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

\begin{align*}
\text{Input:} & \quad \text{initial weights } w_0 = \{w_0[0], ..., w_0[M]\} \\
\text{Input:} & \quad K \text{ data partitions (or data centers); initial significance threshold } T_0 \\
\text{Input:} & \quad \text{local minibatch size } B; \text{ momentum } m; \text{ learning rate } \eta; \text{ local dataset } X_k \\
1: & \quad u_0^k \leftarrow 0; v_0^k \leftarrow 0 \\
2: & \quad w_0^k \leftarrow w_0 \\
3: & \quad \text{for } t = 0, 1, 2, \ldots \text{ do} \\
4: & \quad \quad b \leftarrow \text{sample } B \text{ data samples from } X_k \\
5: & \quad \quad w_{t+1}^k \leftarrow m \cdot w_t^k - \eta \cdot \nabla f(w_t^k, b) \\
6: & \quad \quad w_{t+1}^k \leftarrow w_t^k + w_{t+1}^k \\
7: & \quad \quad v_{t+1}^k \leftarrow v_t^k + u_{t+1}^k \\
8: & \quad \quad \text{for } j = 0, 1, \ldots, M \text{ do} \\
9: & \quad \quad \quad S \leftarrow ||v_{t+1}^k[w_{t+1}^k]|| > T_t \\
10: & \quad \quad \quad \tilde{v}_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot S \\
11: & \quad \quad \quad v_{t+1}^k[j] \leftarrow v_{t+1}^k[j] \odot \neg S \\
12: & \quad \quad \text{end for} \\
13: & \quad \quad \text{for } i = 0, 1, K; i \neq k \text{ do} \\
14: & \quad \quad \quad w_{t+1}^k \leftarrow w_{t+1}^k + \tilde{v}_{t+1}^i \\
15: & \quad \quad \text{end for} \\
16: & \quad T_{t+1} \leftarrow \text{update_threshold}(T_t) \\
17: & \quad \text{end for}
\end{align*}

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

\begin{align*}
\text{Input:} & \quad \text{initial weights } w_0; K \text{ data partitions (or clients)} \\
\text{Input:} & \quad \text{local minibatch size } B; \text{ local iteration number } \text{Iter}_{\text{Local}} \\
\text{Input:} & \quad \text{momentum } m; \text{ learning rate } \eta; \text{ local dataset } X_k \\
1: & \quad u^k \leftarrow 0 \\
2: & \quad \text{for } t = 0, 1, 2, \ldots \text{ do} \\
3: & \quad \quad w_t^k \leftarrow w_t \\
4: & \quad \quad \text{for } i = 0, \ldots, \text{Iter}_{\text{Local}} \text{ do} \\
5: & \quad \quad \quad b \leftarrow \text{sample } B \text{ data samples from } X_k \\
6: & \quad \quad \quad u^k \leftarrow m \cdot u^k - \eta \cdot \nabla f(w_t^k, b) \\
7: & \quad \quad \quad w_t^k \leftarrow u_t^k + w_t^k \\
8: & \quad \quad \text{end for} \\
9: & \quad \text{all_reduce: } w_{t+1} \leftarrow \sum_{k=1}^K \frac{1}{K} w_t^k \\
10: & \quad \text{end for}
\end{align*}
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 around 1,200 images for each class.

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. $w_0^k \leftarrow 0$; $\frac{\partial w_0^k}{\partial w_0^k} \leftarrow 0$
2. for $t = 0, 1, 2, ...$ do
3. 
4. 
5. 
6. 
7. 
8. 
9. for $j = 0, 1, ..., M$ do
10. 
11. 
12. 
13. 
14. end for
15. $w_{t+1} = w_t + \sum_{k=1}^{K} \tilde{v}_t^{k+1}$
16. end for

$\triangleright$ Gradient clipping

$\triangleright$ Accumulate weight updates

$\triangleright$ Determine the threshold for sparsified updates

$\triangleright$ Check if accumulated updates are top $s\%$

$\triangleright$ Share top updates with other $P_k$

$\triangleright$ Clear top updates locally

$\triangleright$ Clear the history of top updates (momentum correction)

$\triangleright$ Apply top updates from all $P_k$

**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.](image-url)
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. As Figure 10 shows, 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 take pictures and use Flickr.

