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

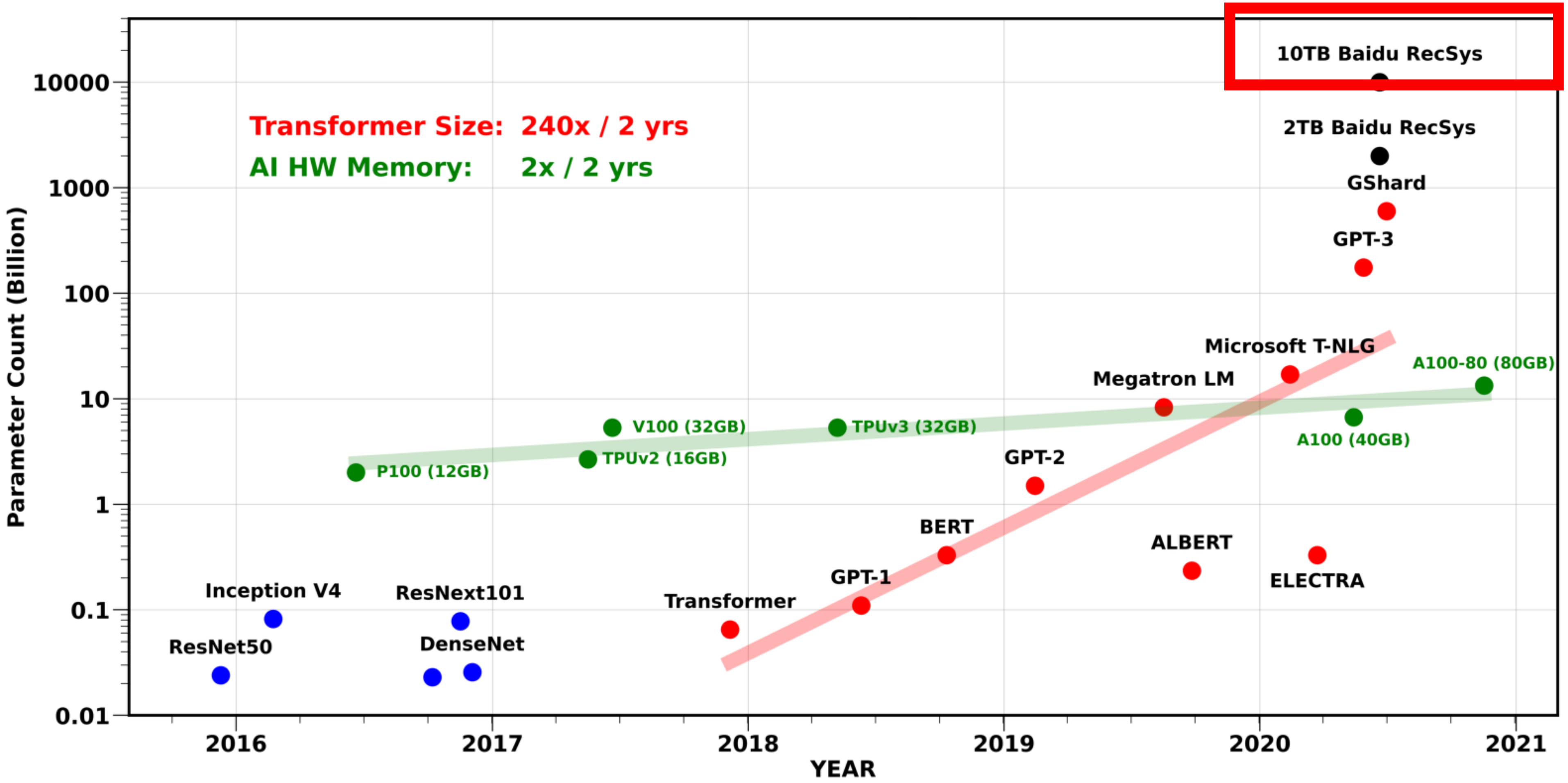
