LEARNAE: Distributed and Resilient Deep Neural Network Training for Heterogeneous Peer to Peer Topologies

Spyridon Nikolaidis and Ioannis Refanidis[1]

[1] University of Macedonia, Thessaloniki 54636, GREECE
{sp.nikola, yrefanid}@uom.edu.gr

Abstract. LEARNAE is a framework proposal for decentralized training of Deep Neural Networks (DNN). The main priority of LEARNAE is to maintain a fully distributed architecture, where no participant has any kind of coordinating role. This solid peer-to-peer concept covers all aspects: Underlying network protocols, data acquiring/distribution and model training. The result is a resilient DNN training system with no single point of failure. LEARNAE focuses on use cases where infrastructure heterogeneity and network unreliability result to an always changing environment of commodity-hardware nodes. In order to achieve this level of decentralization, new technologies had to be utilized. The main pillars of this implementation are the ongoing projects of IPFS and IOTA. IPFS is a platform for a purely decentralized filesystem, where each node contributes local data storage. IOTA aims to be the networking infrastructure of the upcoming IoT reality. On top of these, we propose a management algorithm for training a DNN model collaboratively, by optimal exchange of data and model weights, always using distribution-friendly gossip protocols.

Keywords: Decentralized Neural Network Training, Distributed Asynchronous Stochastic Gradient Decent, Model Averaging, Peer-to-Peer Topologies, Distributed Ledger Technology, IPFS, IOTA.

1 Introduction

During the past years, scientific community has made significant advances regarding parallel DNN training. There are many aspects that can be dealt with in a decentralized way. Each implementation may also have a different level of decentralization, based on the intended use case. In most cases the proposed works rely on a centralized approach like a parameter server [1][2]. These approaches require high performance infrastructure, since all nodes have to communicate with the server and exchange models after each optimization step. Decentralized attempts, such as [3], use local optimization and asynchronous model merging reducing the communication demand, but the parameter server is still a bandwidth bottleneck limiting scalability. Other distributed deep learning systems [4][5] are able to overcome this bottleneck, but for doing so they need low-latency networking like InfiniBand, which results to very high setup cost and narrows

down use cases. Proposals like [6] focus on medium performance hardware but they still utilize synchronous frameworks like [7], which are not the optimal choice for loosely connected peers.

In this paper we propose a framework that stretches decentralization and tolerance to maximum. Based on purely distributed peer-to-peer technology, it has no need for servers or any kind of strict synchronization. Its indented use case are environments with commodity-hardware nodes and networking infrastructure with moderate latency and connectivity. Our approach supports versatile data acquiring from different types of sources, including lightweight IoT devices, and uses novel Distributed Ledger Technologies as the data diffusion mechanism.

The rest of the paper is structured as follows: Section 2 presents technological background knowledge, whereas Section 3 presents the LEARNAE architecture. Section 4 presents experimental results to evaluate the whole approach and, finally, Section 5 concludes the paper and identifies future research challenges.

2 Technological Background

This section presents the related technological background that forms the basis for the LEARNAE framework.

2.1 IPFS

IPFS is a distributed filesystem for peer-to-peer networks, where no node has special privileges compared to others. It is comprised of previously established and successful techs, plus a novel block exchange protocol. Its main goal is to achieve efficient storage and availability of big data, with no need of a centralized authority, supporting features like immutability, deduplication, versioning and content-based addressing [8].

IPFS may be seen as an intuitive combination of Git [9], the distributed source code version control system, and BitSwap, a novel data replication incentivizing algorithm inspired by BitTorrent [10], the decentralized block exchange protocol.

In IPFS every node is identified by a unique ID which is the cryptographic hash of a public key. Although a node owner has the freedom to change this ID, it is not advised, since by doing so they will lose all the benefits the node has gained by its participation to the network. Unlike most filesystems, IPFS is not using an addressing method based on the location the data are stored in, instead any file can be acquired by just using the cryptographic hash of its contents. Files larger than a specific size are sliced to blocks which are assigned with their corresponding hash. In order to route a request, each IPFS node utilizes a Distributed Hash Table (DHT), a ledger (based on [11][12][13]) that contains pairs of block/peer information. For a small block, DHT contains the actual block data, while for a larger one it contains the IDs of peers that can serve the specific block.

