Exploring Efficient Deep Learning Training on AI Accelerators

NC STATE UNIVERSITY

Department of Electrical and Computer Engineering

Milan Shah, Advisor: Michela Becchi

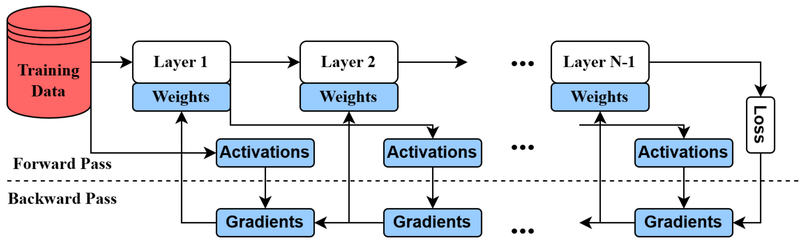
Introduction

Since the inception of deep neural network (DNN) training on GPUs with AlexNet, DNNs have shown growing promise for machine learning tasks. Models have exploded in size, from tens of millions to hundreds of *billions* of parameters, giving way to larger computational and memory requirements for training [1]. Several companies have created novel AI chips tuned for training and inference to challenge GPU-supremacy in DNN training, including the Cerebras CS-2/3, Graphcore IPU, SambaNova SN30, and Groq GroqChip. These accelerators feature large on-chip memory capacity and massive parallelism capabilities, attempting to address on-chip/off-chip memory exchange bottlenecks encountered on GPU and CPU. While lossy compression has been employed as a tool to reduce the memory footprint of DNN training on GPU, the potential of lossy compression across both GPU *and* novel AI accelerators has not been thoroughly studied.

Research Questions

- Why and how can compression work well on novel AI accelerators?
- What is the potential of activation memory reduction on an accelerator?
- How can activation footprint management be improved on the GPU? **(On-going)**
- How do different DNN architectures perform on a novel accelerator? **(On-going)**

Background



Important data structures in DNN training and potential targets for compression

Emergent Lossy Compressors			
Type	**Parameter**	**Guarantee**	**Example**
Error-bounded	Error-bound ϵ	$x' = x \pm \epsilon$	**SZ**: Prediction → Quant. → Lossless Encoding
Fixed-Rate	Target bitrate R	$size(c) = \frac{size(x)}{R}$	**ZFP**: Transform → Lossy Encoding

Examples of lossy compressors (SZ [2], ZFP [3]) for scientific data

Challenges

Codebases

Lossy Compression

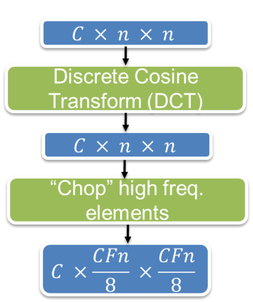
- Preserve model accuracy
- High enough throughput
- Greater CR relative to quantization
- Compressor parameter selection

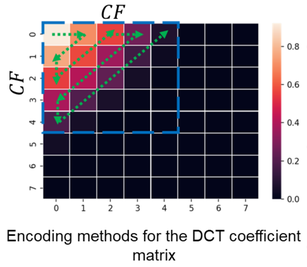
Programming AI Accelerators

- Fixed tensor sizes
- Limited operator support
- Efficient code for platform
- Data structure access

- Lossy compression **requires compute** and introduces **distortion** of the data
- **Stricter code requirements** on computational graph due to accelerator compilers
- Aim for **high CR** and **high throughput**

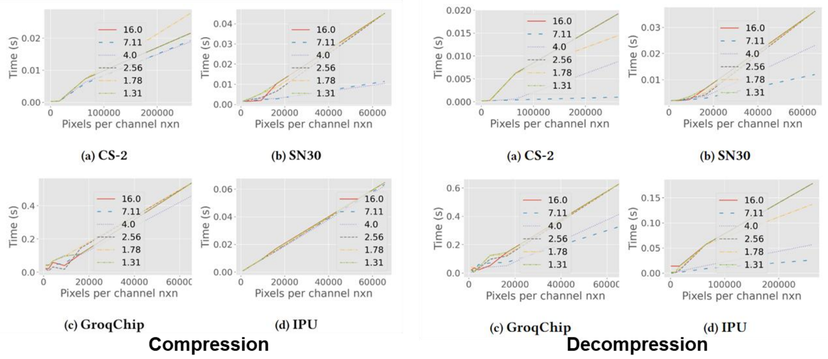
Training Data on AI Accelerators





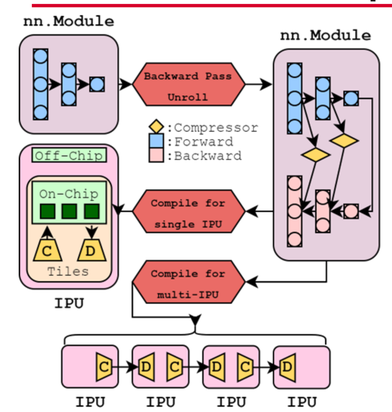
Encoding methods for the DCT coefficient matrix

- **DCT+Chop**: DCT from JPEG [4] as a training data compressor
- **Portable:** Implemented with only PyTorch operators
- **Fast:** Two matrix multiplications

