GACT: Activation Compressed Training for Generic Network Architecture

Lily (Xiaoxuan) Liu, Lianmin Zheng, Dequan Wang, Yukuo Cen, Weice Chen, Xun Han, Jianfei Chen, Zhiyuan Liu, Jie Yang, Joseph E. Gonzalez, Michael W. Mahoney, Alvin Cheung

UC Berkeley, Tsinghua University

ICML 2022

AI and Memory Wall

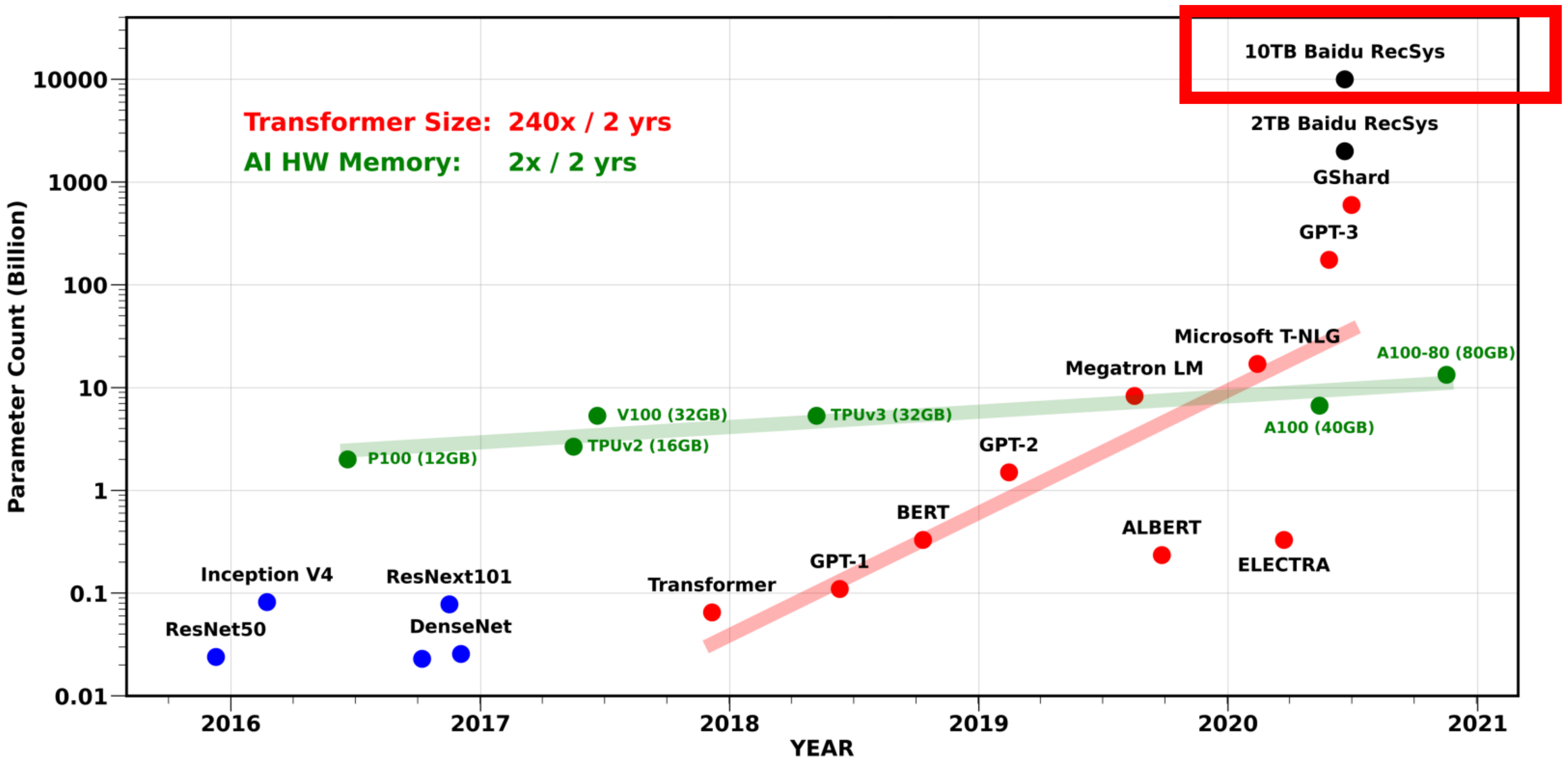
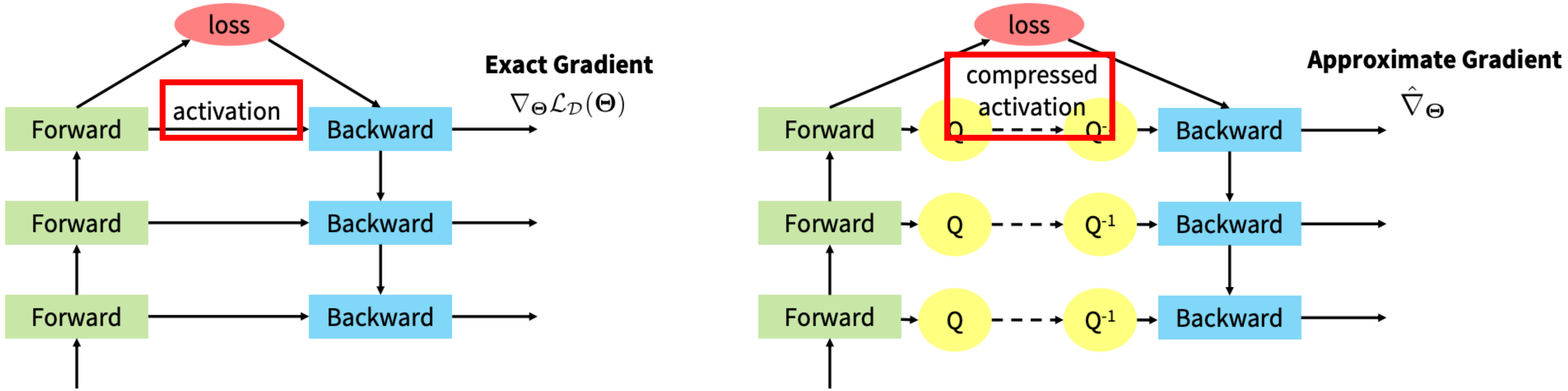


