FlexEnt: Entropy Coding to Curb Stragglers in Large-Scale Distributed Machine Learning

James Salamy Ayush Sharma Manya Ghobadi Muriel Médard
Massachusetts Institute of Technology

Abstract

Distributed Machine Learning training is emerging in today's datacenters. The training time of a distributed ML job is closely tied to the parallelism technique used to distribute the job across worker nodes, as well as the available compute, storage, and network capabilities in the datacenter. Current parallelism techniques often implicitly make two simplifying assumptions: (*i*) datacenter nodes are homogeneous and (*ii*) there is little to no congestion in the network. However, at scale, these assumptions do not always hold and stragglers are a pervasive challenge in today's datacenter job scheduling. A straw-man approach to reduce the impact of stragglers is to replicate (sometimes speculatively) each sub-task. However, this approach commonly incorporates a timeout component to recover from straggler sub-tasks. In this work, we address the problem of how to curb the impact of stragglers in distributed ML jobs without having to speculate. Our approach leverages the unique nature of image recognition workloads and the *theory of graph entropy*. Our proposal, FlexEnt, identifies the entropy present in a training batch using a set of "feature filters" and replicates only a fraction of the training data on a predetermined set of "shadow workers". In doing so, FlexEnt will decode the training parameters as soon as a sub-set of sub-tasks are completed. This approach will enable datacenter operators to adjust a knob between the number of additional shadow workers and the degree of stragglers protection.

1 Introduction

Today's distributed Machine Learning (ML) workloads rely on either synchronous [1–3] or asynchronous updates between nodes [4–7]. A synchronous approach loads a fraction of the training data on to each node, then waits for the training algorithm to run, aggregating the results at each iteration using a parameter server or ring-reduce arrangement [3, 8–12]. In an asynchronous approach, the goal is to remove the dependency on scheduling and the behaviour of other machines to remove the effect of slow machines or congestion in the network [6, 12–15].

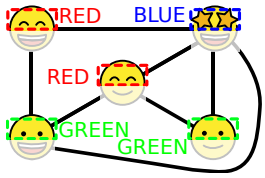
Synchronous approaches are broadly successful for small numbers of workers, as the synchronisation constraints are not overly limiting at small scale [7, 16]. However, the probability of a delay caused by a congested link or a slow worker grows exponentially as the system is scaled up [17]. This problem is exacerbated when a training task is run across heterogeneous nodes and workloads, since heterogeneity is a major source of stragglers [16, 18–20]. New approaches, such as Horovod [3], reduce the dependency of training on the network bandwidth available for sharing gradient updates between workers using the all-reduce algorithm's ring structure [3]. However, by its nature this structure is vulnerable to stragglers, as a single slow machine or a congested link can limit the completion of the overall calculation [21].

Large-scale distributed ML workloads are expected to run in the heterogeneous cloud environments, hence many jobs of various sizes share the same infrastructure [13, 16, 20, 22]. Therefore, stragglers will be prevalent due to outdated hardware, failures, or congested links. Many systems use a hard deadline or a speculative approach to decide when to abandon the slow workers [18, 23]. To avoid using deadlines, one approach is to replicate the entire training task [23]. While this approach ensures there is always a redundant copy available of each worker, it also consumes a significant quantity of additional resources per level of protection obtained. In this work, we present FlexEnt, a coding-based scheduling system for training jobs. FlexEnt curbs the effect of stragglers in large-scale training jobs by selectively replicating high-entropy data as described in the next section.

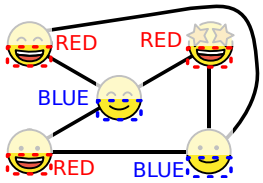
2 FlexEnt System Description

The key to FlexEnt is to create sub-sampled sets of 'shadow' training data, formed from coding amalgams of the original training data. These high-entropy subsets can be trained on 'shadow' workers, and made available to the 'primary' workers performing the main training for a just-in-time replacement of gradient updates from stragglers. The number of shadow workers creating coded jobs is dependent on the desired level of straggler protection. The effectiveness of coding techniques for ML inference has been demonstrated in prior work [24]. However, a unique challenge of using any coding theory for training workloads is that the fundamental assumption of linear processing does not apply to ML training. We solve this challenge by 'porting' the required coding to the gradient aggregation step, which is a linear operation carried out on the results of a non-linear computation, and hence can be coded.

