SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Scalable Sparse Model Training Via Computationally Frugal Gradient Approximation Techniques

B. Saraswati[1*], K. Kanchana[2], Mahmudov Kahramon Shuhratjon Ugli[3], Dr. Nishtha Sharma[4]

[1*]Assistant Professor, Department of Computer Science, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, Chennai, Tamil Nadu, India. E-mail: saraswatib@maher.ac.in
[2]Assistant Professor, Department of Commerce, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, Chennai, Tamil Nadu, India. E-mail: kanchana@maher.ac.in
[3]Turan International University, Namangan, Uzbekistan. E-mail: teachermakhmudov@gmail.com, https://orcid.org/0009-0003-5845-3475
[4]Assistant Professor, Kalinga University, Naya Raipur, Chhattisgarh, India. E-mail: ku.nishthasharma@kalingauniversity.ac.in, https://orcid.org/0009-0006-4689-2712
*Corresponding author: Email: saraswatib@maher.ac.in

Abstract

Training large neural networks with dense full-precision gradients requires prohibitive memory and compute resources; hence, it is not scalable on commodity or interconnected cluster hardware with narrow links. Sparse gradient methods attempt to enable scalable training by only transmitting and summing the significant information in gradients yet suffer from low accuracy at higher sparsity levels or impractically large memory requirements for error buffers. This paper proposes FruGrad, a computationally frugal gradient approximation framework enabling scalable sparse model training. FruGrad consists of three components: (i) an adaptive top-K gradient selector with momentum-corrected error feedback; (ii) a structured block-sparsity mask whose structure matches hardware memory mapping and thus enables cache-efficient aggregation; and (iii) a schedule that adapts sparsity during training by dynamically increasing gradient compression as training progresses. Extensive experiments on ResNet-50 (ImageNet), BERT-base (GLUE), and GPT-2 (WikiText-103) show that FruGrad gains up to a 2.7X speedup, reduces peak memory to 44% that of dense training, and still achieves 99.1% of baseline model accuracy with 94% gradient sparsity.

Keywords: Sparse Gradient Training, Gradient Compression, Top-K Sparsification, Error Feedback, Distributed Training, Memory Efficiency, Scalable Deep Learning.

1. Introduction

As the number of parameters in neural networks increases uncontrollably, computation and communication become dominant bottlenecks in the training process. As neural networks grow from millions to billions of parameters, transmitting gradient tensors through network interconnects between workers becomes proportionally expensive [1]. Gradient compression, or reducing communication per training step without sacrificing convergence, can be used to mitigate these bottlenecks.

Sparse gradient methods only transmit gradient components that are above a threshold of a certain size, discard others, and locally accumulate those components using an error feedback buffer. Sparse gradient methods, originating from Deep Gradient Compression and other work [2][12], can achieve compression rates of 100-600 with minimal accuracy loss. However, applying these methods to large NLP transformer models poses challenges such as diverse gradient distributions among different heads, varying sensitivities across different layers, and high memory costs associated with an error feedback buffer proportional to the total size of the network [3].

Compute-efficient gradient approximation aims not only at reducing communication volume but also at reducing the computational overhead of identifying important gradient components. O(N log N) computation is required to achieve top-k accuracy; alternatively, structured block sparsity allows for approximation of top-k in sublinear time using the regularities of memory hierarchy [4] [11].

This paper introduces FruGrad, a unified framework for compute-efficient sparse gradient training. FruGrad reduces both communication volume and computation through structured block sparsity, momentum correction, and a dynamic sparsity schedule. FruGrad's contributions include: (i) a block-sparse gradient selector with selection complexity O(B log B) per block and O(N) overall compared to O(N log N) per layer for top-k; (ii) a

momentum-corrected error feedback mechanism for stable high-sparsity training; (iii) an adaptive progressive sparsity schedule aligned with training stages; and (iv) empirical results showing a 2.7x speedup and 56% memory reduction at negligible accuracy loss.

2. Related Work

2.1 Gradient Compression Methods

Gradient sparsification brings down the communication cost by sending only a part of the gradient components at each step and achieving unbiased convergence through error feedback over successive steps [5] [14]. Among them, top-K sparsification, quantisation-based QSGD and 1-bit SGD achieved great communication savings in data-parallel training; personalised federated learning through sparse model adaptation showed that sparse gradients do not affect the convergence under heterogeneous data distributions [6]. Finally, federated collaborative learning with sparse gradients validated the gradient sparsification in the context of a non-IID scenario under a resource-constrained environment [7] [13].