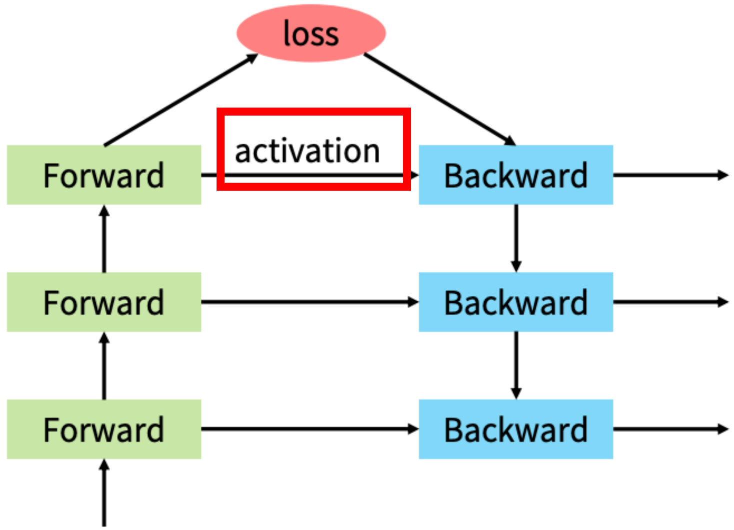
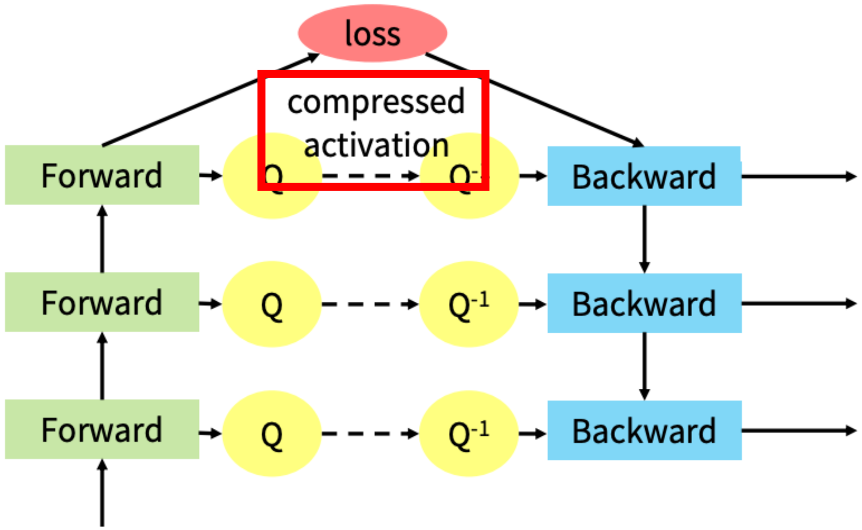