Figure credit: Gholami A, Yao Z, Kim S, Mahoney MW, Keutzer K. AI and Memory Wall. RiseLab Medium Blog Post, University of California Berkeley, 2021, March 29.

Activation Compressed Training (ACT)



Activation Compressed Training (ACT) is a promising approach to reduce the memory footprint.

$$\theta_{t+1} \leftarrow \theta_t - \eta g(Q(h(x;\theta_t));\theta_t)$$

Previous Work

Previous Work: A white box solution that is specific to network architecture and operator type.

- ActNN (CNN), Mesa (Vision Transformer), EXACT (GNN).

To support a new network architecture with new operators:

- 👎 Require to derive new convergence guarantee.
- 👎 Require ML experts to design compression schemes (e.g., bits/dim.).
- 👎 Require engineering effort to support for new operators.

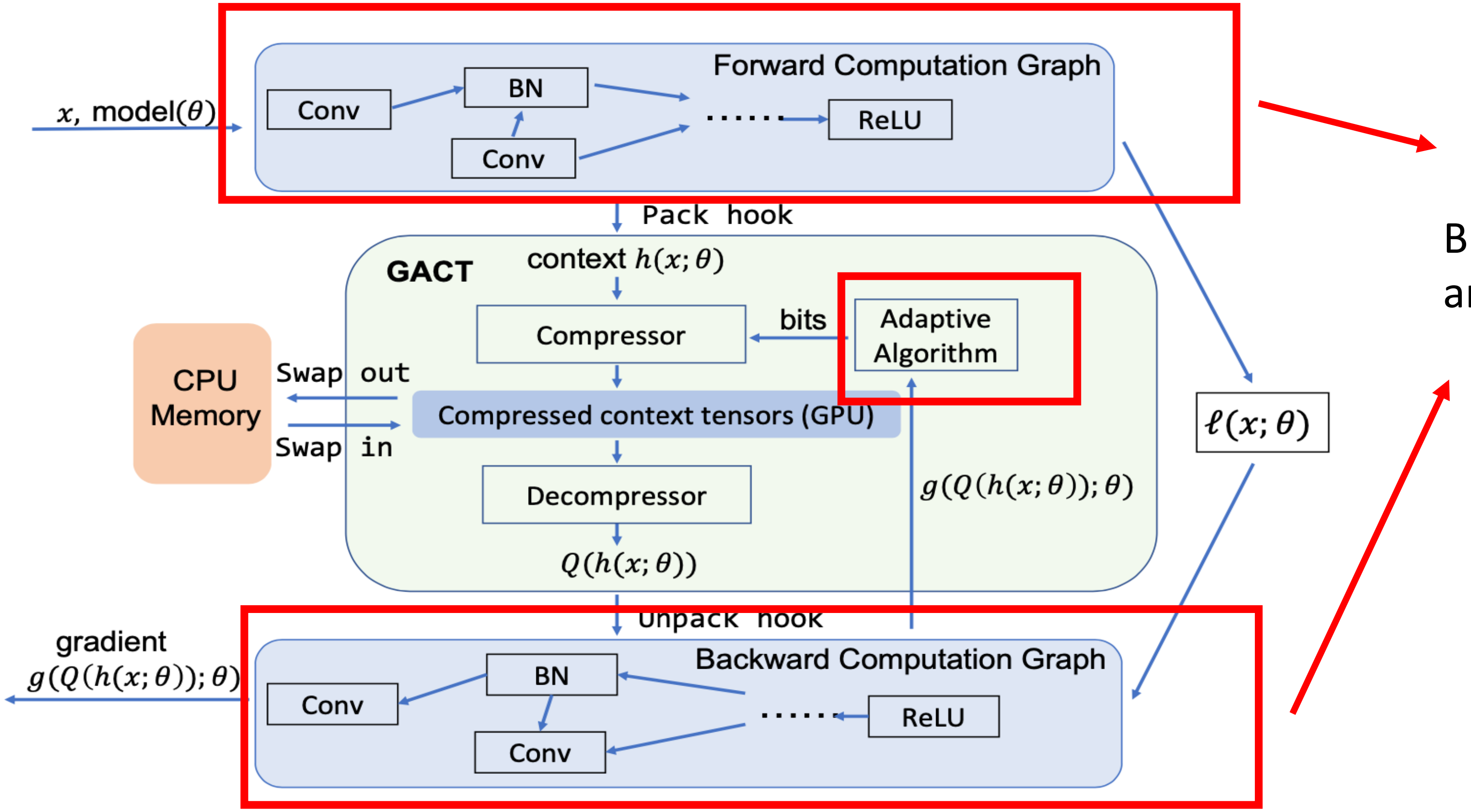
We want a general ACT framework that works with any network architecture and operator type!