![Figure 10. Flickr-Mammal dataset: The number of images in each continent.](image)

**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 only has 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 very few image samples for these mammal classes.

![Figure 11. Flickr-Mammal dataset: The share of images in each continent based on raw samples.](image)

![Figure 12. Flickr-Mammal dataset: The share of images in each continent based on normalized samples.](image)
Share of normalized samples across continents. As we are mostly interested in the distribution of labels ($P(y)$) 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 this normalized samples. Here, Figure 12 illustrates the normalized sample share for all mammals across continent. As we see, the overall label distribution is similar between normalized (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). 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.

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, Elephant and Hippopotamus are mostly in Sub-Saharan Africa, and Mule and Skunk are mostly in Northern America. On average, 5 of the 13 subcontinents consist of less than 1% of images for each mammal class. We also show the normalized sample share across subcontinents (Figure 15), and we can see the difference of $P(y)$ among subcontinents. Overall, our analysis shows that skew 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.
The Non-IID Data Quagmire of Decentralized Machine Learning

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

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Minibatch size per node (8 nodes)</th>
<th>Momentum</th>
<th>Weight decay</th>
<th>Learning rate</th>
<th>Total epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet</td>
<td>32</td>
<td>0.9</td>
<td>0.0002</td>
<td>$\eta_0 = 0.0025$, polynomial decay, power = 0.5</td>
<td>60</td>
</tr>
<tr>
<td>ResNet-10</td>
<td>32</td>
<td>0.9</td>
<td>0.00125</td>
<td>$\eta_0 = 0.00125$, polynomial decay, power = 1</td>
<td>64</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Minibatch size per node (4 nodes)</th>
<th>Momentum</th>
<th>Weight decay</th>
<th>Learning rate</th>
<th>Total epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>center-loss</td>
<td>64</td>
<td>0.9</td>
<td>0.0005</td>
<td>$\eta_0 = 0.025$, divides by 10 at epoch 4 and 6</td>
<td>7</td>
</tr>
</tbody>
</table>

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

D. 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 gets 500 image classes. According to the hyper-parameter criteria in §3, we select $T_0 = 40\%$ for Gaia, $\text{Iter}_{\text{Local}} = 200$ for FederatedAveraging, and $E_{\text{warm}} = 4$ for DeepGradientCompression.
The Non-IID Data Quagmire of Decentralized Machine Learning

<table>
<thead>
<tr>
<th>Model</th>
<th>Minibatch size per node (5 nodes)</th>
<th>Momentum</th>
<th>Weight decay</th>
<th>Learning rate</th>
<th>Total epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet</td>
<td>32</td>
<td>0.9</td>
<td>0.0002</td>
<td>$\eta_0 = 0.004$, polynomial decay, power = 0.5</td>
<td>55</td>
</tr>
</tbody>
</table>

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

![Figure 16. Top-1 validation accuracy for IMAGE CLASSIFICATION over the ImageNet dataset. Each “-x%” label indicates the accuracy loss from BSP in the IID setting.](image)

The same trend in different datasets. Figure 16 illustrates the validation accuracy in the IID and Non-IID settings. Interestingly, we observe the same problems in the ImageNet dataset as in the CIFAR-10 dataset (§4.1), even though the number of ImageNet classes is two orders of magnitude more than the number of CIFAR-10 classes. 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 not only exists in various decentralized learning algorithms and DNNs, but also exists in different datasets.

E. Reasons for Model Quality Loss

Gaia. We extract the Gaia-trained models from both partitions (denoted DC-0 and DC-1) for IMAGE CLASSIFICATION over the ImageNet dataset, and then evaluate the validation accuracy of each model based on the image classes in each partition. As Figure 17 shows, the validation accuracy is pretty consistent among 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 partition, but they perform very poorly for the image classes that are not in their respective partition. 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 data partition.

This raises the following question: How does Gaia produce completely different models in the Non-IID setting, given that Gaia synchronizes all significant updates ($\Delta w_j$) 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 that the threshold for significance in the last epoch was 1%). However, we find that given the same input image, the neuron values are vastly different (at an average difference of 173%). This finding suggests that small model differences can result in completely different models. Mathematically, this is because weights are both positive and negative: a small percentage difference in individual weights of a neuron can lead to a large percentage difference in its value. 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.