Regarding block exchange strategy, every IPFS node maintains a list of blocks it needs and one of blocks it already has. A major difference compared to BitTorrent's method [14] is that those lists are not limited to a specific torrent or a group of such,

but may include any block that has appeared to the network, no matter what file they are part of. Since there will be cases where two peers will not need a block stored in each other, or cases where a node needs nothing, some kind of credit has to be applied. Each node is incentivized to increase its credit with its peers, because by doing so it also increases the possibility others will help it acquire the blocks it will need in the future. For a pair or peers this credit is in fact the balance of verified bytes exchanged between the two.

Since the main concept of IPFS is decentralization, no data servers could be used to store the files. All nodes participating to the network must provide some local storage (similar to [15]). Every requested block is fetched from another peer and then saved to the local storage. If a peer owner wishes to permanently retain a file locally, they can "pin" it to avoid garbage collection. Considering all the above, IPFS should be able to be used as resilient big dataset distribution method, ensuring load balancing between the nodes and a configurable data replication, in order to anticipate node downtime on loosely connected commodity hardware.

PubSub. IPFS supports a Publish-Subscribe scheme which allows instant messaging. If a peer needs access to a specific "topic", it subscribes to it and by doing so can listen and broadcast data to that channel. As expected, PubSub has no need for centralized authorities and all its messages propagate using gossip protocol. LEARNAE utilizes IPFS PubSub feature to exchange training metadata among the peers.

2.2 IOTA

The IOTA network [16] aims to create a decentralized infrastructure to support all forms of data exchange in the upcoming Internet of Things (IoT) era. Although it supports a cryptocurrency token, automated micropayments between devices is just a portion of its use cases. It is based on a structure known as Directed Acyclic Graph (DAG), which in the IOTA community is referred to as the "Tangle" [17]. In general IOTA attempts to achieve a consensus via permissionless procedures, using the Tangle as a Distributed Ledger Technology (DLT) and a gossip protocol to propagate transactions throughout the network. In order to attach a new transaction, a node has to confirm two previous transactions, so each addition contributes to the overall stability and performance of the network [18].

Masked Authenticated Messages. The IOTA feature-of-interest for this paper is known as "Masked Authenticated Messages" (MAM). This feature allows lightweight IoT devices to attach zero-valued transactions to the Tangle by just performing a small amount of "Proof of Work" (PoW), which is solving a cryptographic puzzle. PoW is necessary in order to avoid spamming from bad actors. Each of these transactions includes a data portion, making MAM an IoT-oriented way to broadcast data streams, which can support both encryption and authentication.

MAM messaging is based on a publish-subscribe logic. A data stream is constructed as a singly linked list, where each transaction points to the next one. If someone knows

the address (called “root”) of a specific transaction, they can read all of the following stream, but will have no access to previous data. Knowing the first root of a MAM chain grants access to all the messages it contains.

A MAM stream may have one of three different accessibility modes: Public, Restricted and Private. In Public mode, everyone can view the messages. In Restricted mode, only those who know a “sidekey” can read the data. In private mode, only the stream owner can access the data, since it requires knowing the seed that created the MAM root sequence.

3 Implementation

3.1 Design decisions

Parallelism type. The first decision that has to be made is between model [19] and data [20][21] parallelism, although there are works that propose hybrid systems. LEARNAE adopts data parallelism. According to this approach, each worker keeps the entire model locally and processes it using a subset of the training data.

Propagation. After processing on workers, the produced models have to be combined. The two major methods for doing so are weight and update averaging, each of them having its own advantages and drawbacks. This proposal uses weight averaging, thus after training the actual value (not just the update) of each model parameter is -under specific conditions- averaged with the actual value of the correspondent parameter of a selected worker’s model [22].

Coordination. The method for merging the above models can also have different levels of decentralization, as seen in the following Table 1:

Table 1. Different approaches regarding training coordination.

Level of decentralization	Coordinator entity
None	Parameter server
Low	Cluster of parameter servers
Medium	Some peers have elevated role
High	None

A parameter server has the task of receiving, combining and redistributing the averaged data. The use of a server in most cases results in a training speed increase, but it also creates a single point of failure and -in large scale networks- a bandwidth bottleneck. This drawback can be reduced using more servers that cooperate with each other. For cases where the presence of a server is not feasible or desired, some of the participating peers are assigned special coordinating tasks, while also functioning as all other training workers. At the edge of this spectrum are the implementations where

no node has additional duties regarding coordination, creating a fully distributed environment, which is the design followed by the current proposal.

