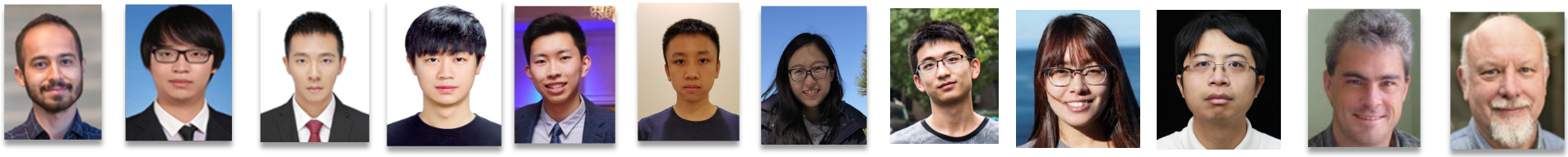
Systematic Neural Network Quantization and Pruning

Amir Gholami, Z. Yao, Z. Dong, S. Kim, Z. Zheng, E. Tan, Q. Huang, S. Yu, L. Wang, Y. Wang, Michael Mahoney, Kurt Keutzer

University of California Berkeley

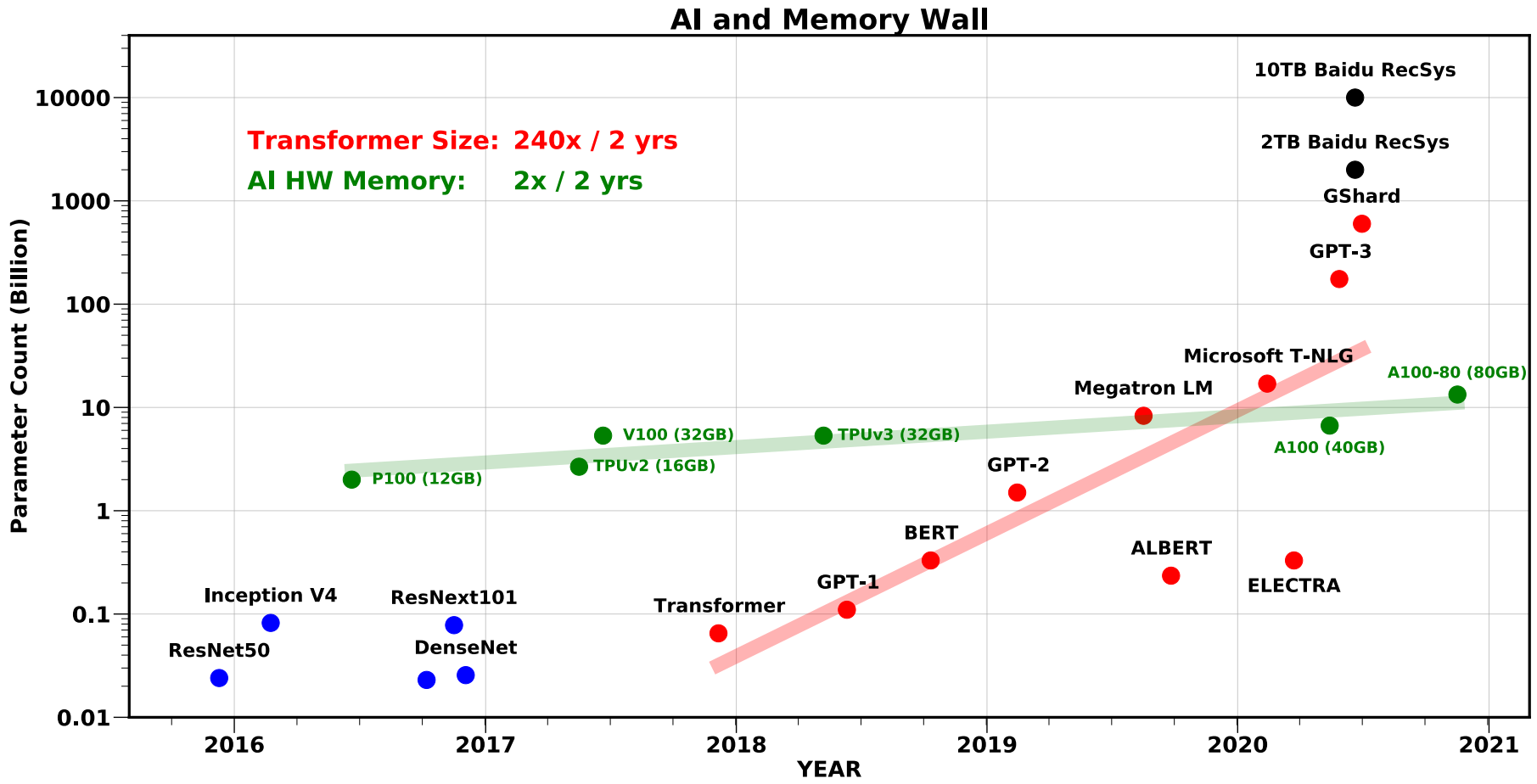
Microsoft Research Summit, Oct 2021



Outline

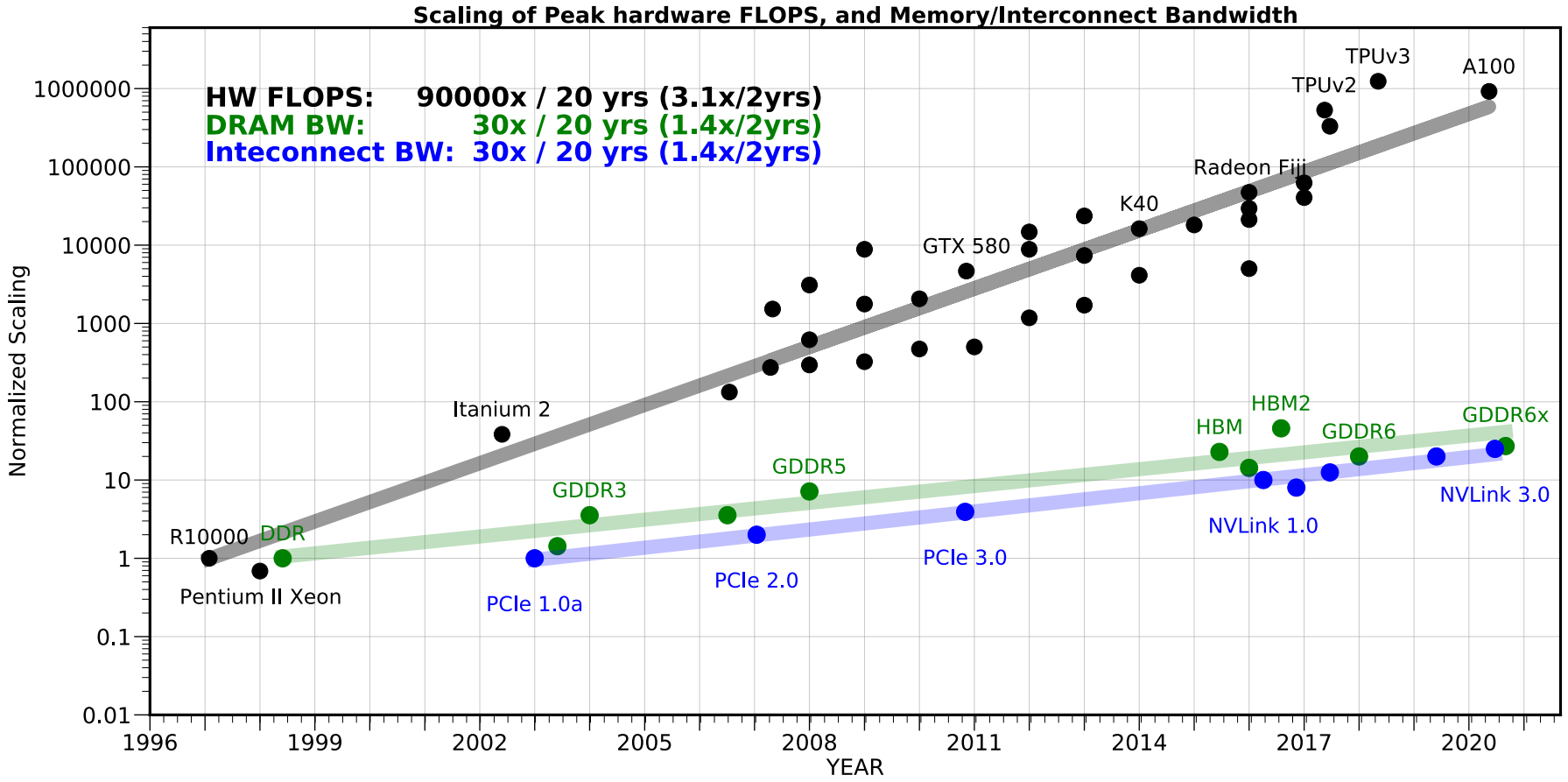
- **Introduction**
- Quantization
- Pruning
- Conclusions and Future Work