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 look at 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 $P_k$, due to its communicating 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 18 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 the training diverges after that 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 show DeepGradientCompression often diverging.

**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 ($||\Delta w_i/w_i||$, where $w_i$ is the averaged global model weight). Figure 19 depicts this number for the first 25 training epochs when training AlexNet over the CIFAR-10 dataset. 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.

**F. 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 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 lose significant accuracy 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
The Non-IID Data Quagmire of Decentralized Machine Learning

**Figure 19.** Average local update delta (%) for FederatedAveraging over the first 25 epochs.

FederatedAveraging, the most communication-intensive of the choices shown, we still see 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 suggests that we may be able to use more frequent communication among the $P_k$ for higher model quality in the Non-IID setting (mitigating the “tug-of-war” among the $P_k$ (§2.1)).

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AlexNet</th>
<th>GoogLeNet</th>
<th>LeNet</th>
<th>ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IID</td>
<td>Non-IID</td>
<td>IID</td>
<td>Non-IID</td>
</tr>
<tr>
<td>BSP</td>
<td>74.9%</td>
<td>75.0%</td>
<td>79.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td>$T_0 = 2%$</td>
<td>73.8%</td>
<td>70.5%</td>
<td>78.4%</td>
<td>56.5%</td>
</tr>
<tr>
<td>$T_0 = 5%$</td>
<td>73.2%</td>
<td>71.4%</td>
<td>77.6%</td>
<td>75.6%</td>
</tr>
<tr>
<td>$T_0 = 10%$</td>
<td>73.0%</td>
<td>10.0%</td>
<td>78.4%</td>
<td>68.0%</td>
</tr>
<tr>
<td>$T_0 = 20%$</td>
<td>72.5%</td>
<td>37.6%</td>
<td>77.7%</td>
<td>67.0%</td>
</tr>
<tr>
<td>$T_0 = 30%$</td>
<td>72.4%</td>
<td>26.0%</td>
<td>77.5%</td>
<td>23.9%</td>
</tr>
<tr>
<td>$T_0 = 40%$</td>
<td>71.4%</td>
<td>20.1%</td>
<td>77.2%</td>
<td>33.4%</td>
</tr>
<tr>
<td>$T_0 = 50%$</td>
<td>10.0%</td>
<td>22.2%</td>
<td>76.2%</td>
<td>26.7%</td>
</tr>
</tbody>
</table>

Table 6. Top-1 validation accuracy (CIFAR-10) varying Gaia’s $T_0$ hyper-parameter. The configurations with more than 2% accuracy loss from BSP in the IID setting are highlighted. Note that larger settings for $T_0$ mean significantly greater communication savings.

G. More Alternatives to Batch Normalization

**Weight Normalization (Salimans & Kingma, 2016).** Weight Normalization (WeightNorm) is a normalization scheme that normalizes the weights in a DNN as oppose to the neurons (which is what BatchNorm and most other normalization techniques do). WeightNorm is not dependent on minibatches as it is normalizing 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 $\mu_B$ among the $P_k$ in the Non-IID setting ($\S$5.1).

**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 $B \times C \times W \times H$ (batch \times channel \times width \times height), LayerNorm computes the mean and variance across all channels of each sample. This makes LayerNorm invariant to the number of samples in a minibatch, which can be beneficial in the Non-IID setting where the number of samples in each group may vary significantly.
The Non-IID Data Quagmire of Decentralized Machine Learning

