Hewlett Packard Enterprise

Technical white paper

Merge Methods in Swarm Learning

A hyper-parameter for the swarm training.

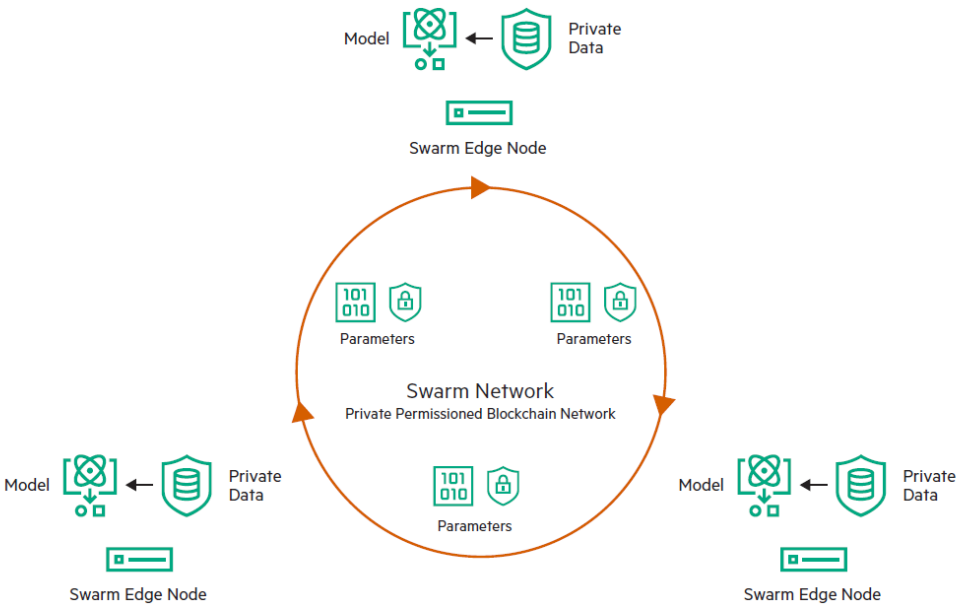
CONTENTS

EXECUTIVE SUMMARY

Swarm Learning is a cross-silo federated learning framework, where overall model training happens with the merge of model parameters obtained from the data of different swarm participants. To ensure overall learning, the model parameters from each participant are collected and merged by the swarm container to arrive at the final swarm model. The mechanism with which the merging happens is called the swarm merge process and the method used in the merge process is called the swarm merge method. In the recent advancements of swarm learning, users can now merge the model parameters using three different merge methods. Those are mean, coordinate-wise median, and geometric-wise median. These methods use statistical techniques to merge the model parameters. Any merge method in swarm training is expected to merge model parameters effectively so that model convergence is reached. The provision of choosing the merge method is available through the swarm callback method defined in the swarm learning wheel package. Just like how minPeers is passed to the swarm training, in the same way user can pass the merge method as a string to the 'mergeMethod' parameter. Once the swarm training starts, for each sync round, the elected leader node performs the swarm merge based on the merge option that was passed within the callback.