Jianfei Chen, Lianmin Zheng, Zhewei Yao, Dequan Wang, Ion Stoica, Michael W Mahoney, and Joseph E Gonzalez. Actnn: Reducing training memory footprint via 2-bit activation compressed training. In International Conference on Machine Learning, 2021.
Zizheng Pan, Peng Chen, Haoyu He, Jing Liu, Jianfei Cai, and Bohan Zhuang. Mesa: A memory-saving training framework for transformers. arXiv preprint arXiv:2111.11124, 2021.
Anonymous. EXACT: Scalable graph neural networks training via extreme activation compression. In Submitted to The Tenth International Conference on Learning Representations, 2022.

Challenge & System Architecture

Developing a generic ACT framework is challenging:

- **Theory:** convergence guarantees must be made without assumptions on the network architecture.
- **Algorithm:** find effective compression strategies for all kinds of networks **automatically**.
- **System:** support arbitrary NN operations, including user-defined ones.



Black box solution that deals with any network architecture

Convergence of ACT

Key idea: Construct an unbiased approximation of the Activation Compressed (AC) gradient by linearizing the gradient function. Consider the first-order Taylor expansion of $g(\cdot;\theta)$ at h:

$$\hat{g}(Q(h); h, \theta) = g(h;\theta) + J(h,\theta)(Q(h) - h)$$

When the compression is accurate:

- The linearized gradient $\hat{g}$ is accurate
- The variance of the unbiased gradient dominates the linearized gradient variance

ACT behaves as if the activation compressed gradient is unbiased

Adapt the Compression Rate

Formalize the adaptive algorithm as an optimization problem: find the compression scheme to minimize the gradient variance given the bits constraint.

$$\min_{b} V(b; h; \theta), \qquad \text{s.t. } \sum_{i=1}^{L} b_l D_l \leq B$$

Use linearized approximation to make the problem solvable

$$\min_{b} \sum_{l=1}^{L} c_l(h, \theta) S(b_l), \qquad \text{s.t. } \sum_{i=1}^{L} b_l D_l \leq B$$