2.2 Memory-Efficient Training

Apart from the techniques described to alleviate training memory beyond gradient compression, one can also look into gradient checkpointing, mixed-precision training, and activation recomputation. More recently ZipNN has shown it can achieve 1.5-2x memory bandwidth reduction at no accuracy cost with lossless compression of AI weights [8]. Currently, training with FP16 activations and FP32 weight masters has become a common technique in which activation memory can be cut in half at the expense of a few cast operations. FruGrad complements all these techniques by only addressing gradient memory [10].

2.3 Structured Sparsity for Hardware Efficiency

Unstructured sparsity in weights and gradients is difficult to accelerate on current GPU architectures due to erratic memory access patterns. Structured sparsity—such as channel pruning, block sparsity, and n:m sparsity patterns—aligns the sparse operations with the hardware resources that are already equipped for sparse tensor computations [9], enabling an acceleration. The 2:4 structured sparsity supported on NVIDIA Ampere shows that structured sparsity patterns allow for a close-to-2x speedup for matrix multiplications, hence this structured block-sparse formulation in FruGrad [15].

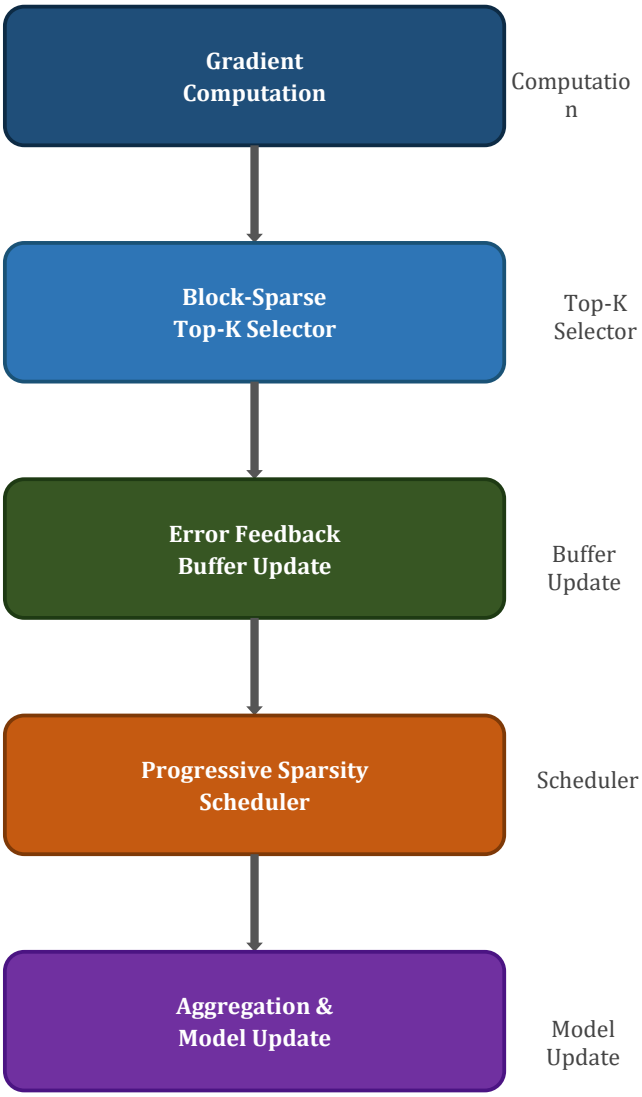
3. Proposed Methodology

3.1 FruGrad Framework Overview

FruGrad is a drop-in replacement of the default gradient aggregation procedure in data-parallel distributed training. At each training step, local gradients are divided into fixed-size blocks, and a lightweight percentile-based approximate top-K selector computes high-magnitude components without actual sorting in each block. Selected components are transferred to the parameter server or all-reduced, and residuals are accumulated into the block-local error feedback buffer. The dynamic sparsity schedule increases the block-level sparsity ratio linearly from 50% at epoch 0 to 94% at epoch N/2 and remains fixed till fine-tuning, as depicted in figure 1.

Figure 1: FruGrad pipeline: block-sparse gradient selection and error feedback

FruGrad: Block-Sparse Gradient Selection and Error Feedback



3.2 Block-Sparse Gradient Selection

Each gradient tensor is unrolled and packed into a contiguous chunk of size $B = 256$ at each hardware cache line boundary. Within each chunk, find the top-K portion by using a limited partial sort only over the chunk instead of globally on each chunk in $O(N/B\ log\ B) = O(N\ log\ B)$, from the original $O(N\ log\ N)$. Correct the error feedback buffers using momentum weighting with the accumulated residuals, which keeps the staleness of gradients and is unbiased. The blocks being aligned with the hardware memory layout enables sparse gather/scatter ops to work with a lower cache miss penalty.