AI and Memory Wall



Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W. Mahoney, Kurt Keutzer, AI and Memory Wall, Riselab Medium Blogpost, 2021.

AI and Memory Wall



Amir Gholami, Zhewei Yao, Sehoon Kim, Michael W. Mahoney, Kurt Keutzer, AI and Memory Wall, Riselab Medium Blogpost, 2021.

Outline

- Introduction
- **Quantization**
- Pruning
- Conclusions and Future Work

Quantization Enables Fewer Memory Accesses

- Memory accesses are the principal cost in both latency and energy
- Lower precision weights in DNN mean each memory access brings more data values
- More data values few accesses overall

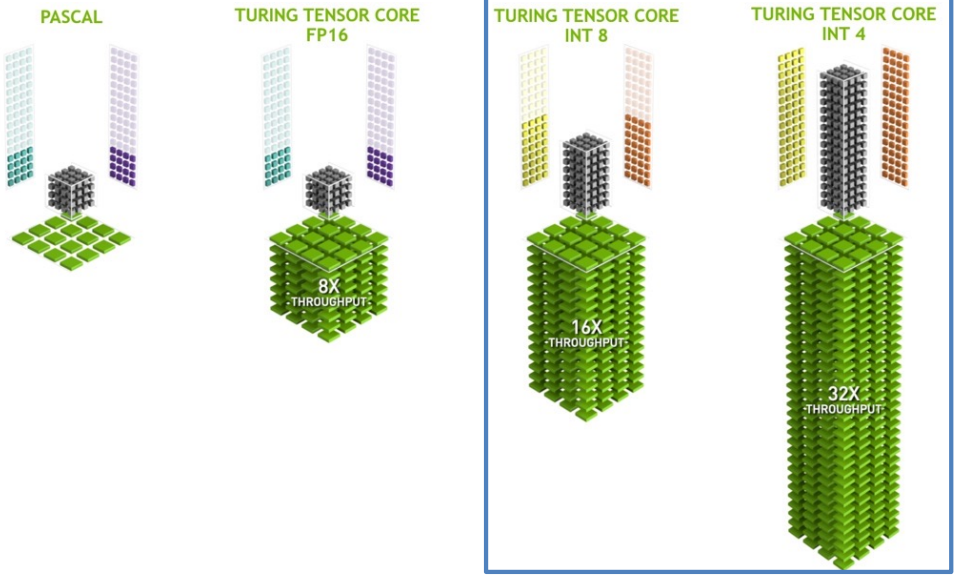
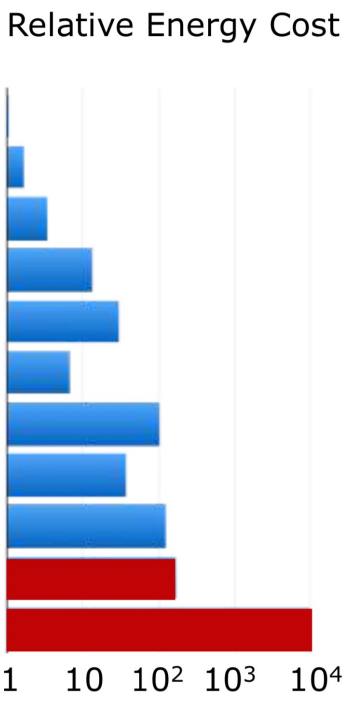