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AlexNet</th>
<th>GoogLeNet</th>
<th>LeNet</th>
<th>ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IID</td>
<td>Non-IID</td>
<td>IID</td>
<td>Non-IID</td>
</tr>
<tr>
<td>BSP</td>
<td>74.9%</td>
<td>75.0%</td>
<td>79.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 5$</td>
<td>73.7%</td>
<td>62.8%</td>
<td>75.8%</td>
<td>68.9%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 10$</td>
<td>73.5%</td>
<td>60.1%</td>
<td>76.4%</td>
<td>64.8%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 20$</td>
<td>73.4%</td>
<td>59.4%</td>
<td>76.3%</td>
<td>64.0%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 50$</td>
<td>73.5%</td>
<td>56.3%</td>
<td>75.9%</td>
<td>59.6%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 200$</td>
<td>73.4%</td>
<td>53.2%</td>
<td>76.8%</td>
<td>52.9%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 500$</td>
<td>73.0%</td>
<td>24.0%</td>
<td>76.8%</td>
<td>20.8%</td>
</tr>
<tr>
<td>$\text{Iter}_{Local} = 1000$</td>
<td>73.4%</td>
<td>23.9%</td>
<td>76.1%</td>
<td>20.9%</td>
</tr>
</tbody>
</table>

Table 7. CIFAR-10 Top-1 validation accuracy with various FederatedAveraging hyper-parameters. The configurations that lose more than 2% accuracy are highlighted. Note that larger settings for $\text{Iter}_{Local}$ mean significantly greater communication savings.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AlexNet</th>
<th>GoogLeNet</th>
<th>LeNet</th>
<th>ResNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IID</td>
<td>Non-IID</td>
<td>IID</td>
<td>Non-IID</td>
</tr>
<tr>
<td>BSP</td>
<td>74.9%</td>
<td>75.0%</td>
<td>79.1%</td>
<td>78.9%</td>
</tr>
<tr>
<td>$E_{\text{warm}} = 8$</td>
<td>75.5%</td>
<td>72.3%</td>
<td>78.3%</td>
<td>10.0%</td>
</tr>
<tr>
<td>$E_{\text{warm}} = 4$</td>
<td>75.5%</td>
<td>75.7%</td>
<td>79.4%</td>
<td>61.6%</td>
</tr>
<tr>
<td>$E_{\text{warm}} = 3$</td>
<td>75.9%</td>
<td>74.9%</td>
<td>78.9%</td>
<td>75.7%</td>
</tr>
<tr>
<td>$E_{\text{warm}} = 2$</td>
<td>75.7%</td>
<td>76.7%</td>
<td>79.0%</td>
<td>58.7%</td>
</tr>
<tr>
<td>$E_{\text{warm}} = 1$</td>
<td>75.4%</td>
<td>77.9%</td>
<td>78.6%</td>
<td>74.7%</td>
</tr>
</tbody>
</table>

Table 8. CIFAR-10 Top-1 validation accuracy with various DeepGradientCompression hyper-parameters. The configurations that lose more than 2% accuracy are highlighted. Note that smaller settings for $E_{\text{warm}}$ mean significantly greater communication savings.

channel $\times$ width $\times$ height), BatchNorm produces $C$ means and variances along the $B \times W \times H$ dimensions. On the other hand, LayerNorm produces $B$ means and variances along the $C \times W \times H$ 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, $\mu_B$, and variance, $\sigma_B$). BatchReNorm achieves this by incorporating the estimated global mean ($\mu$) and variance ($\sigma$) during training, and introducing two hyper-parameters to contain the difference between ($\mu_B$, $\sigma_B$) and ($\mu$, $\sigma$). 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 see 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 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.
The Non-IID Data Quagmire of Decentralized Machine Learning

<table>
<thead>
<tr>
<th></th>
<th>BatchNorm</th>
<th>BatchReNorm</th>
</tr>
</thead>
<tbody>
<tr>
<td>IID</td>
<td>78.8%</td>
<td>78.1%</td>
</tr>
<tr>
<td>Non-IID</td>
<td>65.4%</td>
<td>75.3%</td>
</tr>
</tbody>
</table>

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

H. Accuracy Loss Details

Figure 20 plots the accuracy loss between different data partitions when training GoogleNet over CIFAR-10 with Gaia. Two observations are in order. First, the accuracy drop 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 drop in the Non-IID setting. For example, the accuracy drop for $T_0 = 2\%$ is significantly smaller than for larger settings of $T_0$. This is also intuitive as model divergence can be controlled by tightening communication between data partitions.

![Training accuracy drop over time (epochs) between data partitions when training GoogleNet over CIFAR-10 with Gaia. Each bar represents a $T_0$ for Gaia](image)

I. 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 are left 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:
The Non-IID Data Quagmire of Decentralized Machine Learning

- **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 few 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 that 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).