Auto Tuned Swarm Learning Models

S, Madhusoodhana Chari, Nambiar, Divya C, Dhimate, Vikrant Mah, Jain, Hemant

HPC and AI GBU

madhusoodhana.cha.s@hpe.com, dnambiar@hpe.com, vikrant.mah.dhimate@hpe.com, hemant.jain@hpe.com

Abstract.

HPE swarm learning is a de-centralized and privacy preserving solution which is emerging today to unleash the power of the big data from different nodes securely. As cited in white paper, the models generated by swarm learning should be with comparable accuracy to centralized one. The current design of swarm learning exceeds in terms of security and latency, but the accuracy of de-centralized learning is still in research. De-centralized learning lacks the part of tuning of the hyper parameters, (finding the best parameter which we can input to train the model to achieve maximum accuracy) the way we can use in the centralized learning. This creates a scope to integrate and improve the accuracy of the de-centralized learning, as accuracy is an important indicator which tells how well the knowledge embedded in the data is captured and drives the future insights. In this paper, we propose inclusion of Optimization Layer consisting of Hyper Parameter optimizer to swarm learning architecture which can be further generalized for other de-centralized learning platforms as well.

Problem statement

Today HPE swarm learning is popular and in demand because of its secure and privacy preserving architecture. The nodes in swarm network shares the parameters (weights), extracted from training the ML model on their individual local node data. These parameters, shared from all the nodes are merged to obtain a global model. Swarm learning mandates that all the edges in consortium should be agreed upon same common ML model during training phase. Parties in the consortium will not be aware of the nature of individuals local data. Thus comparatively, common ML model is less likely to give better accuracy to the local network training with heterogeneous data. This brings the need for improvising common ML model (to be used during training phase) with the help of HPO (Hyper Parameter Optimization) which will ultimately result in generating better global model.

Our solution

As mentioned in figure 1, proposed process of model training includes Enrollment, Optimization, Local model training, Parameter sharing and Merging and Stopping criterion. In the swarm architecture, every SL node (figure 2) trains its local data using the common ML model for one or more data batches with fixed number of iterations. Each SL node shares the learnt parameters to the elected leader (first participating node in the Enrollment) and parameter merging happens in the elected leader. After aggregation, leader shares them with other SL nodes for further Local model training.

We propose inclusion of additional Optimization Layer in existing swarm architecture which performs HPO (Hyper Parameter Optimization) for common ML model to improve the overall accuracy of final model, once Stopping criterion is met.

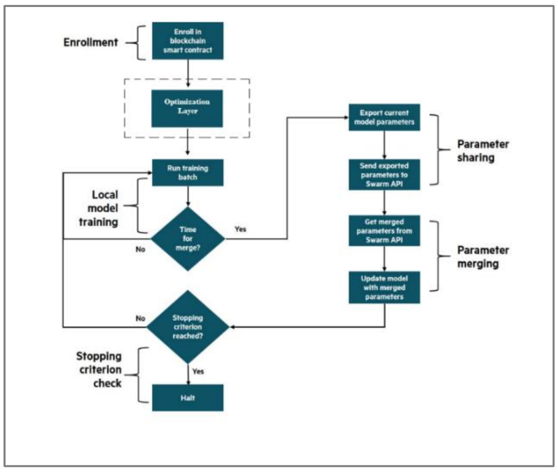

Enrollment
Enroll in blockchain smart contract
Optimization Layer
Run training batch
Local model training
Time for merge?
Yes
No
Export current model parameters
Send exported parameters to Swarm API
Parameter sharing
Get merged parameters from Swarm API
Update model with merged parameters
Parameter merging
Stopping criterion reached?
No
Yes
Stopping criterion check
Halt

figure 1

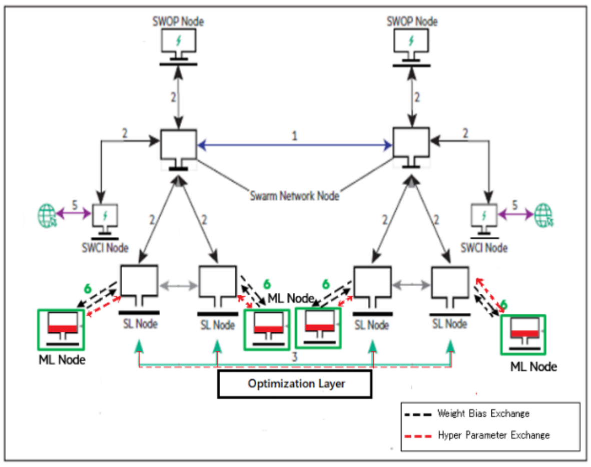


figure 2

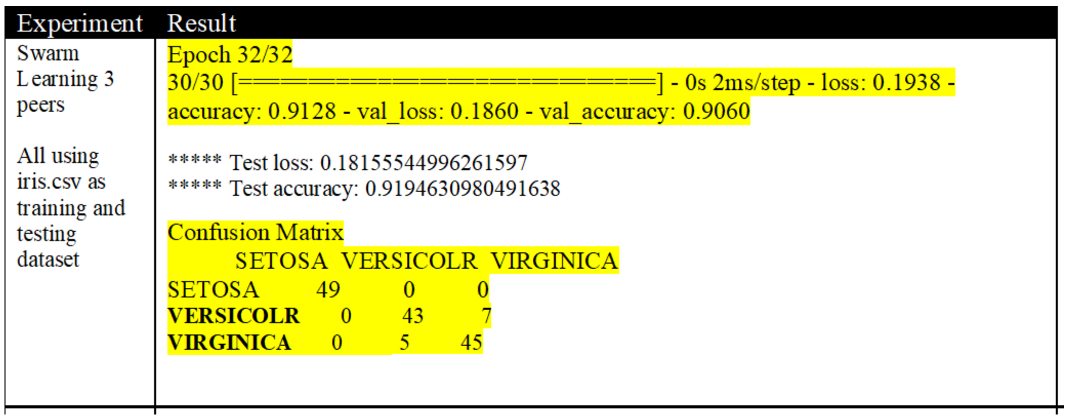
The Optimization Layer functions as below -