Our approach is designed to be applicable to commonly used stochastic ML techniques such as stochastic gradient descent [2, 8, 25]. FlexEnt's goal is to build a reasonably accurate estimate of the output gradient vector from training on a quasi-redundant, but entropy rich, copy, based on the



(a) Entropy graph for a dataset using the "eyes" feature.



(b) Alternative entropy graph for the same dataset using the "mouths" feature.

Figure 1. Intuitive examples of entropy graphs where emojis with different features are connected to each other. We then colour these graphs based on adjacency. To capture multiple dimensions, we repeat this process over many features, which produce different graphs such as (a) and (b). From information theory, illustrated in Fig. 2, we can use the joint entropy to identify the 'entropy-rich' samples.

entropy graph colouring of the input data. To do so, FlexEnt goes through the following steps:

Setup Phase

- Given n worker nodes, select s nodes as shadow workers and $n - s$ nodes as primary workers;
- Assign a fraction of data (batch) to each primary worker.

Sampling Phase

- Calculate the joint entropy of each entry in the dataset under a set of features as explained in section 3;
- Select the entries that maximise the joint entropy and assign these to the shadow workers.

Training Phase

- Repeatedly run training iterations on both primary and shadow workers;
- Use shadow workers' approximate gradients to replace a straggler's gradients during update steps.

3 Entropy Graph Colouring

In information theory, an entropy function is built from a model function's characteristic graph. The colouring of this graph represents the entropy available in the data [26, 27]. Figure 1 illustrates an example of two entropy graphs and their associated colouring. We take a dataset of five images to be processed in a distributed fashion. Our goal is to produce an entropy-rich sub-set of the images to be replicated across shadow workers. We demonstrate this process using two features appropriate to capture the highest entropies across the dataset. Images that contain a feature, shown by boxes in Figure 1, essentially have the same output in this dimension, and are therefore not connected in the graph. This allows us to draw and colour the entropy graph, as shown. We then sample the entropy graphs to obtain the maximum diversity across all colours, represented by the joint entropy shown in Figure 2, allowing a sampling rate that is below the straight sum of each dimension's rate.

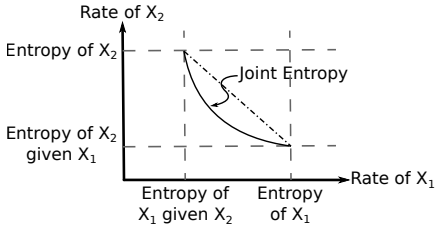


Figure 2. Concave behaviour of the joint entropy output from two features (for example from Figure 1). Rates are the sample rates of data from each entropy feature [27].

János Körner's information theoretic approach shows that the total (joint) entropy for an image set can be estimated by the entropy of colouring graphs of k features [27]. We use this information in conjunction with the approach in [28] to select entropy-rich samples for our shadow nodes. The goal is to select the largest number of colour groups possible, across the features used, to achieve the largest entropy for a given batch size.

Our approach ties into the graph entropy of Körner's [26, 29], further developed to cover the idea of rate-distortion [27]. This theory supports the principle that in terms of overall resource utilisation, it is advantageous to proceed with a fraction of each dimension of data rather than running the same amount of time with the full missing space set [27]. This is because of the concave shape of the rate curve drawn from the entropy graph as shown in Figure 2, where X is a random variable of a feature over the data.

Our preliminary results with the MNIST dataset confirm this theory. We find that when a primary worker is straggling behind, replacing its missing gradients with a shadow worker that trains on a randomly sub-sampled data achieves better accuracy compared to abandoning the primary worker. In future work, we plan to use the features during training iterations to select entropy rich data. Intuitively, gradient estimates produced from high-entropy samples should be significantly improved compared to random sampling, as they contain more information about the training space.

4 Conclusion

FlexEnt is a system designed to mitigate the impact of stragglers on distributed machine learning jobs in datacenters. We leverage concepts from coding and information theory to provide a good estimate at a predictable soft-threshold, rather than enforcing a hard deadline or using speculating approaches to recover from stragglers. Our approach is designed to provide an adjustable knob to specify the level of straggler protection required, so that the system provides the maximum speedup within constraints.

Acknowledgments

We would like to thank Ken Duffy and Seva Shneer for helpful discussions and guidance throughout this research.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association.

[2] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys*, 2019.

[3] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint*, 2018.

[4] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 693–701. Curran Associates, Inc., 2011.