SWARM MERGE

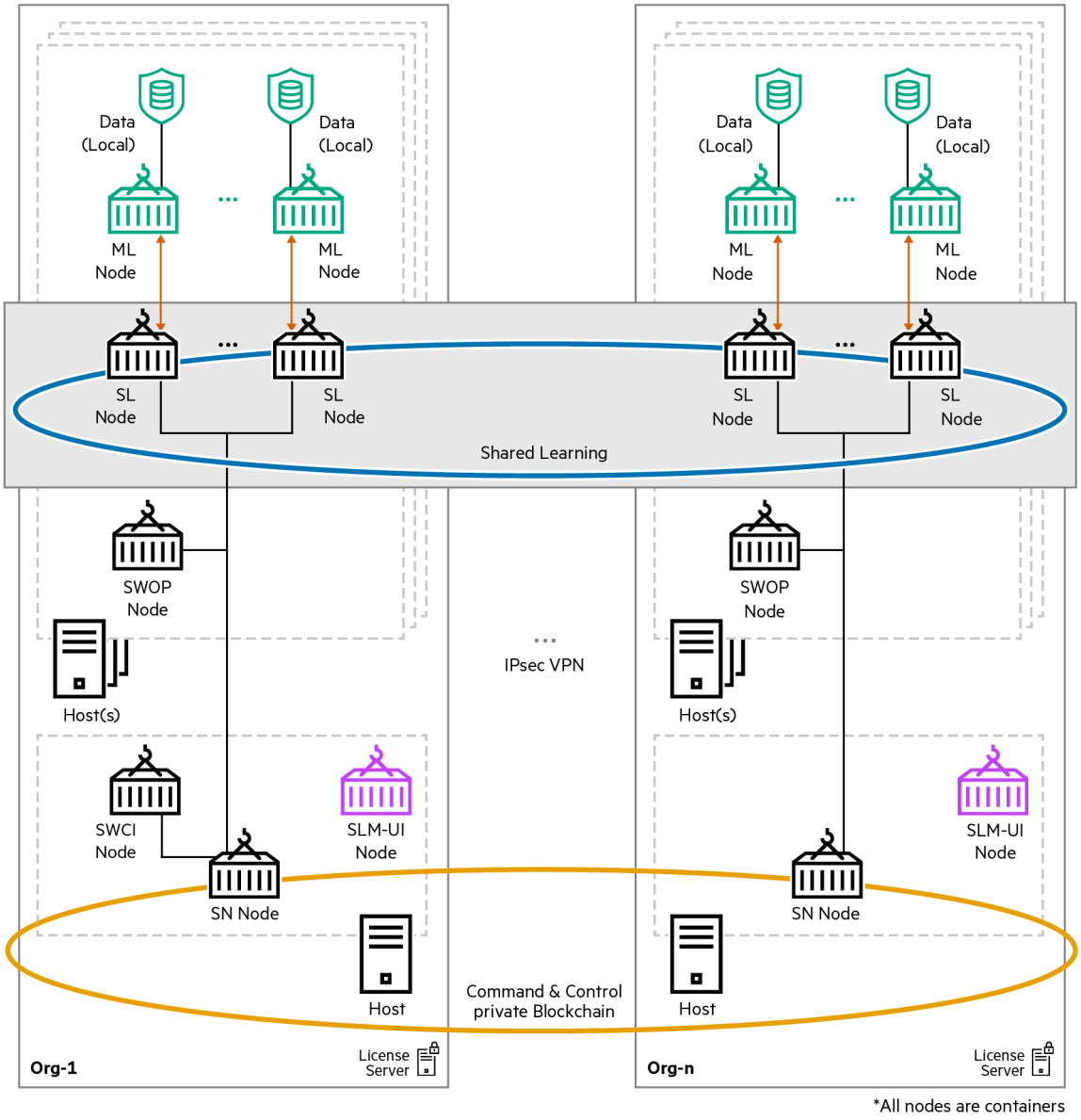
In Swarm Learning, training of the model occurs at the edge, where data is most recent, and where prompt, data-driven decisions are mostly necessary. In this swarm learning's decentralized architecture, only the insights learned are shared with the collaborating ML peers, not the raw data. Only the model parameters (learnings) are exchanged between the various edge nodes and not the raw data. This ensures that the privacy of data is preserved. The following image provides an overview of the Swarm Learning framework.



Swarm Learning node

Swarm Learning is made up of various components known as nodes, such as Swarm Learning (SL) nodes, Swarm Network (SN) nodes, Swarm Learning Command Interface (SWCI) nodes, and Swarm Operator (SWOP) nodes. The merge process is entirely carried out by the SL nodes. Hence, it is better to understand the details about the SL node in the swarm.

Each node of SL is modularized and runs in a separate container. SL nodes run the core of Swarm Learning. An SL node works in collaboration with all the other SL nodes in the network. It regularly shares its learnings with the other nodes and incorporates their insights. These SL nodes act as an interface between the user model application and other Swarm Learning components. These nodes are also responsible for distributing and merging model weights. The learnings from each SL node are shared and hence it is called shared learning.



Swarm Learning leader node

Swarm performs the merge of the model parameters at specific intervals of swarm training. The frequency at which the swarm merge happens is called sync frequency. At the end of every sync frequency, when it is time to share the learning from the individual model, one of the SL nodes is designated as “leader”. This leader SL node collects the individual models from each peer node and merges them into a single model by combining the model parameters of all the individuals. Once the training starts, all SL nodes become aware of the type of merge method passed into its swarm callback. Using this merge method, the SL leader node carries out the entire swarm merge process. This leader node is also responsible for retrieving the learning files from all individual SL follower nodes.