System Implementation

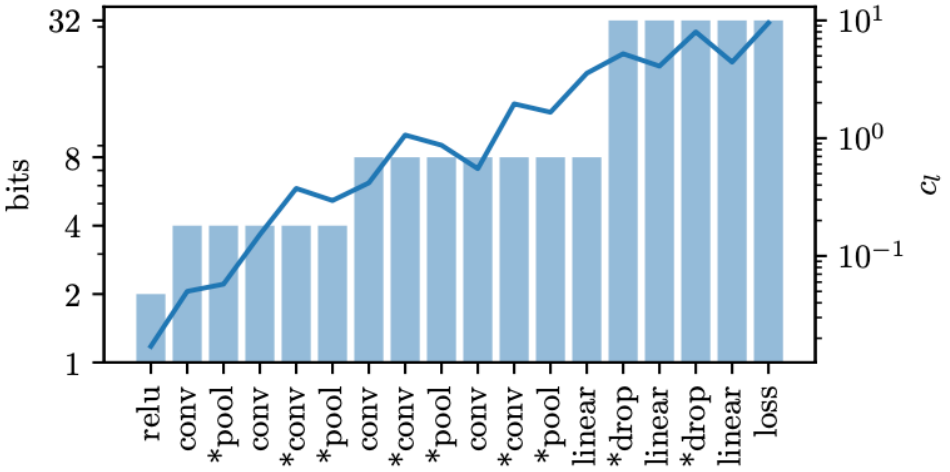
```python
import torch
import gact

model = ... # user defined model
controller = gact.controller(model, opt_level='L2')
controller.install_hook()

# training loop
for epoch in ...
  for iter in ...
    ......
    # instruct gact how to perform forward and backward
    def fwdbwdprop():
      output = model(data)
      loss = loss_func(output, target)
      optimizer.zero_grad()
      loss.backward()

    controller.iterate(fwdbwdprop)
```

- Implementation: pack_hook, unpack_hook

Three lines of modification in PyTorch

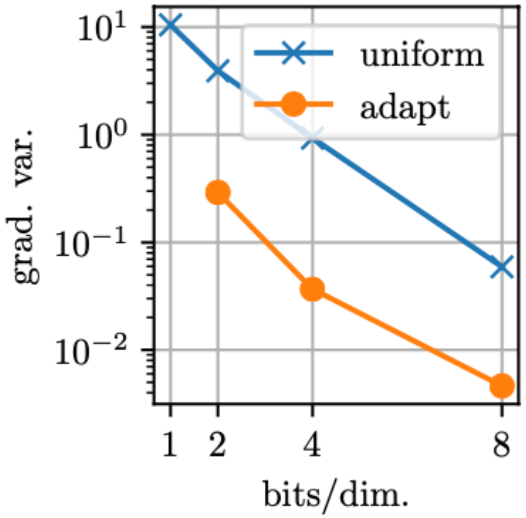
Experiments – Compression Strategy

Inferred per-tensor sensitivity and bits/dim.



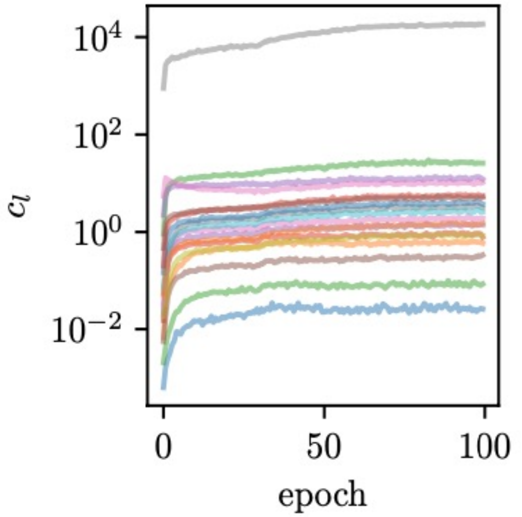
Assign bits based on tensor sensitivity.

Gradient Variance



With the adaptive algorithm, Var(adapt 4 bits/dim) < Var(uniform 8 bits/dim.).

Evolution of the per-tensor sensitivity



Sensitivity remains stable during training.

Experiments

- GACT can be applied to a wide range of deep learning tasks: Computer Vision, NLP, graph NN.
- GACT has negligible accuracy loss compared with full precision training.

Computer Vision:

Task	Model	FP32	GACT Adapt 4bit (L1)
Cls.	VGG11	68.75	68.77 (2.84×)
	ResNet-50	77.29	76.96 (6.69×)
	Swin-tiny	81.18	80.92 (7.44×)
Det.	Faster RCNN	37.4	37.0 (4.86×)
	RetinaNet	36.5	36.3 (3.11×)

Reduce activation by up to 8.1x!