Figure credit: Gholami A, Yao Z, Kim S, Mahoney MW, Keutzer K. AI and Memory Wall. RiseLab Medium Blog Post, University of California Berkeley, 2021, March 29.

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.

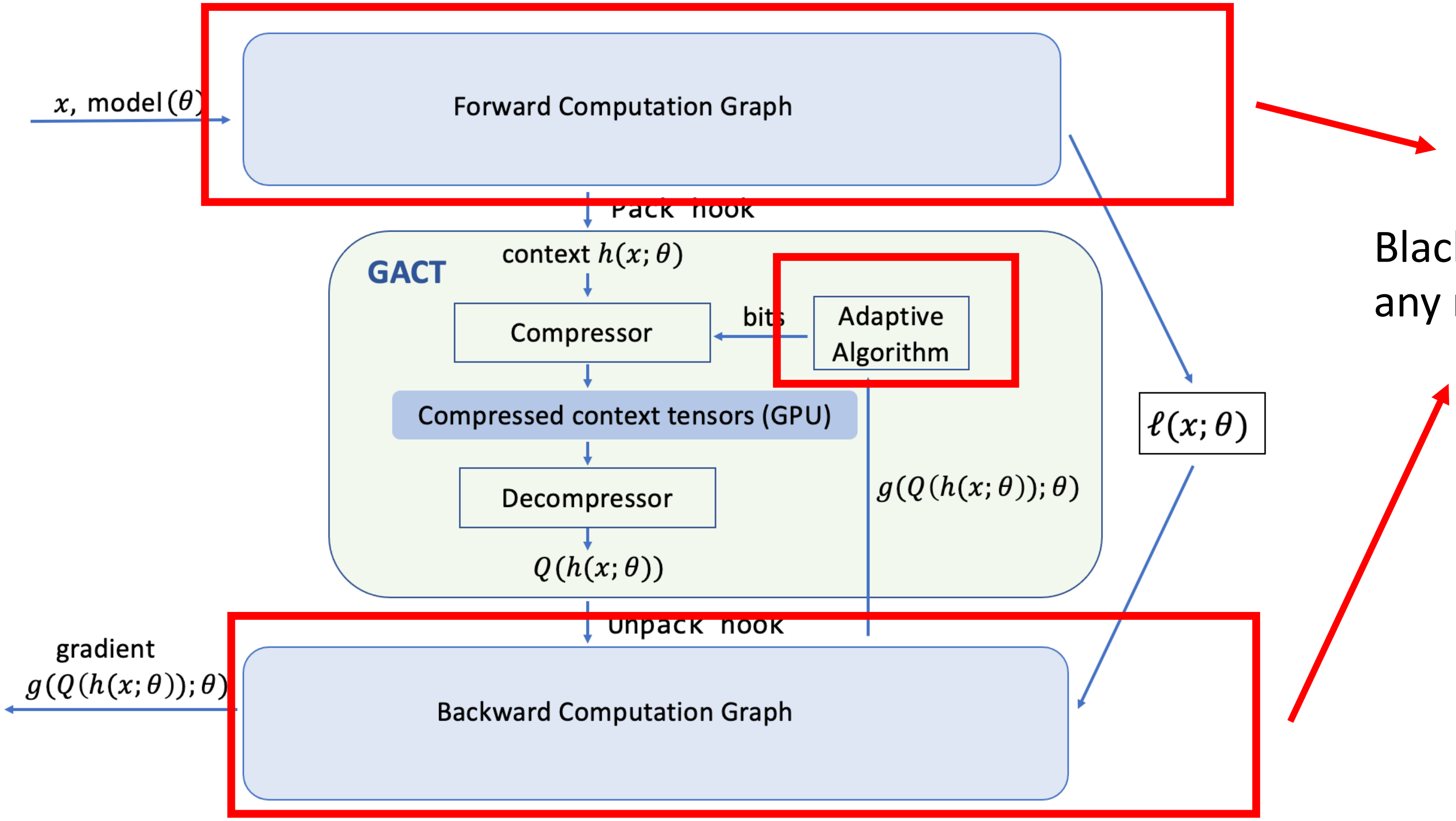
We want a general ACT framework that works with any network architecture and operator type!