Swarm Learning callback

SwarmCallback is a custom callback class that consists of a set of functions that can be applied at various stages of the training process. During training, the SwarmCallback and the set of functions provide a view of internal states and statistics of the model to the Swarm Learning framework. It also executes training-specific operations like saving model parameters at the end of a sync interval.

```
SwarmCallback(syncFrequency=128,
              minPeers=4,
              mergeMethod='mean')
```

The ‘mergeMethod’ is one of the optional swarm parameters to the SwarmCallback method. This parameter is of type string. The user can pass one of the strings shown in the table as a mergeMethod parameter to choose the type of merge that Swarm picks for merging model parameters. The default value of this merge method parameter is ‘mean’.

SNo	Merge Method Name	Description
1	mean	Weighted Mean
2	coordmedian	Weighted Coordinate Median
3	geomedian	Weighted Geometric Median

This mergeMethod value traverses from SwarmCallback to Swarm blackboard. Swarm Blackboard manages check-in, check-out, and merge kind of operations in SL nodes.

Merge process

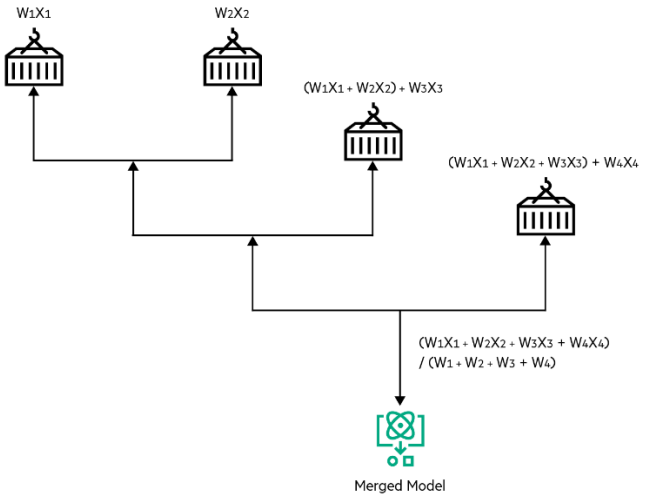
Merge is one of the core operations in Swarm Learning that ensures learning is shared across SL nodes. This merge operation happens several times within the Swarm training. It depends upon the sync frequency and the total number of epochs chosen in the ML model. The model parameters that were trained at intermediate sync rounds are saved in the respective SL nodes. For each sync round, the Swarm Learning framework chooses one of the SL nodes as a leader. This SL leader is responsible for pulling the intermediate trained models from all other SL nodes and then performs Swarm merge operation. This operation is based on the type of merge method that we pass in the SwarmCallback API. Once the merge completes, the aggregated swarm model is saved in the SL leader node. All SL nodes that have participated in the current sync round pull the aggregated model from the leader. ML nodes use this aggregated model as input to continue the training of the next sync round. This process gets repeated until the Swarm training is completed.

SWARM MERGE METHODS

Swarm Learning provides three different merge options via Swarm callback. The merge options are 'mean', 'coordmedian' and 'geomedian'. All these three merge methods will also consider the weightages of each SL node while merging the model parameters. If the weightages of all SL nodes participating in the merge round are 1, then the merge *is a regular mean, coordinate median or geometric median. These merge methods implementations can handle the weightages associated with individual SL nodes.

Weighted mean (Weighted federative averaging)

The 'mean' option merges intermediate model parameters using a weighted mean method. This weighted mean method is also known as weighted federative averaging. This method sums up all the intermediate model parameters in an iterative way. In this iterative approach, the SL leader collects the weightages and intermediate trained models of 2 nodes at a time and sums it up. The previously calculated weighted sum is then added to subsequent nodes and so on. Once the weighted sum of the model parameters is done then it gets divided by the total sum of weightages. If the weightages of all nodes are 1 then, the total sum of weightages is equal to the total number of SL nodes that participated in the current merge/sync round. The following figure depicts the technique behind the mean merge method. Here, W indicates the weightage of the SL nodes and X indicates the intermediate models.