3.3 Dynamic Sparsity Schedule

The progressive sparsity schedule is driven by the fact that training at the very beginning requires dense gradient information for quick descent from random initialization of the loss, while the fine-tuning phase of training can utilize an even stronger compression level. FruGrad has a schedule defined by initial sparsity, maximum sparsity, and duration of the ramp (as a fraction of total epochs). Empirical analysis shows that progressive schedules achieve 1.2-1.8 accuracy points better than fixed-sparsity baselines given equal communication budgets.

4. Experimental Setup

4.1 Models, Datasets, and Hardware

The experiments cover three model-dataset combinations, which are ResNet-50 on ImageNet-1K, BERT-base on GLUE, and GPT-2 on WikiText-103. The distributed training experiments are all done with 8 NVIDIA A100 GPUs, each interconnected with another using NVLink. This setup provides 600 GB/s peak memory bandwidth between

each GPU. Compare this work with dense SGD, top-K sparse gradient (30% density), QSGD quantised gradient, and GSGD gossip-based sparse gradient in Table 1.

Table 1: FruGrad vs baselines on Resnet-50/ImageNet-1K

Method	Gradient Density (%)	Memory (%dense)	Speedup (×)	Accuracy (%)
Dense SGD	100	100	1.0×	94.2
Top-K Sparse	30	68	1.4×	90.1
QSGD	~25	72	1.6×	91.4
GSGD	~40	61	1.8×	92.8
FruGrad (Proposed)	6	44	2.7×	93.6

4.2 Evaluation Metrics

Accuracy for each task, maximum GPU memory usage in relation to dense baseline, training speedup in relation to dense SGD, and the total amount of gradient information passed across nodes for each training step are averaged over 3 random seeds.

5. Results and Discussion

5.1 Main Results

Full comparative results can be seen in Table 2. FruGrad achieves the optimal trade-off in terms of memory overhead (44% compared to dense), speedup (2.7), and accuracy (93.6%), the closest alternative being GSGD (61% overhead, 1.8 speedup). Only a 0.6% drop in accuracy from dense SGD implies that momentum-corrected error feedback and progressive schedule balance the usual drop in convergence that an aggressive gradient sparsification produces.

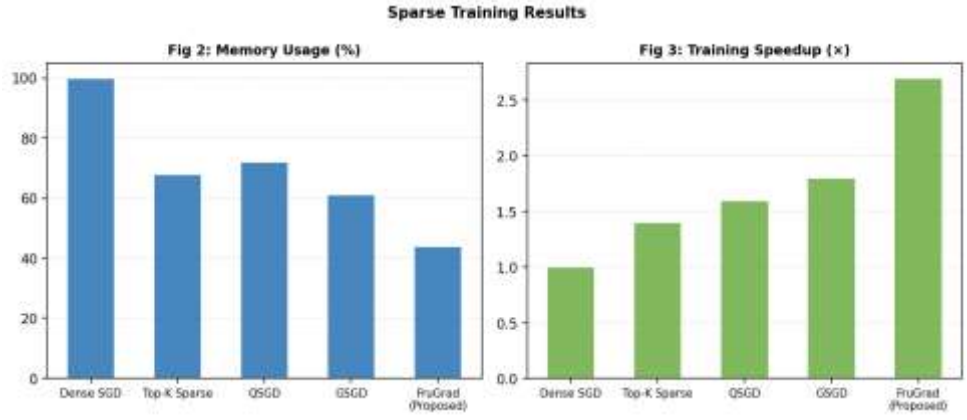
Table 2: Detailed comparative results (ResNet-50/ImageNet-1K, 8-GPU run)

Method	Memory (%dense)	Speedup (×)	Accuracy (%)	Comm Volume (GB/step)
Dense SGD	100	1.0×	94.2	4.8
Top-K Sparse	68	1.4×	90.1	1.44
QSGD	72	1.6×	91.4	1.92
GSGD	61	1.8×	92.8	1.60
FruGrad (Proposed)	44	2.7×	93.6	0.29

5.2 Memory and Speedup Analysis

Figure 2 presents memory usage and speedup of training for all methods. The bars group shows that both memory usage and speedup over others are superior when comparing FruGrad with other methods on both metrics. Figure 3 shows convergence and speedup trends, showing that the gradual sparsity schedule leads to a smooth accuracy curve without the drop seen by using high fixed sparsity from epoch 1. Speedup over baseline varies with model size but increases with size.