Synchronicity. The training procedure may be synchronous or asynchronous. In synchronous designs the coordinating entity ensures that all results are only combined with others produced at the same training phase. In asynchronous designs there is no such need and the results of a worker can be embedded into the global model under more loose time-based rules. Each approach has pros and cons. Synchronous training may converge faster, since it prevents merging of irrelevant data, but it may have locks from slow peers that can delay the whole process. Asynchronous training achieves high worker utilization, but suffers from high gradient staleness, meaning that by the time a worker submits its results they are already out of date compared to the global model. Many strategies have been proposed [23] to mitigate those downsides, resulting to a large number of variants, especially for Asynchronous Stochastic Gradient Decent. LEARNAE adopts an asynchronous design, although it contains features which, if used in future implementations, may inject a configurable amount of synchronicity.

3.2 Architecture

LEARNAE is based on a versatile working scheme and can adapt to different environments. Regarding data, it supports use cases where all training data are poured into the network during the initial phase, before any training. It also supports use cases where data feeding is a continuous task and the training of the DNN model is an always-progressing procedure. Thus, there is no limitation about the time training data may be injected into the network, which is a critical feature when it comes to censor data streaming from IoT devices.

For enhanced versatility there are 4 different types of node roles (Table 2). What role a node has depends on whether it has access to training data and on its computational capability.

Table 2. Supported node types and their features.

Node role	Platform	Model training	Data feeding
Full	IPFS + IOTA	Yes	Yes
Trainer	IPFS + IOTA	Yes	No
Feeder (fat)	IPFS	No	Yes
Feeder (thin)	IOTA	No	Yes

The first three roles in Table 2 require enough computational power to support participation to the IPFS swarm. The fourth role is indented for IoT domain, since data streaming through MAM messages may be broadcasted even by lightweight sensor devices. Fig. 1 demonstrates data flow between nodes of different roles.

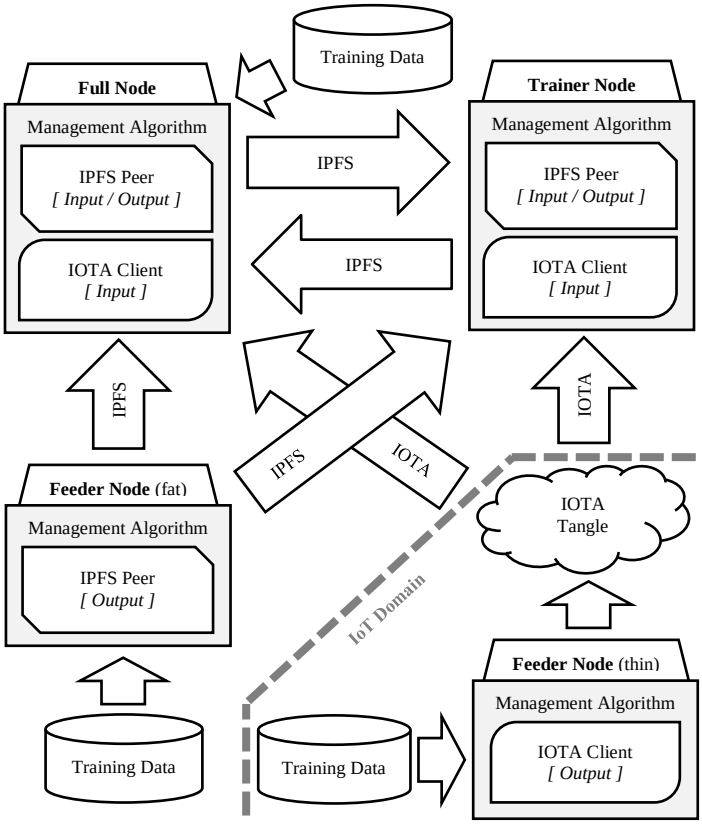

Fig. 1. Data flow between different node roles.

The ID of the used publish-subscribe channel (IPFS and/or IOTA) is the connecting link between peers. Knowing this ID allows a peer to participate to the network by listening and broadcasting messages. Fig. 2 shows the workflow of a node's "Listening thread". There are 3 different types of messages:

Slice hashlist. This message contains the hash of a file that a node made available on IPFS. This file contains a list of hashes of training data slices that were also made available on IPFS by the same peer.