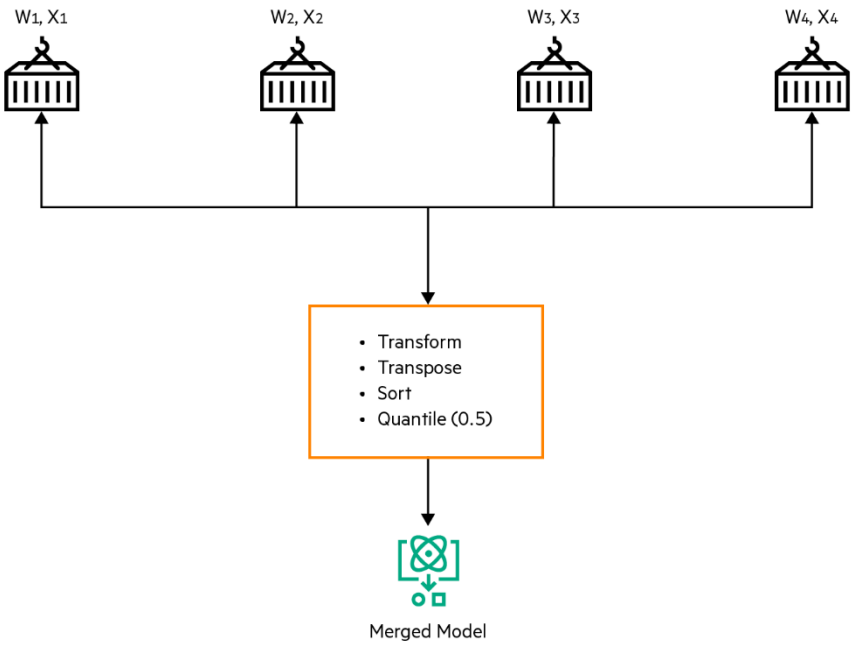
The mean merge method is memory-optimized and is an efficient technique among all merge methods. It takes less space in the memory to perform a merge operation. Only, two node's model parameters are in memory for any

given time even if there are N number of SL nodes. For example, If the intermediate model of SL node is 250MB each, then for a given merge it takes around 500MB at a time to complete the merge process. It is also a generic statistical solution that suits most of the cases. Because of these advantages, it is marked as the default merge method in swarm learning.

Weighted coordinate-wise median

The 'coordmedian' option finds the weighted coordinate-wise median of the intermediate model parameters. It is also known as the component-wise median and is a statistical measure that calculates the 50th quantile (median) of the input data. This method was invented by F. Y. Edgeworth. It is useful as an estimator of central tendency and is robust against outliers. The coordinate median is a central value of the sorted data. If the number of data points is odd and the weightages of each data point are one, then the coordinate median is the exact central data point of the sorted data. If the number of data points is even and the weightages of each data point are one, then the coordinate median is the average of the central two data points.

Let us understand the coordinate median merge process from the following figure. Here, we assume there are four SL nodes participating in the merge round. For each sync round, the SL leader collects the weightages (W_1 to W_4) and intermediate trained models (X_1 to X_4, aka weights and biases). These parameters from all four SL nodes are collected in the leader SL node. Due to the nature of ML models as multi-dimensional matrices, we cannot directly apply quantile function to it. Hence, these model parameters get transformed and transposed to a desired shape so that any statistical techniques can be applied to it. Later, the quantile function is applied to the processed model parameters where sort and 50th percentile (or 0.5 Quantile) calculation happens. Then, the coordinate median based merged model is checked out by the other SL nodes to continue with the next sync round.