1. Performs hyper parameter tuning using GridSearchCV or RandomSearchCV, [inbuilt Scikit-learn libraries which give the best parameter to be feeded to common ML model based on the dataset provided] in all the nodes and sends these hyper parameters along with regular parameters to the elected leader.
2. Parameter sharing and merging step will be improved to aggregate and merge new parameters as well. e.g., The aggregation algorithm for hyper parameters like number of epoch/iterations, can be *mean, median or max*
3. List of some of the hyper parameters (e.g., Neural Network Optimizer) will be created which cannot be merged directly and needs customized logic to use them.
4. Finally, Leader will send these hyper parameters back to other SL nodes and then nodes will re-train using every combination of these parameters and choose the best combination based on the accuracy score.

Hyper parameter exchange will be triggered only once during initialization of Local model training as marked **red** vs regular parameter exchanges **black** in figure 2.

Evidence the solution works

We implemented and applied the solution on several Swarm Learning examples and found around 5 to 7% accuracy improvements. One of the experiments was conducted on iris dataset. The data set contains four features, length and width of sepals and petals and this is a IID dataset with 50 samples of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). We executed the pipeline without including Optimization Layer/Step, with three nodes using keras sequential model with the default epoch and batch size as mentioned in swarm Learning documentation i.e., 5 and 32 respectively and got model accuracy of 91% as below.



Experiment	Result
Swarm Learning 3 peers All using iris.csv as training and testing dataset	Epoch 32/32 30/30 [==============================] - 0s 2ms/step - loss: 0.1938 - accuracy: 0.9128 - val_loss: 0.1860 - val_accuracy: 0.9060 ***** Test loss: 0.18155544996261597 ***** Test accuracy: 0.9194630980491638 Confusion Matrix SETOSA VERSICOLR VIRGINICA SETOSA 49 0 0 **VERSICOLR** 0 43 7 **VIRGINICA** 0 5 45

For testing improvised solution, performed functionality of Optimization Layer where we did a grid search of epoch, batch size and optimizer on the data in one of the nodes and manually created a common ML model with these parameters.

Best batch size and epoch	**Best optimizer**

Best: 0.705729 using {'batch_size': 10, 'epochs': 100}
0.597656 (0.030425) with: {'batch_size': 10, 'epochs': 10}
0.686198 (0.017566) with: {'batch_size': 10, 'epochs': 50}
0.705729 (0.017566) with: {'batch_size': 10, 'epochs': 100}
0.494792 (0.009207) with: {'batch_size': 20, 'epochs': 10}
	Best: 0.697917 using {'optimizer': 'Adam'}
	0.674479 (0.033804) with: {'optimizer': 'SGD'}
	0.649740 (0.040386) with: {'optimizer': 'RMSprop'}
	0.595052 (0.032734) with: {'optimizer': 'Adagrad'}

After running the same experiment again with the common ML model derived from Optimization Layer, we got accuracy of 97% for the same dataset using same learning technique.

Experiment	Result
Swarm Learning 3 peers All using iris.csv as training and testing dataset	Epoch 100/100 30/30 [==============================] - 0s 2ms/step - loss: 0.0622 - accuracy: 0.9732 - val_loss: 0.0750 - val_accuracy: 0.9732 ***** Test loss: 0.056650757789611816 ***** Test accuracy: 0.9731543660163879 SETOSA VERSICOLR VIRGINICA SETOSA 49 0 0 VERSICOLR 0 46 4 VIRGINICA 0 0 50

Competitive approaches

There is competitive approach as cited[1], which performs hyper parameter optimization in Blockchain. Cons of this approach is parameter tuning inside the block chain. Solidity, an object-oriented programming language created specifically by the Ethereum Network team, has a size limit for the number of items in a struct and number of parameters passed to the functions. In addition to that, the data will be read block by block in Ethereum. So, carrying out HPO in block chain is not recommended as it can impact on scale and performance aspects.

Block chain is recommended to be light weight and hence proposed solution in the paper shows better approach over the competitive solutions.

Current status

We are experimenting on this from past couple of months on various datasets. The experimentation specific to iris data set along with model and related code script can be found at https://github.hpe.com/dnambiar/TechCon-Swarm

Next steps

We will be continuing our experimentation and documenting results for further experiments on several other datasets for generalization of the solution. We will take this solution ahead and integrate an API in Swarm Learning for hyper parameter tuning and exchange.

Acknowledgements

The authors would like to thank swarm learning team for their innovation on HPE Swarm Learning.

References

[1] Competitive Approach

https://www.researchgate.net/publication/341518171_Hyperparameter_Optimization_Using_Sustainable_Proof_of_Work_in_Blockchain

[2] Swarm Learning https://www.labs.hpe.com/pdf/Swarm_Learning.pdf

[3] Hyper Parameter Tuning https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

[4] https://www.cs.cmu.edu/~mkhodak/docs/FL2020Workshop.pdf

Keywords: Swarm Learning, Hyper Parameter Optimization