Remote model. This message contains the hash of a file that a node made available on IPFS. This file contains the model of that peer in HD5 format. Other metadata of the model are also included, like the achieved accuracy and the maturity of the model (the number of the training cycles elapsed up to its creation).

Slice use stats. This message informs all participating peers that a specific data slice has been used for training by a peer. This info allows the implementation of "overuse threshold" feature, that is optionally setting a limit on how many times a data slice may be used for training on different nodes.

A node may perform up to four main tasks, as described in Table 3:

Table 3. Node tasks

Task	Description
Adding	Add new data to IPFS.
Pinning	Make remote IPFS data available locally.
Training	Train local model.
Averaging	Average local and a remote model.

In order to maximize node utilization, Learnae uses a different execution thread for each task. All of them can work simultaneously, with the exception of the pair Training/Averaging, since those both need read-write access to local model. Fig. 3 demonstrates the work cycle of a full node.

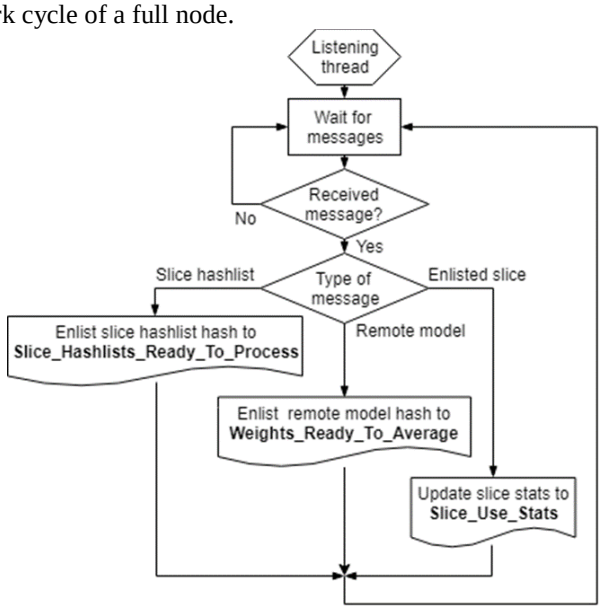

Fig. 2. Workflow of a node's "Listening thread".

4 Evaluation

4.1 Current scope

This first paper is a preliminary study on the viability of using modern DLTs as data diffusion mechanism of an asynchronous Stochastic Gradient Decent algorithm. Since this is the initial approach, the following scope limits are applied:

- The performance of IPFS under the described use case will be reviewed. All tests will be run on a single machine, simulating different peers using virtualization. This paper focuses on proof-of-concept metrics and all tests will be short runs of one-

hour timespan. Although the training dataset contains approximately 10 million instances, only a limited fraction of those were used.

- Full deployment on actual network of commodity hardware, comparison to traditional methods, extended time runs, quantity-based metrics and full training data usage will be studied in a subsequent work.