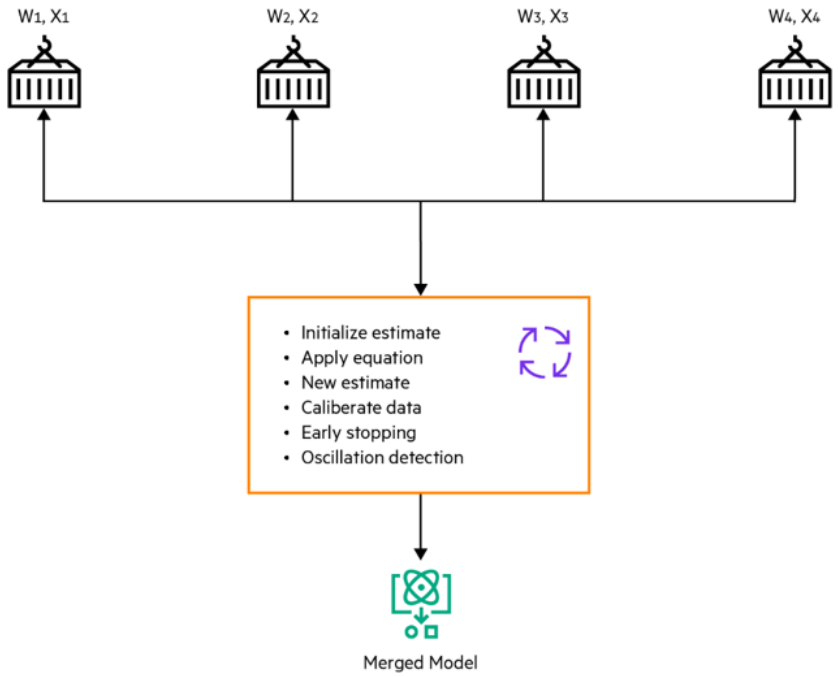


In this coordinate-wise median method, the combined intermediate model parameters (either by layers or as a whole) have to be available in run-time memory to apply any transformational techniques. Hence, it takes a lot of memory especially when the model parameters size is big. The optimization of this problem for coordinate and geometric median is discussed in the subsequent sections of this document. This merge method involves operations like transform, transpose, and sort, it takes time to generate a merged model when compared to the mean method. However, there are a few advantages when compared to the mean. The comparison between the merge methods is discussed in detail in the later part of this document.

Weighted geometric-wise median

The 'geomedian' option finds the weighted geometric median of the intermediate model parameters. It is the value that is minimal in distance to all the data points. There is no fixed way to find a geometric median and multiple algorithms exist to find it. In Swarm, Weiszfeld's algorithm is used to find the geometric median. This is an iterative estimator technique. This approach needs an initial estimator, and the algorithm finds a new estimated value for the

next round. This process is repeated N number of times or until it reaches convergence. In Swarm, the average of all model parameters is chosen as an initial estimate for faster convergence.



$$y_{k+1} = \left(\sum_{i=1}^{m} \frac{x_i}{\|x_i - y_k\|}\right) \Big/ \left(\sum_{i=1}^{m} \frac{1}{\|x_i - y_k\|}\right)$$

For each iteration, this merge method captures the delta (a difference between the old estimate and the new estimate) and maintains the latest three iterations' results. These metrics help the geometric median method to bail out from the iterations if one of the following criteria is met.

1) Minimal threshold of delta is reached.
2) Early stopping when convergence is achieved.
3) Oscillation in median is detected.

As per this method, the minimum threshold of error that can be acceptable between the old and new estimates is 1e-6. This means the geometric median merge method bails out with a new estimate once the threshold of error is reached. Generally, this threshold gets reduced over the iterations as the geometric median converges.

In early stopping, similar to how ML models exit once the loss/accuracy is maintained consistently over the last few sets of epochs, the same way if the delta of old and new estimates over the last three iterations remains the same, then merge method exits of these iterations with the last estimate. This prevents the merge method from going further iterations as the algorithm itself has reached the convergence.

Oscillation in the geometric median is generally observed when there is more than one convergence point. This indicates that there is more than one center which is equidistant from all the data points. In such cases, the median value oscillates between these centers. This merge method detects such oscillations in the centers and exits out of the iteration loop. This oscillation detection technique implemented in this merge method can detect oscillations when there are less than three centers. It works generally in most of the cases. Also, when it is not able to detect any oscillation, the merge method completes its total iterations and exits with the last updated estimate value.

All these techniques applied to the geometric median reduce the total number of iterations if the desired criteria are met. Thus, by reducing the total time, it takes to find the optimal geometric median value.

Optimization in SL for coordinate and geometric median methods

Unlike mean, coordinate and geometric median methods take more memory for the merge calculation. The need for optimization arises when the SL nodes are built with limited memory. Also, for the complex ML use cases where the size of the intermediate models is extremely large, the memory required at the SL leader node is ‘n’ (total number of SL nodes in the training) times the intermediate model size at a given node. For such a scenario, a regular memory size in the SL nodes is not sufficient to perform the merge technique.

