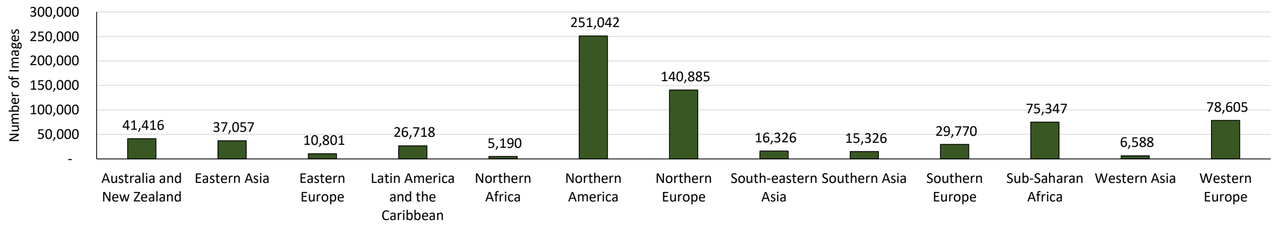


Figure 13. Flickr-Mammal dataset: The number of images in each subcontinent.

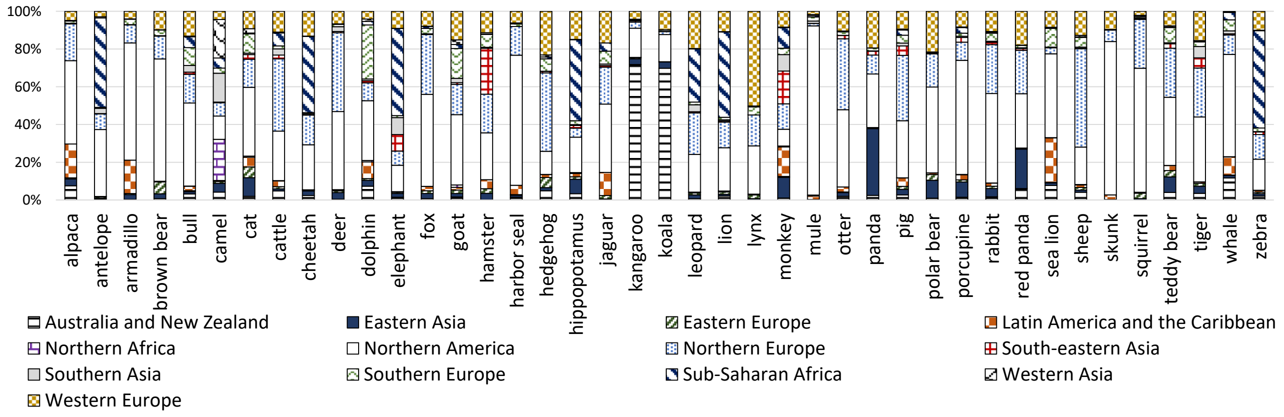


Figure 14. Flickr-Mammal dataset: The share of images in each subcontinent based on raw samples.

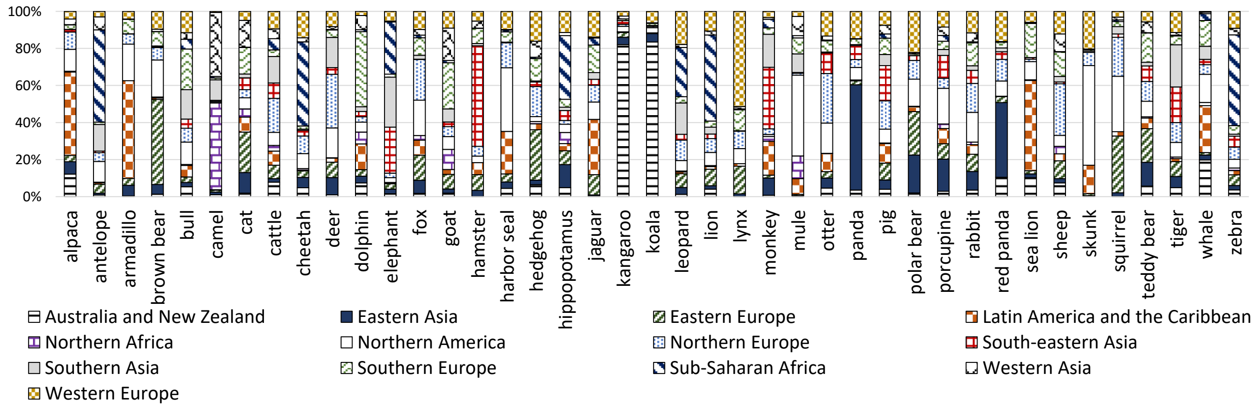


Figure 15. Flickr-Mammal dataset: The share of images in each subcontinent based on normalized samples.