Image Credit: Sdxcentral, nvidia
Table credit: Mark Horowitz

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

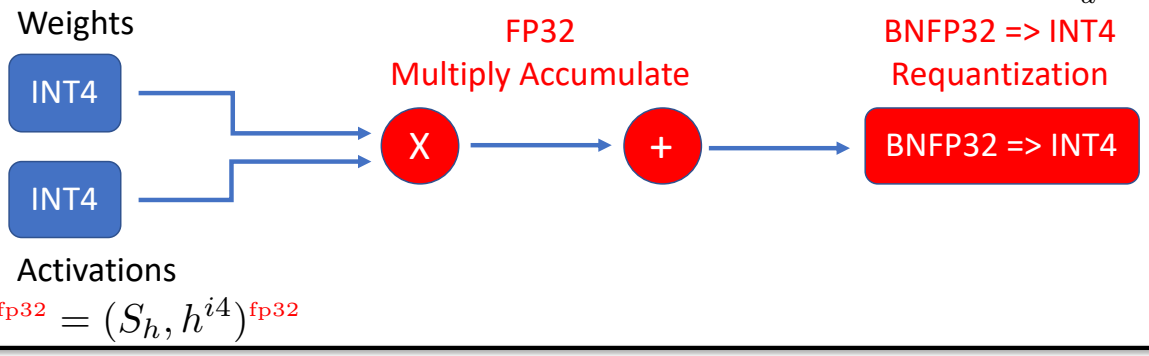
[**Horowitz**, *ISSCC* 2014]



Simulated Quantization!

Simulated Quantization: (fake quantization)

$$W^{\text{fp32}} = (S_w, W^{i4})^{\text{fp32}} \qquad a^{\text{fp32}} = W^{\text{fp32}} h^{\text{fp32}} \qquad a^{\text{i4}} = Int(\frac{(a^{\text{fp32}} - \mu^{\text{fp32}})/\sigma^{\text{fp32}}}{S_a})$$



$$h^{\text{fp32}} = (S_h, h^{i4})^{\text{fp32}}$$

- Simulated quantization performs arithmetic in software using **FP32** arithmetic, but this can lead to discrepancy when deployed on integer-only hardware
- Also many components such as **BN** or **residual connection** are computed using **FP32** which is not executable on integer-only hardware

Z. Yao*, Z. Dong*, Z. Zheng*, A. Gholami*, E. Tan, J. Li, L. Yuan, Q. Huang, Y. Wang, M. W. Mahoney, K. Keutzer, HAWQ-V3: Dyadic Neural Network Quantization in Mixed Precision, ICML'21.