Swarm learning provides a flag called 'SL_RAM_INTENSIVE' to optimize the usage of memory for these discussed use cases. This flag is specific to each SL node and can be passed as a configurable parameter within the swop profile via the 'slenvvars' key. Users can set this flag to True or False depending on the hardware capacity of the individual SL nodes. This flag gets passed to the respective SL container via an environmental variable and this variable gets picked up while the swarm performs the merge operation.

RAM intensive means consuming the maximum available memory in the SL leader node to complete the merge process. SL leader node will combine all the intermediate models, store them in a single variable, and perform either coordinate or geometric median. This is the default nature in the swarm merge process and hence the default value of the flag SL_RAM_INTENSIVE is set to True. One can set this flag to False when they encounter memory issues in the SL nodes while performing a merge operation. Instead of loading all the model parameters at a time, it picks the parameters of one layer at a time. This option identifies the number of layers (like convolution, kernel, dense, normalization, etc.) in the intermediate model parameters, picks parameters of one layer at a time, and performs merge calculation. This process is continued for all the layers and the merged model is built from the outcome of the individual layer's merged parameters. Thus, setting the SL_RAM_INTENSIVE flag as False will handle memory issues in the swarm.

Example:

```
- slnodedef:
        slenvvars : [SL_RAM_INTENSIVE : False]
```

EXPERIMENTATION AND RESULTS

Let's understand the behaviour of each merge method over different datasets. In these experiments, different biased datasets, and different numbers of SL nodes were used to understand the nature of the merge methods.

CIFAR-10

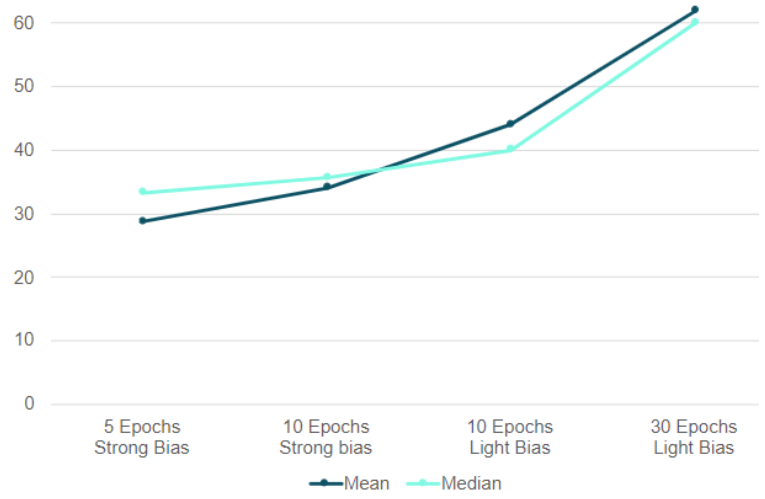
The CIFAR-10 dataset is one of the most widely used datasets for machine learning. It contains 60,000 colour images of size 32x32 pixels. There are 10 different classes in the dataset that include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class is contributed with 6,000 images. This dataset is divided as 50,000 as a training set and 10,000 as a test set. Essentially, we are training with 5,000 images per class and testing with a dataset that contains 1,000 images per class.

To apply this to swarm learning where the training is supposed to happen with the data across the nodes, it is distributed unequally among the nodes. In these experiments with the cifar-10 dataset, there were five SL and ML nodes, and the average F1 score was used as an evaluation metric for the results. To evaluate the biased nature of the dataset over the merge methods, we have applied data distribution strategies such as Absolute bias, Strong bias, and Light bias.

1. Absolute bias: Each node has exactly 2 biased classes of 5000 data points each.
2. Strong bias: Each node has 2 biased classes of 5000 datapoints and 800 datapoints for other 8 classes with 100 data points each.
3. Light bias: Each node has 3 biased classes with 2000 data points and other 7 classes with 250 to 750 data points each. The following table shows the distribution for the light bias:

Class	1	2	3	4	5	6	7	8	9	0
Node1	2000	2000	2000	750	250	750	250	250	250	250
Node2	750	750	2000	2000	2000	750	250	250	250	250
Node3	750	750	500	750	2000	2000	2000	500	500	750
Node4	750	750	250	750	500	750	2000	2000	2000	0.75
Node5	750	750	250	750	250	750	500	2000	2000	2000

With these data strategy, the Swarm Learning solution was run over different numbers of epochs like 5, 10, and 30 numbers. The following graph depicts a few observations between mean and coordinate median merge methods. The observations are Coordinate median merge method tends to perform better than the mean merge method when the number of epochs is small. This is helpful when the user has a time limitation to run for a lower number of epochs. A coordinate median is also helpful when the nature of datasets is biased over SL nodes. With the decrease in the biased nature of data, both the merge methods tend to perform similar over a larger number of epochs.