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

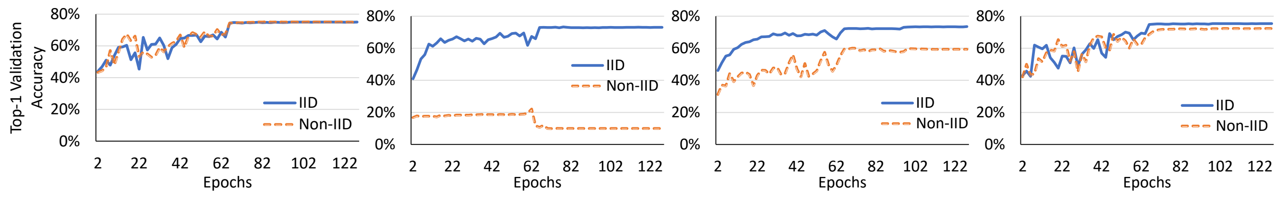
Table 4. Major training parameters for FACE RECOGNITION over CASIA-WebFace.

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.

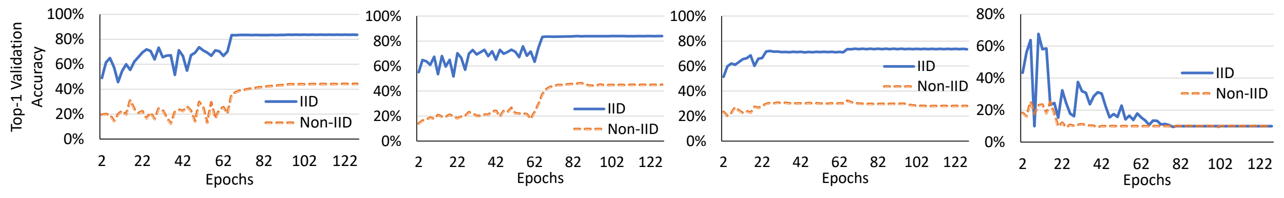
D. Training Convergence Curves

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.

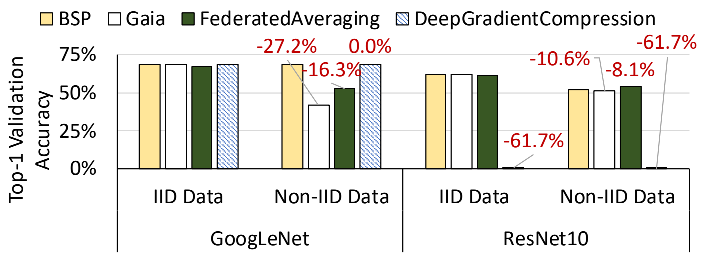


Figure 18. Top-1 validation accuracy for IMAGE CLASSIFICATION over the ImageNet dataset. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting.

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.

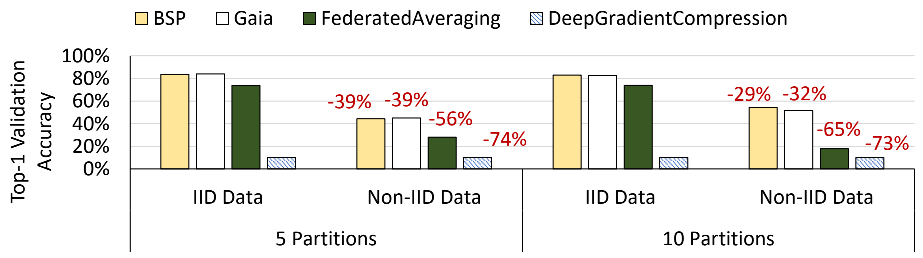


Figure 19. Top-1 validation accuracy for ResNet20 over the CIFAR-10 dataset, with 5 and 10 data partitions. The five partition results are repeated from Figure 1. Each "-x%" label indicates the accuracy loss relative to BSP in the IID setting.

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.