Simulated/Fake Quantization Error

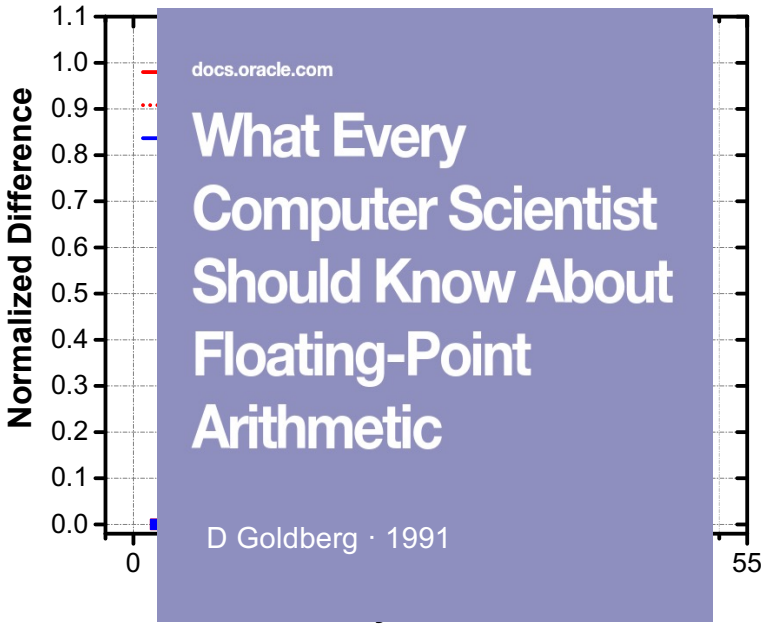
- Simulated/Fake quantization can create O(1) error for simple operations such as residual connection. This is because rounding operation is not linear:

$$Int(a+b) \neq Int(a) + Int(b)$$

Example:

$$Int(2.4+1.3) = 4 \neq Int(2.4) + Int(1.3) = 3$$

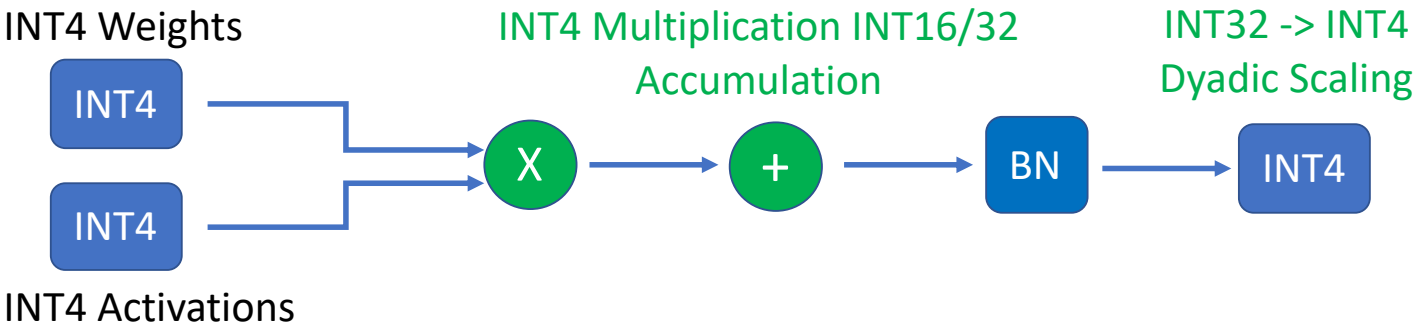
- This difference is O(1) and will propagate throughout the network, especially for low precision quantization as shown in the figure



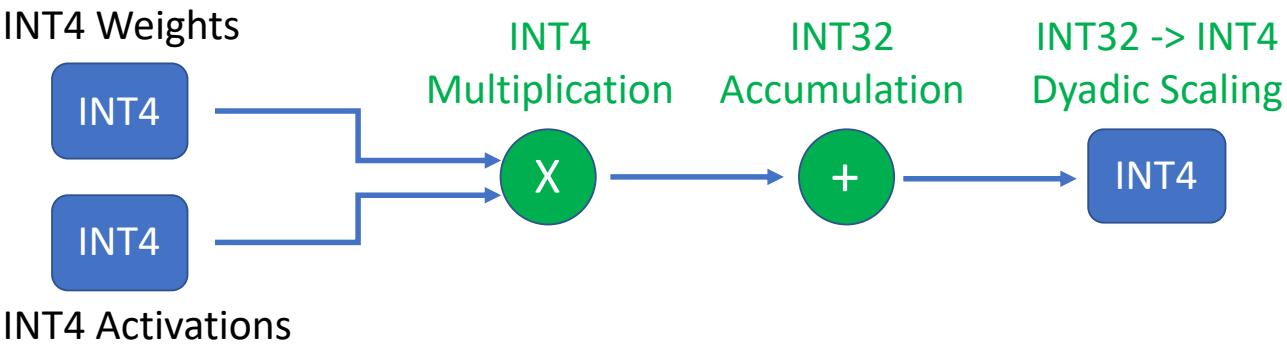
The normalized difference between fake quantization in PyTorch and corresponding values when deployed in hardware for ResNet50 on ImageNet. As one can see the error becomes quite significant, especially for low precision quantization