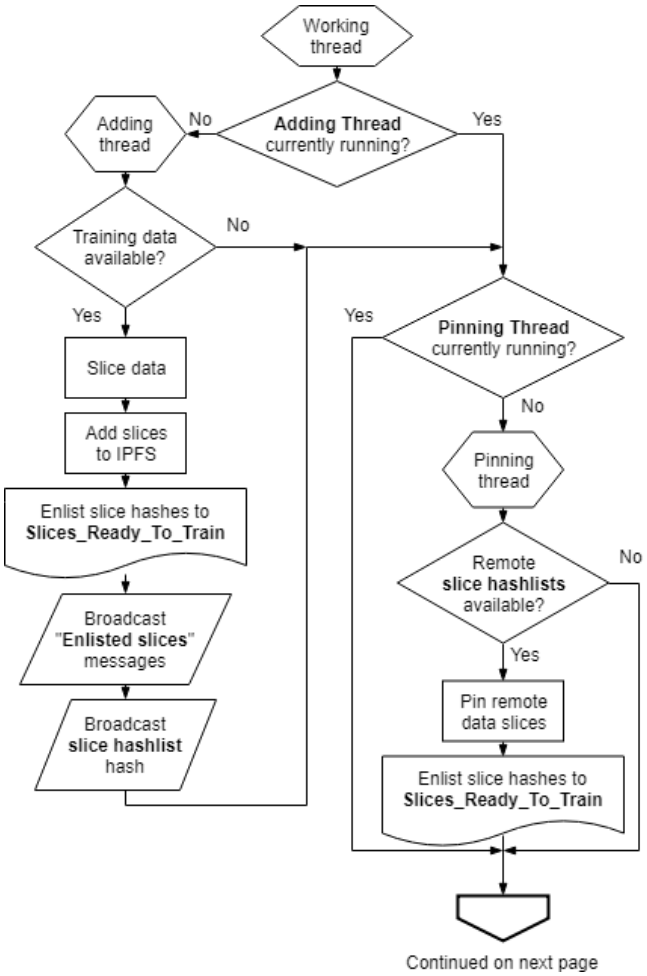

Fig. 3a. Workflow of a node's "Working thread" (1/2).

4.2 Simulation

The simulation runs of this first approach were executed on a virtual network of 10 workstations. The network was implemented on a single commodity computer with Docker containers running the IPFS daemons. The coordinating application was developed in C# and it contains both the distributed training management algorithm and the logging mechanism. The dataset used was HEPMASS [24] (exotic particle detection).

As seen in the following figures, simulation aims to study the effect of some key characteristics like data slice size and overuse threshold, among other resilience-oriented features like duplication level and overhead.

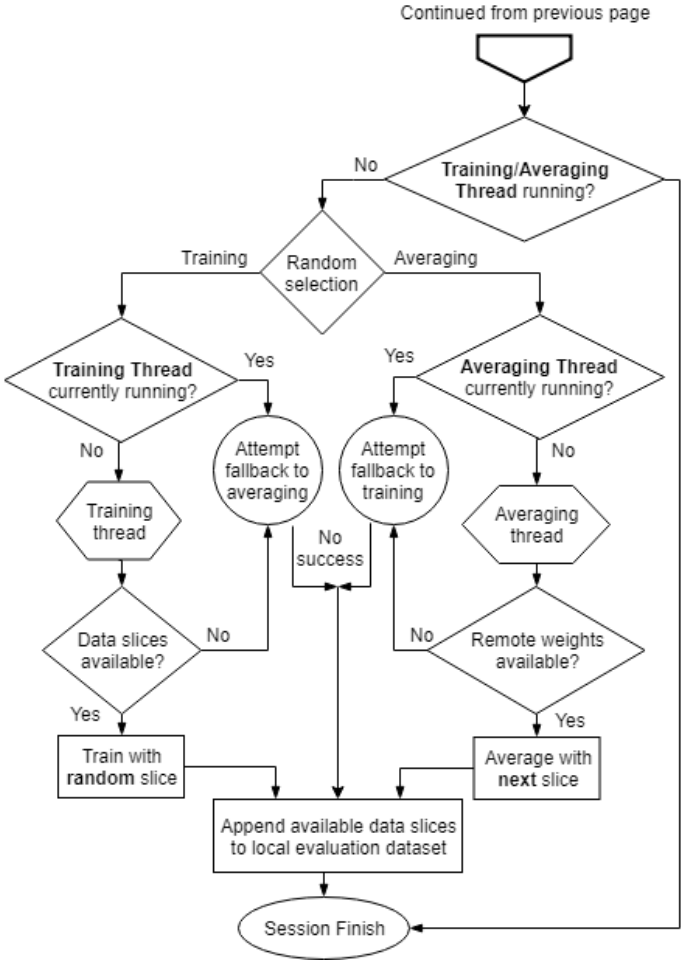

Fig. 3b. Workflow of a node's "Working thread" (2/2).

Simulation shows that increase in slice size has a positive impact on average accuracy of the produced models (Fig. 4). Lower overuse threshold results to slightly better accuracy, since it reduces repetitive training with same data (Fig. 5).
As expected, total bytes sent are proportional to selected slice size (Fig. 6). The same applies to average resilience (Fig. 7), which is defined as the number of nodes owning a requested slice/model. Fig. 8 demonstrates the percent of duplicate data sent, that is an unavoidable overhead due to gossip-based protocol. This percentage, although enormously high at the beginning, is quickly declining by time and minimized for larger slice sizes.