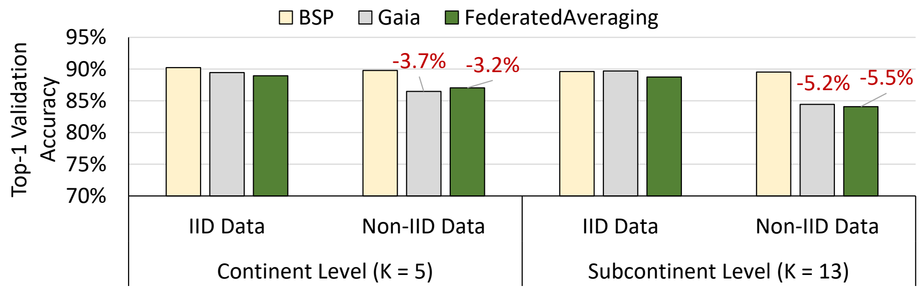


Figure 20. Top-1 validation accuracy for GoogLeNet over the Flickr-Mammal dataset, which is partitioned at the continent level and the subcontinent level. 5% of data are randomly selected as the validation set. The five partition results are repeated from Figure 2. Non-IID Data is based on real-world 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.

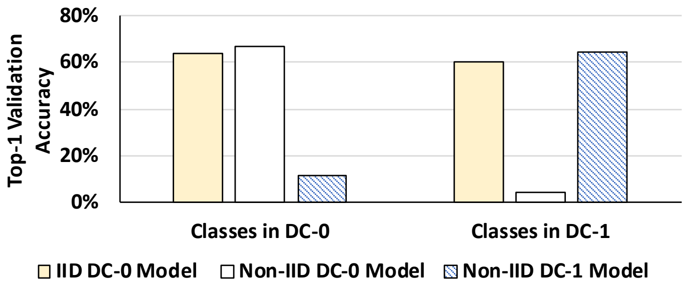


Figure 21. Top-1 validation accuracy (ImageNet) for models in different partitions.

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.

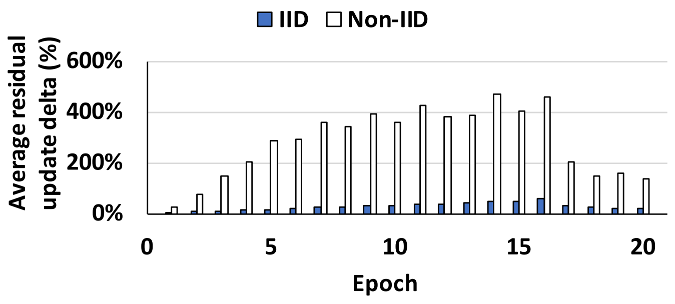


Figure 22. Average residual update delta (%) for `DeepGradientCompression` over the first 20 epochs.

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`.

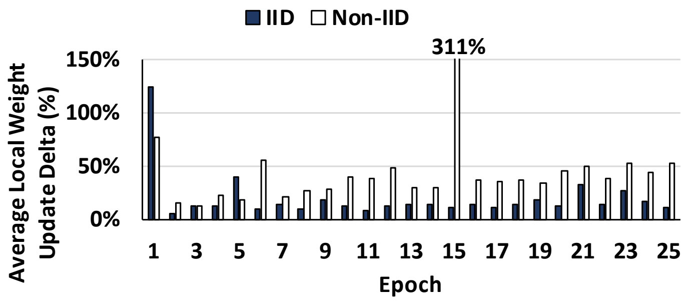


Figure 23. Average local update delta (%) for `FederatedAveraging` over the first 25 epochs.

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 $\times$ channel $\times$ width $\times$ 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.

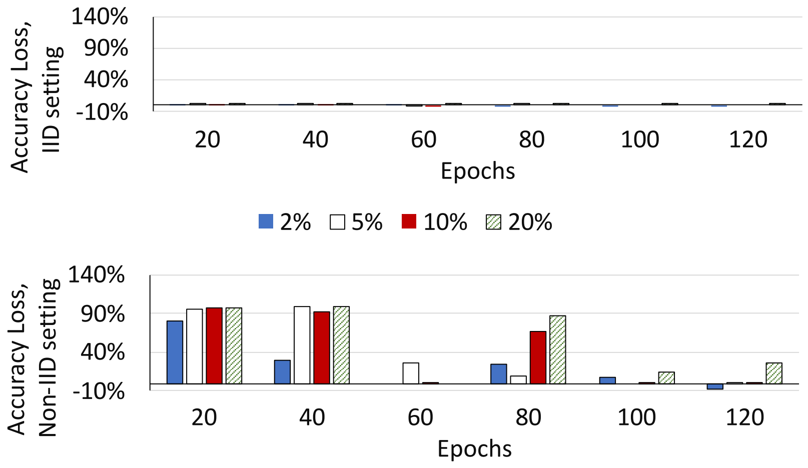


Figure 24. Training accuracy loss over time (epochs) between data partitions when training GoogLeNet over CIFAR-10 with `Gaia`. Each bar represents a T_0 for `Gaia`.

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).