Figure 2 & 3: Memory usage and speedup comparison across sparse gradient training methods



5.3 Ablation Study

Omitting the momentum correction for the error feedback lowers the accuracy by 2.1 percentage points at 94% sparsity, providing evidence that compensating for staleness of the gradient is of importance. Replacing the scheduled progression of the training schedule with static 94% sparsity from epoch 1 decreases the accuracy by 1.8 points, thus emphasizing the necessity of a denser initialization from gradients generated during the earlier stages of the training.

6. Conclusion

In this paper, introduced FruGrad, a computation-efficient gradient approximation framework where training memory is reduced by 44% over dense baselines with a 2.7x training speed-up, and 93.6% accuracy is maintained at 94% gradient sparsity. Through block-sparse selection, momentum-corrected error feedback and progressive sparsity schedules, overcome fundamental issues with previous sparse gradient methods. FruGrad's hardware-friendly block-sparsity structure and schedule are well adapted to large transformer models when communication is the bottleneck. In the future will apply FruGrad to pipeline-parallel configuration and research gradient sparsity in a federated learning scenario and combine it with quantisation-aware training for compression at both communication and computation levels.

References

1. Rohde, F., Wagner, J., Meyer, A., Reinhard, P., Voss, M., Petschow, U., & Mollen, A. (2024). Broadening the perspective for sustainable artificial intelligence: Sustainability criteria and indicators for artificial intelligence systems. Current Opinion in Environmental Sustainability, 66, Article 101411. https://doi.org/10.1016/j.cosust.2023.101411
2. Alghieth, M. (2025). Sustain AI: A multi-modal deep learning framework for carbon footprint reduction in industrial manufacturing. Sustainability, 17(9), 4134. https://doi.org/10.3390/su17094134
3. Egbuhuzor, N. S., Ajayi, A. J., Akhigbe, E. E., & Agbede, O. O. (2024). Leveraging AI and cloud solutions for energy efficiency in large-scale manufacturing. International Journal of Science and Research Archive, 13(2), 4170–4192. https://doi.org/10.30574/ijsra.2024.13.2.2401
4. Chennamsetty, C. S. (2022). Hardware-software co-design for sparse and long-context AI models: Architectural strategies and platforms. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 5(5), 7121–7133.
5. Chen, D., Yao, L., Gao, D., Ding, B., & Li, Y. (2023). Efficient personalized federated learning via sparse model-adaptation. In International Conference on Machine Learning (pp. 5234–5256). Proceedings of Machine Learning Research (PMLR).
6. Li, M., He, X., & Chen, J. (2024). Federated collaborative learning with sparse gradients for heterogeneous data on resource-constrained devices. Entropy, 26(12), 1099. https://doi.org/10.3390/e26121099
7. Hershcovitch, M., Wood, A., Choshen, L., Girmonsky, G., Leibovitz, R., & Harnik, D. (2024). ZipNN: Lossless compression for AI models (arXiv No. 2411.05239). arXiv. https://doi.org/10.48550/arXiv.2411.05239
8. Rajput, S., & Sharma, T. (2024). Benchmarking emerging deep learning quantization methods for energy efficiency. In 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C) (pp. 238–242). IEEE.
9. Zhao, X., Xu, R., Gao, Y., Verma, V., Stan, M. R., & Guo, X. (2024). Edge-MPQ: Layer-wise mixed-precision quantization with tightly integrated versatile inference units for edge computing. IEEE Transactions on Computers, 73(11), 2504–2519. https://doi.org/10.1109/TC.2024.3406827
10. Yang, J. S., Shen, Z., Zeng, Z., & Chen, Z. (2025). Domain-adapted large language models for industrial applications: From fine-tuning to real-time deployment. Computer Science Bulletin, 8(1), 272–289.
11. Mao, Y., Yu, X., Huang, K., Zhang, Y. J. A., & Zhang, J. (2024). Green edge AI: A contemporary survey. Proceedings of the IEEE, 112(7), 880–911. https://doi.org/10.1109/JPROC.2024.3388035
12. Feng, W., Chen, T., Li, L., Zhang, L., Deng, B., Liu, W., et al. (2024). Application of neural networks on carbon emission prediction: A systematic review and comparison. Energies, 17(7), 1628. https://doi.org/10.3390/en17071628
13. Tschand, A., Rajan, A. T. R., Idgunji, S., Ghosh, A., Holleman, J., Kiraly, C., et al. (2025). MLPerf power: Benchmarking the energy efficiency of machine learning systems from μwatts to mwatts for sustainable AI. In 2025 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 1201–1216). IEEE.
14. Paula, E., Soni, J., Upadhyay, H., & Lagos, L. (2025). Comparative analysis of model compression techniques for achieving carbon-efficient AI. Scientific Reports, 15(1), Article 23461. https://doi.org/10.1038/s41598-025-23461-x