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

For the first time, we prove ACT behaves as if the activation compressed gradient is unbiased for any network architecture!

Key idea: analyze a linear approximation of the Activation Compressed (AC) gradient. Consider the first-order Taylor expansion of the gradient function $g(\cdot;\theta)$ at activation h:

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

When the compression is unbiased and accurate ($Q(h) \approx h$):

- The linearized gradient $\hat{g}$ is unbiased.
- The approximation error $||\mathrm{E}(\hat{g} - g(Q(h))||$ is small compared to the gradient variance $\mathrm{Var}(\hat{g})$.

Adapt the Compression Rate

Some activations are very sensitive to compression (e.g. input of CrossEntropy loss).

Assign b_l (bits/dim) to lth activation tensor according to its sensitivity c_l.

Sensitivity c_l is computed on the fly automatically.

Find the compression scheme to minimize the gradient variance V given the bits constraint B.
Gradient variance has a simple **linear** form:

$$V \approx \min_{b} \sum_{l=1}^{L} c_l \; 2^{-2b_l}, \qquad \text{s.t.} \sum_{i=1}^{L} b_l D_l \leq B$$

System Implementation

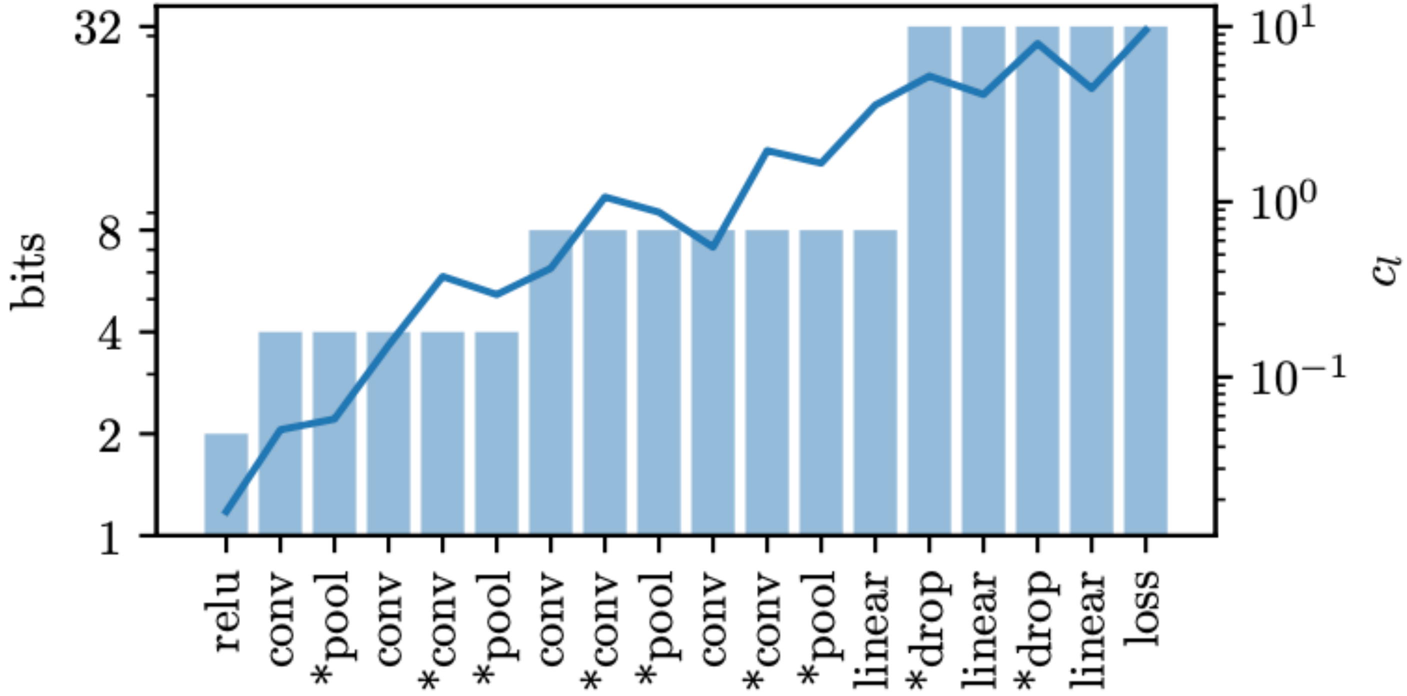
```
import torch
import gact

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)
```

Three lines of modification in PyTorch

- Implementation: pack_hook, unpack_hook

Experiments – Compression Strategy

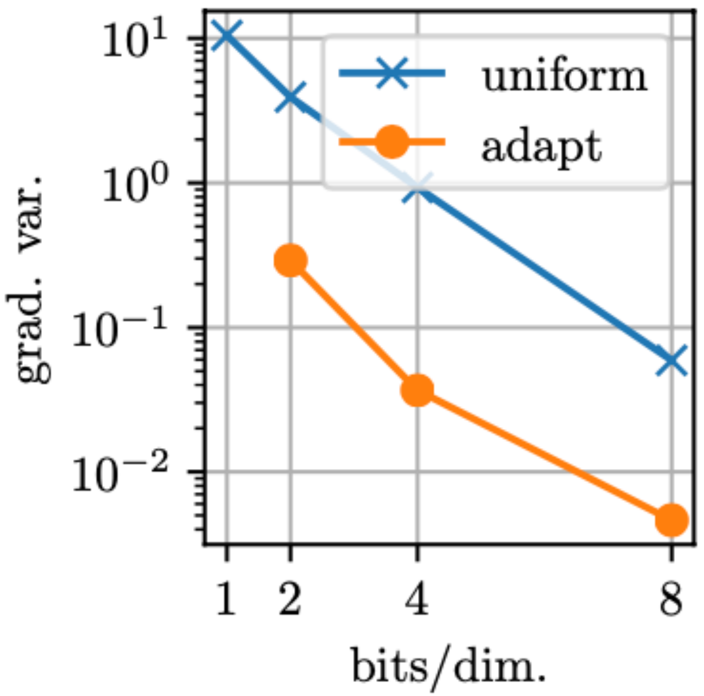
Inferred per-tensor sensitivity and bits/dim.



Assign bits based on tensor sensitivity.