[5] Jin Kyu Kim, Abutalib Aghayev, Garth A. Gibson, and Eric P. Xing. Strads-ap: Simplifying distributed machine learning programming without introducing a new programming model. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 207–222, Renton, WA, July 2019. USENIX Association.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[7] Julaiti Alafate and Yoav Freund. Tell me something new: A new framework for asynchronous parallel learning. *Preprint*, 2018.

[8] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1709–1720. Curran Associates, Inc., 2017.

[9] Jie Shen and Ping Li. Partial hard thresholding: Towards a principled analysis of support recovery. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3124–3134. Curran Associates, Inc., 2017.

[10] Michael Kamp, Mario Boley, Olana Missura, and Thomas Gärtner. Effective parallelisation for machine learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6477–6488. Curran Associates, Inc., 2017.

[11] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7564–7575. Curran Associates, Inc., 2018.

[12] Soojeong Kim, Eunji Jeong, Joo Seong Jeong, Gyeong-In Yu, Hojin Park, and Byung-Gon Chun. Auto-parallelizing deep learning for multi-machine, multi-gpu environments. In *Workshop on AI Systems at Symposium on Operating Systems Principles (SOSP)*, 2017.

[13] Hang Shi, Yue Zhao, Bofeng Zhang, Kenji Yoshigoe, and Athanasios V. Vasilakos. A free stale synchronous parallel strategy for distributed machine learning. In *Proceedings of the 2019 International Conference on Big Data Engineering (BDE 2019) - BDE 2019*. ACM Press, 2019.

[14] Aaron Harlap, Henggang Cui, Wei Dai, Jinliang Wei, Gregory R. Ganger, Phillip B. Gibbons, Garth A. Gibson, and Eric P. Xing. Addressing the straggler problem for iterative convergent parallel ml. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, SoCC '16, pages 98–111, New York, NY, USA, 2016. ACM.

[15] Constantinos Daskalakis, Nishanth Dikkala, and Siddhartha Jayanti. Hogwild!-gibbs can be panaccurate. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 32–41. Curran Associates, Inc., 2018.

[16] Celestine Dünner, Thomas Parnell, Dimitrios Sarigiannis, Nikolas Ioannou, Andreea Anghel, Gummadi Ravi, Madhusudanan Kandasamy, and Haralampos Pozidis. Snap ml: A hierarchical framework for machine learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 252–262. Curran Associates, Inc., 2018.

[17] Farshid Farhat, Diman Zad Tootaghaj, Yuxiong He, Anand Sivasubramaniam, Mahmut Kandemir, and Chita R. Das. Stochastic modeling and optimization of stragglers. *IEEE Transactions on Cloud Computing*, 6(4):1164–1177, oct 2018.

[18] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association.

[19] Bojie Li, Zhenyuan Ruan, Wencong Xiao, Yuanwei Lu, Yongqiang Xiong, Andrew Putnam, Enhong Chen, and Lintao Zhang. Kv-direct: High-performance in-memory key-value store with programmable nic. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 137–152, New York, NY, USA, 2017. ACM.

[20] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 153–167, New York, NY, USA, 2017. ACM.

[21] Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, feb 2009.

[22] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4424–4434. Curran Associates, Inc., 2017.

[23] Rashish Tandon, Qi Lei, Alexandros G. Dimakis, and Nikos Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3368–3376, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[24] Jack Kosaian, KV Rashmi, and Shivaram Venkataraman. Parity models: Erasure-coded resilience for prediction serving systems. In *Proceedings of the 27th Symposium on Operating Systems Principles*, SOSP '19, 2019.

[25] Hsiang-Fu Yu, Cho-Jui Hsieh, and Inderjit S. Dhillon. Parallel asynchronous stochastic coordinate descent with auxiliary variables. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2641–2649. PMLR, 16–18 Apr 2019.

[26] János Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *6th Prague Conference on Information Theory*, pages 411–425, 1973.

[27] Soheil Feizi and Muriel Medard. On network functional compression. *IEEE Transactions on Information Theory*, 60(9):5387–5401, sep 2014.

[28] Maya Kabkab, Azadeh Alavi, and Rama Chellappa. Dcnns on a diet: Sampling strategies for reducing the training set size. *preprint*, 2016.

[29] János Körner and Alon Orlitsky. Zero-error information theory. *IEEE Transactions on Information Theory*, 44(6):2207–2229, 1998.