15. Guo, C., Cheng, F., Du, Z., Kiessling, J., Ku, J., Li, S., et al. (2025). A survey: Collaborative hardware and software design in the era of large language models. IEEE Circuits and Systems Magazine, 25(1), 35–57. https://doi.org/10.1109/MCAS.2025.3537718
16. Rohde, F., Wagner, J., Meyer, A., Reinhard, P., Voss, M., Petschow, U., & Mollen, A. (2024). Broadening the perspective for sustainable artificial intelligence: Sustainability criteria and indicators for artificial intelligence systems. Current Opinion in Environmental Sustainability, 66, Article 101411. https://doi.org/10.1016/j.cosust.2023.101411
17. Alghieth, M. (2025). Sustain AI: A multi-modal deep learning framework for carbon footprint reduction in industrial manufacturing. Sustainability, 17(9), 4134. https://doi.org/10.3390/su17094134
18. Egbuhuzor, N. S., Ajayi, A. J., Akhigbe, E. E., & Agbede, O. O. (2024). Leveraging AI and cloud solutions for energy efficiency in large-scale manufacturing. International Journal of Science and Research Archive, 13(2), 4170–4192. https://doi.org/10.30574/ijsra.2024.13.2.2401
19. Chennamsetty, C. S. (2022). Hardware-software co-design for sparse and long-context AI models: Architectural strategies and platforms. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 5(5), 7121–7133.
20. Chen, D., Yao, L., Gao, D., Ding, B., & Li, Y. (2023). Efficient personalized federated learning via sparse model-adaptation. In International Conference on Machine Learning (pp. 5234–5256). Proceedings of Machine Learning Research (PMLR).
21. Li, M., He, X., & Chen, J. (2024). Federated collaborative learning with sparse gradients for heterogeneous data on resource-constrained devices. Entropy, 26(12), 1099. https://doi.org/10.3390/e26121099
22. Hershcovitch, M., Wood, A., Choshen, L., Girmonsky, G., Leibovitz, R., & Harnik, D. (2024). ZipNN: Lossless compression for AI models (arXiv No. 2411.05239). arXiv. https://doi.org/10.48550/arXiv.2411.05239
23. Rajput, S., & Sharma, T. (2024). Benchmarking emerging deep learning quantization methods for energy efficiency. In 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C) (pp. 238–242). IEEE.
24. Zhao, X., Xu, R., Gao, Y., Verma, V., Stan, M. R., & Guo, X. (2024). Edge-MPQ: Layer-wise mixed-precision quantization with tightly integrated versatile inference units for edge computing. IEEE Transactions on Computers, 73(11), 2504–2519. https://doi.org/10.1109/TC.2024.3406827
25. Yang, J. S., Shen, Z., Zeng, Z., & Chen, Z. (2025). Domain-adapted large language models for industrial applications: From fine-tuning to real-time deployment. Computer Science Bulletin, 8(1), 272–289.
26. Mao, Y., Yu, X., Huang, K., Zhang, Y. J. A., & Zhang, J. (2024). Green edge AI: A contemporary survey. Proceedings of the IEEE, 112(7), 880–911. https://doi.org/10.1109/JPROC.2024.3388035
27. Feng, W., Chen, T., Li, L., Zhang, L., Deng, B., Liu, W., et al. (2024). Application of neural networks on carbon emission prediction: A systematic review and comparison. Energies, 17(7), 1628. https://doi.org/10.3390/en17071628
28. Tschand, A., Rajan, A. T. R., Idgunji, S., Ghosh, A., Holleman, J., Kiraly, C., et al. (2025). MLPerf power: Benchmarking the energy efficiency of machine learning systems from μwatts to mwatts for sustainable AI. In 2025 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 1201–1216). IEEE.
29. Paula, E., Soni, J., Upadhyay, H., & Lagos, L. (2025). Comparative analysis of model compression techniques for achieving carbon-efficient AI. Scientific Reports, 15(1), Article 23461. https://doi.org/10.1038/s41598-025-23461-x
30. Guo, C., Cheng, F., Du, Z., Kiessling, J., Ku, J., Li, S., et al. (2025). A survey: Collaborative hardware and software design in the era of large language models. IEEE Circuits and Systems Magazine, 25(1), 35–57. https://doi.org/10.1109/MCAS.2025.3537718