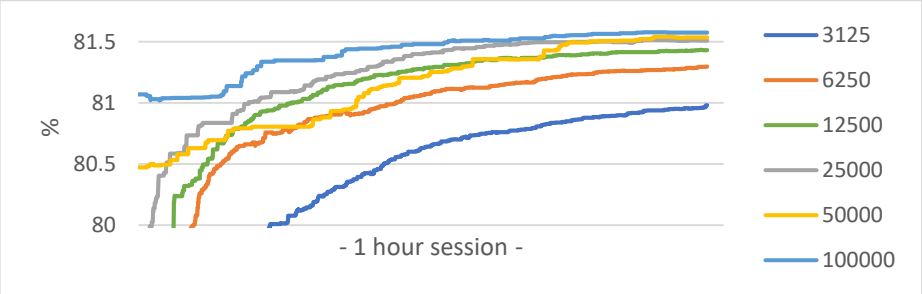

Fig. 4. Average Accuracy per Slice Size (Overuse Threshold: 6)

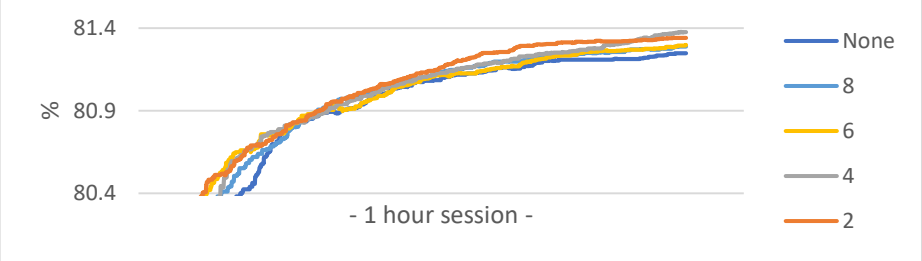

Fig. 5. Average Accuracy per Overuse Threshold (Slice Size: 6250)

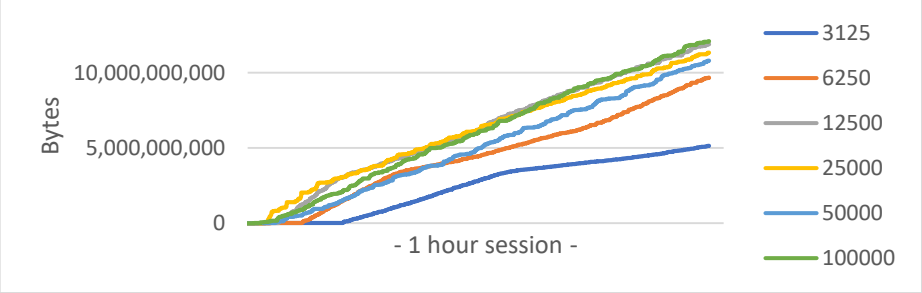

Fig. 6. Total Bytes Sent per Slice Size (Overuse Threshold: None)

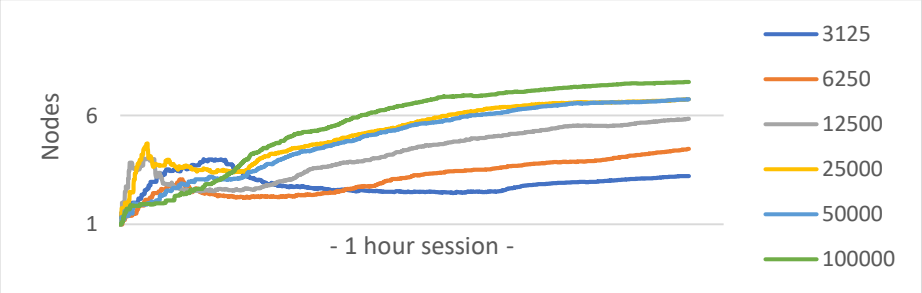

Fig. 7. Average Resilience per Slice Size (Overuse Threshold: 2)

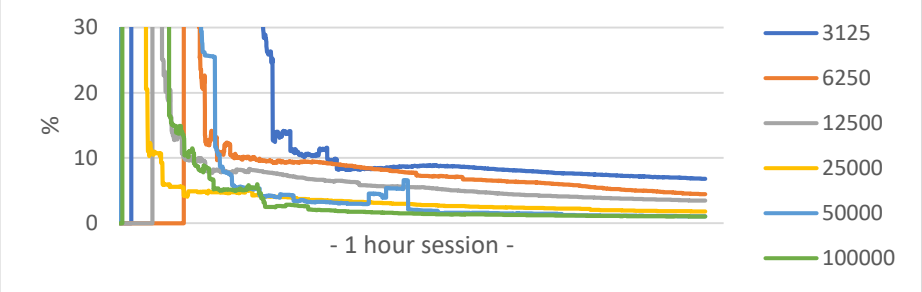

Fig. 8. Duplicate Data Sent per Slice Size (Overuse Threshold: None)