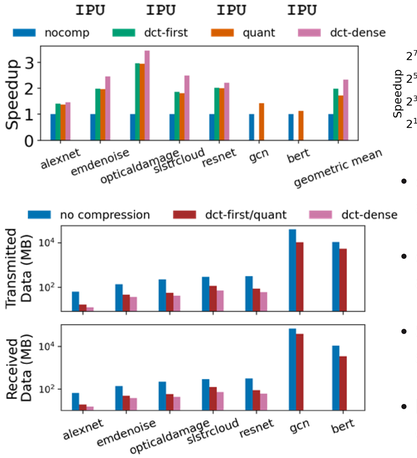


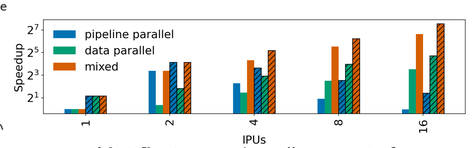
Throughput: 100s of MB/s to 10s of GB/s, Compression Ratios: 1.3 to 16

IPU Activation Compression



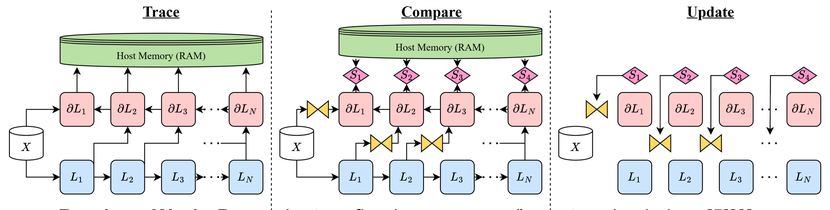
- **Graphcore IPU:** MIMD-style, 1472 compute units, 900 MB on-chip memory
- **Activation Access:** Given a user-specified forward pass, backward pass unrolling tool inserts custom PyTorch gradient operators to expose activation data for compression/decompression
- **Compression:** Apply any accelerator-compatible compressor to activations with the tool (DCT+Chop, symmetric quantization)
- **Benefit:** Reduces on-chip/off-chip exchange overhead which can improve training perf.





- **quant/dct-first**: quantize all or most of activations
- **dct-dense**: DCT+Chop for convolutional layer activations, quantize rest
- **Single IPU (figures to left):** 1.1-3.5X speedup, 5X less data exchange
- **Multi-IPU (figure above):** *dct-dense* outperforms no compression 1.9X

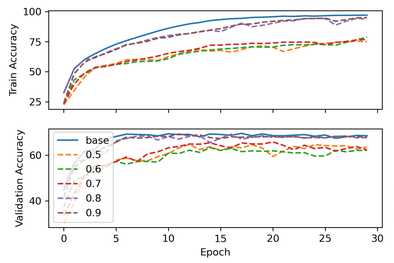
Adaptive GPU Activation Compression



- **Previous Work:** Dependent on fixed compressor/layer type/optimizer [5][6]

Proposed Design

- *Trace:* Store full precision gradients of each operator
- *Compare* : Cosine similarity between gradients from compressed activations and stored gradients.
- *Update:* Tune lossy compressor parameters if target cosine similarity score not met



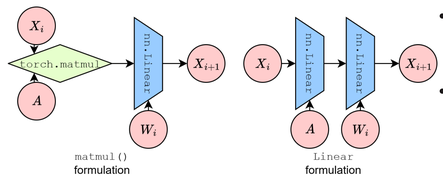
Preliminary Results

- Left: accuracy with varying average score for AlexNet/CIFAR10 with cuSZp
- Average score of 0.7+ within 1-2% of baseline, act. CR of up to 12X

On-going Work

- Varying DNNs (i.e. GNNs, transformer)
- Study combinations of different compressors

GNNs on the Cerebras CS-2



- **CS-2:** Wafer-scale, 850K compute units, dataflow architecture
- **Graph Neural Network (GNN):** Neural network that operates on graph data, typically stored in sparse formats like CSR
- **GNN on CS-2 (figure above):** Two PyTorch-based implementations to operate on dense format since PyTorch sparse tensors not supported. *Linear* order of magnitude faster than *matmul.* Dense format exhausts memory around 10^5 nodes
- **On-going work:** Low-level language (CSL) implementation to operate on sparse adjacency matrix, reducing memory utilization and data-loading overhead.

Conclusions

- Lossy compression ⇒ memory reduction for GPU and emerging accelerators
- Can reduce strain on on-chip/off-chip memory exchange overhead
- Emerging accelerators demand further study due to their potential for DNN training

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.

References

[1] Tu, X., He, Z., Huang, Y. et al. An overview of large AI models and their applications. Vis. Intell. 2, 34 (2024). https://doi.org/10.1007/s44267-024-00065-8

[2] Sheng Di and Franck Cappello. 2016. Fast Error-Bounded Lossy HPC Data Compression with SZ. In Proc. Of IPDPS '16 (IPDPS). 730–739. https://doi.org/10.1109/IPDPS.2016.11

[3] P. Lindstrom, "Fixed-Rate Compressed Floating-Point Arrays," in IEEE Transactions on Visualization and Computer Graphics, vol. 20, no. 12, pp. 2674-2683, 31 Dec. 2014, doi: 10.1109/TVCG.2014.2346458.

[4] "JPEG - JPEG 1." [Online]. Available: https://jpeg.org/jpeg/index.html

[5] X. Liu, L. Zheng, D. Wang, Y. Cen, W. Chen, X. Han, J. Chen, Z. Liu, J. Tang, J. Gonzalez, M. Mahoney, and A. Cheung, "GACT: Activation compressed training for genericnetwork architectures," in Proceedings of the 39th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp.14 139–14 152. [Online]. Available: https://proceedings.mlr.press/v162/liu22v.html

[6] J. Chen, L. Zheng, Z. Yao, D. Wang, I. Stoica, M. Mahoney, and J. Gonzalez, "ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training," in Proceedings of the 38th International Conference on Machine Learning. PMLR, Jul. 2021, pp. 1803–1813, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v139/chen21z.html