Fashion-MNIST

Fashion-MNIST is a dataset consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each sample is of 28x28 size and is of grayscale type. It has 10 classes namely t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. The experiments are conducted on 6-node and 12-node setups with varying number of epochs. The data distribution is moderately biased across all the SL nodes. Roughly, each node consists of 3-4 biased classes, and the ratio of biased and unbiased classes is 1:3. For example, if there are 9000 data points for 3 classes (approx. 3000 datapoints per each class) for a given SL node then the corresponding 7 classes were evenly distributed over 3000 data points (approx. 430 datapoints per each class). In these experiments, there is no strong and light nature in bias. All nodes are fed with a single kind of data bias strategy. The idea is to understand the behaviour in the number of epochs and the number of nodes the data is distributed along.

In an experiment with 6 Node Fashion MNIST example, there are six SL and ML pairs spawned by the SWOP container. Each node contains near to equal size of data and the distribution of that data is biased over specific classes in all nodes. This experiment captured accuracy, F1-Score, and loss from the confusion matrix using the test data which is 10000 data points (1000 data points for each class). From the following results, Coordinate and Geometric median methods are preferable for running the training with a lesser number of epochs. On increasing the number of epochs, even though the mean looks better than the other two, the difference is not so significant. This implies all merge methods converge when there is a chance to run the model for a larger number of epochs.

6 Node FMNIST Example Results				
Accuracy	**Epochs**	**Mean**	**C-Median**	**G-Median**
	2	82.18	83	82.74
	5	88.2	88.06	88.05
	10	90.09	89.85	89.92
F1-Score	**Epochs**	**Mean**	**C-Median**	**G-Median**
	2	80.54	82.1	81.52
	5	88.02	87.94	87.91
	10	89.93	89.72	89.81
Loss	**Epochs**	**Mean**	**C-Median**	**G-Median**
	2	0.45	0.446	0.448
	5	0.324	0.329	0.327
	10	0.27	0.275	0.272

An experiment with 12 Node Fashion MNIST example also followed a similar trend as the experimented earlier. But here, when the data is distributed among 12 nodes instead of 6, median-based methods are still better at lower to medium number of epochs. It means it takes more epochs for the mean to perform better than the median methods. If we run the same experiment for even more epochs like 20 or 50, then all the results are expected to be the same.

12 Node FMNIST Example Results				
Accuracy	**Epochs**	**Mean**	**C-Median**	**G-Median**
	2	78.54	78.68	78.89
	5	82.92	84.23	84.43
	10	88.23	88.23	88.21
F1-Score	**Epochs**	**Mean**	**C-Median**	**G-Median**
	2	76.77	77.09	77.29
	5	81.37	83.25	83.54
	10	88.05	88.17	88.09
Loss	**Epochs**	**Mean**	**C-Median**	**G-Median**
	2	0.557	0.556	0.551
	5	0.432	0.422	0.418
	10	0.324	0.325	0.324

RECOMMENDATIONS

Mean must work well for most of the use cases and is the default merge method in Swarm. HPE recommends you trying Coordinate median and Geometric median methods under the following circumstances.

1. The coordinate or geometric median merge methods are recommended for scenarios where the model is relatively complex and takes a longer duration to run each epoch. The rate at which the model converges over epochs is better in median based merge methods for such complex models.

2. The strong bias nature of the datasets tends to have non-uniform weights and biases in the intermediate models across ML nodes. For such biased datasets, coordinate median and geometric median methods are known to perform better than the mean.

CONCLUSION

Users must consider this merge method as one of the swarm hyper-parameters and expect to experiment with it. In this paper, we shared our findings and recommendations specific to the experimentations. Depending on the use case, the behaviour of the methods might be significantly better as well. For Generic use cases, HPE recommends you to try specific merge methods based on the suggested recommendations.

Authors

- Giri Sai Ram Ragam, Swarm Learning Team

Reviewers

- Suresh Soundararajan, Swarm Learning Team
- Milind Desai, Swarm Learning Team

October 2023