5 Conclusions and Future Work

This paper is a first approach on using Distributed Ledger Technology as the data diffusion mechanism for decentralized DNN training, in order to achieve a fully distributed and peer-to-peer architecture. Thus, this study should be seen as a proof-of-concept for the specific recipe. Although the used dataset contains approximately 10 million instances, only a small fraction of those (100,000) were used in this phase.

Many interesting questions remain to be addressed, especially those concerning quantity-based metrics: What is the performance on real-life commodity hardware with realistic network latencies and downtimes? How well can such an ecosystem scale? What is the achieved accuracy when using more data? What is the maturity level of new concepts like IOTA Tangle? What are the specifics of the performance/resilience tradeoff? All these questions will be the subject of future work.

6 Acknowledgements

This research is funded by the University of Macedonia Research Committee as part of the “Principal Research 2019” funding program.

References

1. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A System for Large-Scale Machine Learning” Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation, pp. 265–283, 2016.
2. J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Mao, M. Razato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, “Large Scale Distributed Deep Networks,” in Advances in Neural Information Processing Systems, 2012, pp. 1223–1231.
3. S. Zhang, A. Choromanska, and Y. LeCun, “Deep learning with Elastic Averaging SGD,” Advances in Neural Information Processing Systems, pp. 685–693, 2015.

4. T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, “MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems”, in Proceedings of LearningSys, 2015.
5. F. N. Iandola, K. Ashraf, M. W. Moskewicz, and K. Keutzer, “FireCaffe: near-linear acceleration of deep neural network training on compute clusters”, 2015.
6. M. Langer, A. Hall, Z. He and W. Rahayu, "MPCA SGD—A Method for Distributed Training of Deep Learning Models on Spark" in IEEE Transactions on Parallel and Distributed Systems, 2018.
7. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing,” in Proc. of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012, pp. 15–28.
8. Juan Benet. IPFS - Content Addressed, Versioned, P2P File System.
9. A. J. Mashtizadeh, A. Bittau, Y. F. Huang, and D. Mazieres. Replication, history, and grafting in the ori file system. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pages 151–166. ACM, 2013.
10. B. Cohen. Incentives build robustness in bittorrent. In Workshop on Economics of Peer-to-Peer systems, volume 6, pages 68–72, 2003.
11. I. Baumgart and S. Mies. S/kademlia: A practicable approach towards secure key based routing. In Parallel and Distributed Systems International Conference, 2007.
12. M. J. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In NSDI, volume 4, pages 18–18, 2004.
13. L. Wang and J. Kangasharju. Measuring large-scale distributed systems: case of bittorrent mainline dht. In Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on, pages 1–10. IEEE, 2013.
14. D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent’s incentives. In ACM SIGCOMM Computer Communication Review, volume 38, pages 243–254. ACM, 2008.
15. J. Dean and S. Ghemawat. leveldb–a fast and lightweight key/value database library by google, 2011.
16. IOTA Foundation homepage, https://www.iota.org, last accessed 2019/02/14.
17. Serguei Popov. The Tangle, April 30, 2018.
18. Serguei Popov, Olivia Saa, Paulo Finardi. Equilibria in the Tangle, March 3, 2018.
19. Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Ng Andrew, “Deep learning with cots hpc systems” in Proceedings of The 30th International Conference on Machine Learning, 2013, pp. 1337–1345.
20. Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur, “Improving deep neural network acoustic models using generalized maxout networks” in Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. IEEE, 2014.
21. Yajie Miao, Hao Zhang, and Florian Metze, “Distributed learning of multilingual dnn feature extractors using gpus” 2014.
22. Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur, “Parallel training of deep neural networks with natural gradient and parameter averaging”, 2014.
23. Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu, “1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns” in Fifteenth Annual Conference of the International Speech Communication Association, 2014.
24. HEPMASS Dataset homepage, http://archive.ics.uci.edu/ml/datasets/hepmass, last accessed 2019/02/14.