Z. Yao*, Z. Dong*, Z. Zheng*, A. Gholami*, E. Tan, J. Li, L. Yuan, Q. Huang, Y. Wang, M. W. Mahoney, K. Keutzer, HAWQ-V3: Dyadic Neural Network Quantization in Mixed Precision, ICML, 2021.

Integer-only Quantization!



- Key idea is to not fold BN during training and only do BN folding at the end, otherwise you will effectively be training a model without BN which is known to be hard to optimize



Integer-only Quantization Works: CV

(a) *MobileNetV2*

Method	Int	Uni	BL	Precision	Size	BOPS	Top-1
Baseline	✗	✓	73.03	W32A32	13.2	322	73.03
RVQuant (Park et al., 2018)	✗	✗	70.10	W8A8	3.3	20	70.29
CalibTIB(Hubara et al., 2020)	✗	✓	71.90	W8A8	3.3	20	71.60
HAWQV3	✓	✓	73.03	W8A8	3.3	20	**73.01**

(c) *InceptionV3*

Method	Int	Uni	BL	Precision	Size	BOPS	Top-1
Baseline	✗	✓	78.88	W32A32	90.9	5850	78.88
Integer Only (Jacob et al., 2018)	✓	✓	78.30	W8A8	22.7	366	74.20
RVQuant (Park et al., 2018)	✗	✗	74.19	W8A8	22.7	366	74.22
HAWQV3	✓	✓	78.88	W8A8	22.7	366	**78.76**

+4%

Z. Yao*, Z. Dong*, Z. Zheng*, A. Gholami*, E. Tan, J. Li, L. Yuan, Q. Huang, Y. Wang, M. W. Mahoney, K. Keutzer, HAWQ-V3: Dyadic Neural Network Quantization in Mixed Precision, ICML'21.
Online Code: https://github.com/Zhen-Dong/HAWQ

Integer-only Quantization Works: Transformers

(a) RoBERTa-Base

	Int-only	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Baseline	✗	**87.8**	**87.4**	**90.4**	**92.8**	94.6	61.2	**91.1**	90.9	78.0	86.0
I-BERT	✓	87.5	**87.4**	90.2	**92.8**	**95.2**	**62.5**	90.8	**91.1**	**79.4**	**86.3**
Diff		-0.3	0.0	-0.2	0.0	+0.6	+1.3	-0.3	+0.2	+1.4	+0.3