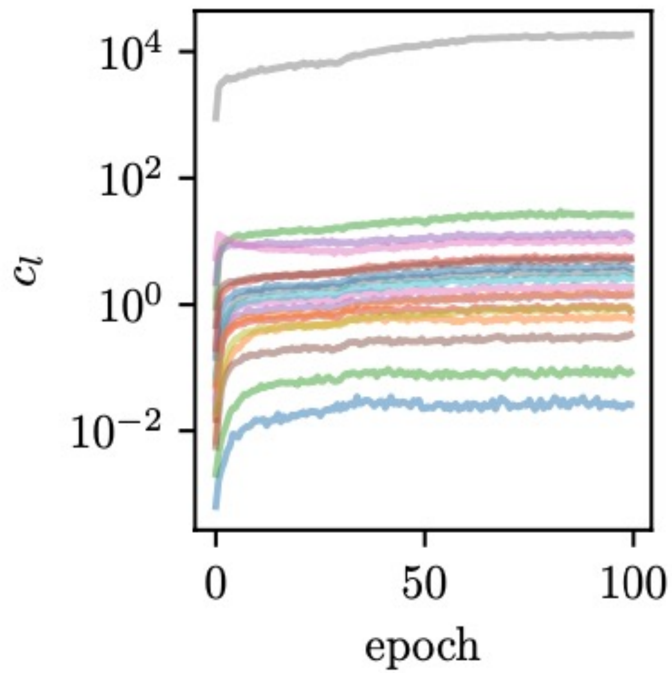
Experiments – Compression Strategy

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

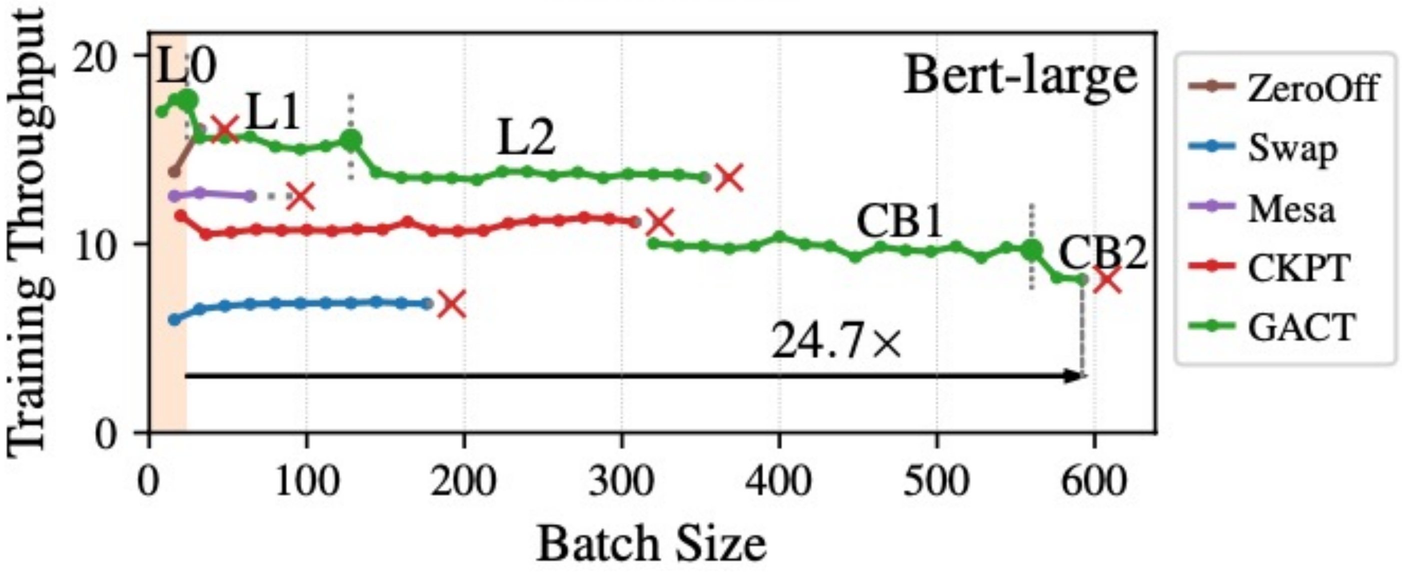
- GACT can be applied to a wide range of deep learning tasks: Computer Vision, NLP, graph NN.
- GACT has negligible accuracy loss compared with full precision training.

	Task	Model	Accuracy Full Precision / GACT 4 bit	Memory Reduction
Computer Vision:	Image Classification	ResNet-50	77.3 / 77.0	6.69 X
		Swin-tiny	81.2 / 81.0	7.44 X
	Object Detection	Faster RCNN	37.4 / 37.0	4.86 X
		RetinaNet	36.5 / 36.3	3.11 X
Graph NN:	Graph Classification (Yelp dataset)	GCN	39.9 / 40.0	6.42 X
		GAT	52.4 / 52.2	4.18 X
		GCNII	62.3 / 62.3	5.34 X
NLP:	Text Classification (MNLI dataset)	Bert-large	86.7 / 86.6	7.38 X

Reduce activation by up to 3X – 8X!

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: A activation compressed training framework for **generic** network architecture.
- 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/gact-icml