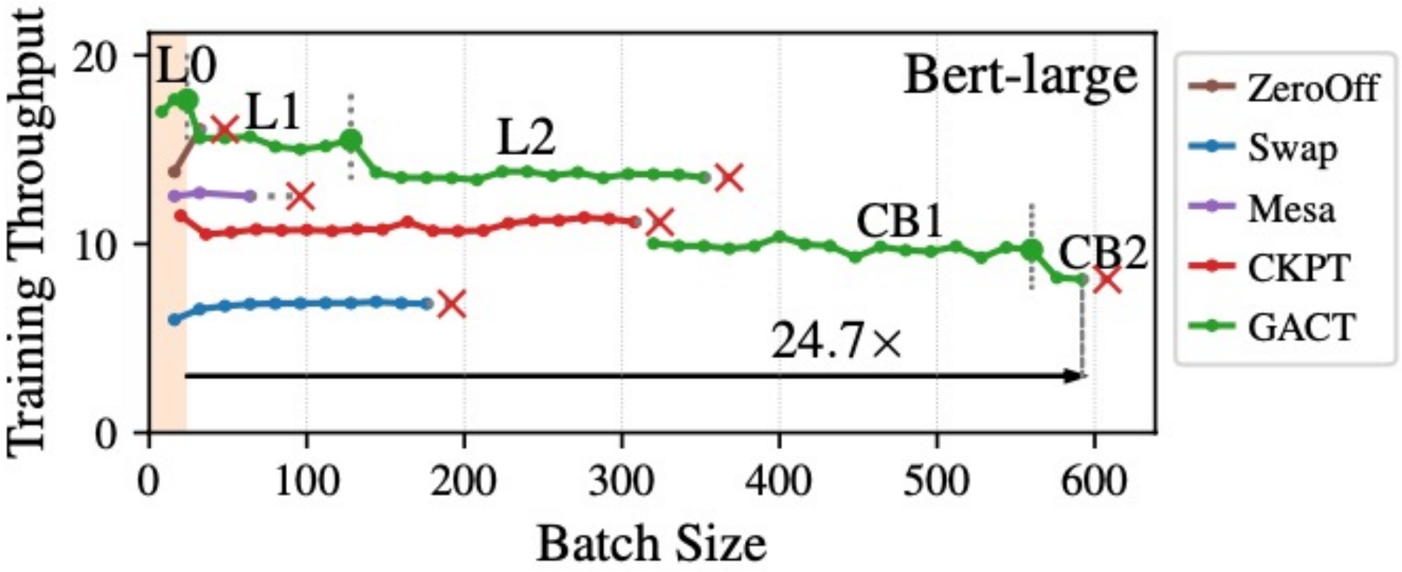
Graph NN:

NLP:

Model	Dataset	FP32	GACT Adapt 4bit (L1)
GCN	Flickr	51.17 ± 0.19	51.08 ± 0.18 (7.93×)
	Reddit	95.33 ± 0.07	95.32 ± 0.07 (7.90×)
	Yelp	39.86 ± 0.94	40.06 ± 0.74 (6.42×)
	ogbn-arxiv	71.51 ± 0.65	71.35 ± 0.36 (8.09×)
GAT	Flickr	52.40 ± 0.28	52.26 ± 0.31 (4.34×)
	Reddit	95.95 ± 0.06	96.02 ± 0.09 (4.29×)
	Yelp	52.41 ± 0.69	52.18 ± 0.38 (4.18×)
	ogbn-arxiv	71.68 ± 0.54	71.80 ± 0.47 (5.09×)
GCNII	Flickr	52.37 ± 0.16	52.31 ± 0.16 (4.91×)
	Reddit	96.32 ± 0.24	96.11 ± 0.22 (4.52×)
	Yelp	62.33 ± 0.20	62.28 ± 0.26 (5.34×)
	ogbn-arxiv	72.52 ± 0.12	72.28 ± 0.35 (6.74×)
Bert-large	MNLI	86.74 ± 0.24	86.61 ± 0.11 (7.38×)
	SST-2	93.69 ± 0.30	93.54 ± 0.52 (7.30×)
	MRPC	88.20 ± 0.02	87.90 ± 0.10 (7.40×)
	QNLI	92.29 ± 0.14	92.44 ± 0.07 (7.42×)

Experiments

Level	Compression Strategy	Bits
L0	Do not compress	32
L1	per-group quantization with auto-precision	4
L2	L1 + swapping/prefetching	4
CB1	L1 + gradient checkpointing	4
CB2	CB1 + efficient self-attention	4



- GACT can be combined with other memory-efficient training techniques (e.g. efficient-softmax, gradient checkpointing).
- GACT enables training with a **4.2**x to **24.7**x larger batch size.

Conclusion

- GACT: Reducing memory footprint by quantizing the activation
- Theory: Convergence guarantee for general networks
- Algorithm: Adaptive quantization techniques to find compression schemes automatically
- System: A Plug-and-Play PyTorch library that supports arbitrary NN operations
- GitHub: https://tinyurl.com/mr274yfs