(b) RoBERTa-Large

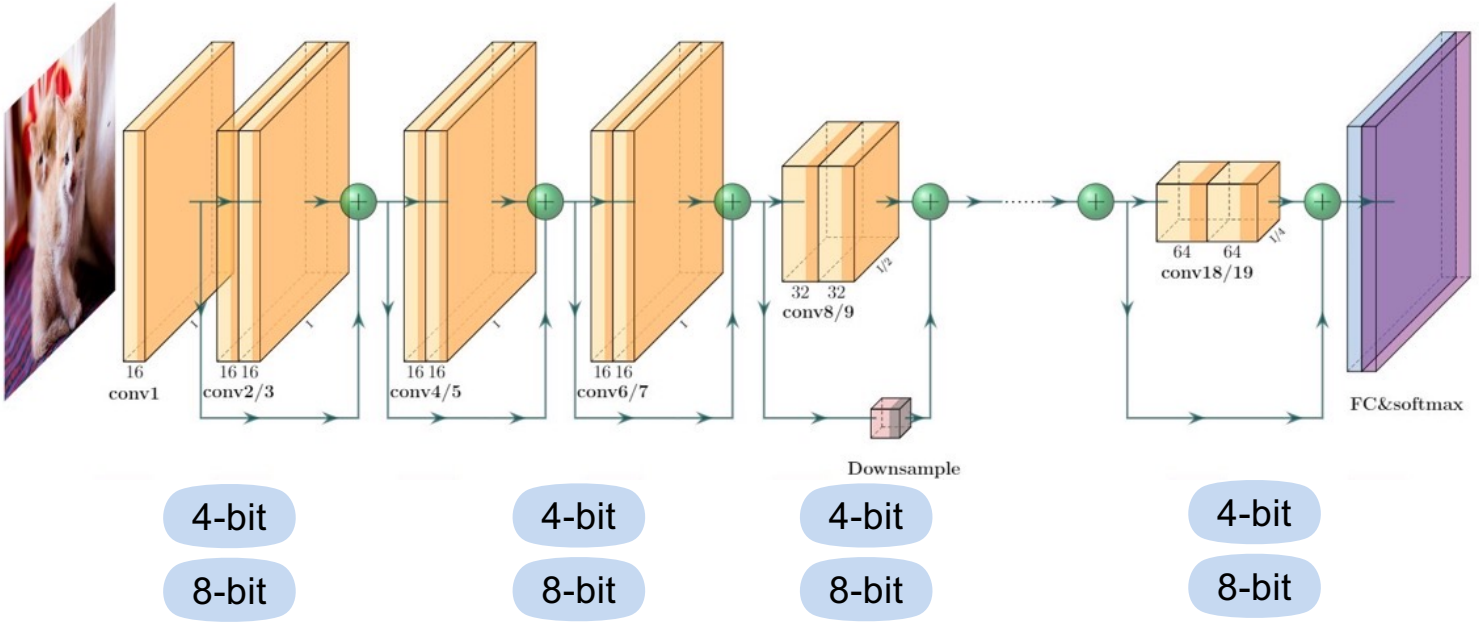
	Int-only	MNLI-m	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Baseline	✗	90.0	89.9	92.8	94.1	96.3	68.0	**92.2**	91.8	86.3	89.0
I-BERT	✓	**90.4**	**90.3**	**93.0**	**94.5**	**96.4**	**69.0**	**92.2**	**93.0**	**87.0**	**89.5**
Diff		+0.4	+0.4	+0.2	+0.4	+0.1	+1.0	0.0	+1.2	+0.7	+0.5

- I-BERT is integrated in **HuggingFace** and soon will also be available in **Google's Model Garden**

S. Kim*, A. Gholami*, Z. Yao*, M. Mahoney, K. Keutzer, **I-BERT: Integer-only BERT Quantization**, ICML, 2021. [Online Code]
S. Kim, A. Gholami, Z. Yao, A. Nrusimha, B. Zhai, T. Gao, M. Mahoney, K. Keutzer, **Q-ASR: Integer-only Zero-shot Quantization for Efficient Speech Recognition** arxiv: 2103.16827, 2021. [Online Code]

What about Sub-INT8 Quantization?

Can we go even further and perform lower precision quantization?



Uniform low precision does not work as it can significantly degrade accuracy => **Use mixed-precision**

- But how should we set the precision for each kernel?

Hessian Aware Quantization

This is somewhat similar to the **Jenga** game. We only remove blocks that are not sensitive.

- Only use low precision quantization for insensitive parameters (flat loss landscape)
- Use high precision quantization for sensitive parameters (sharp loss landscape)

This sensitivity can be calculated through Hessian which quantifies the relative sharpness/flatness of the loss landscape.

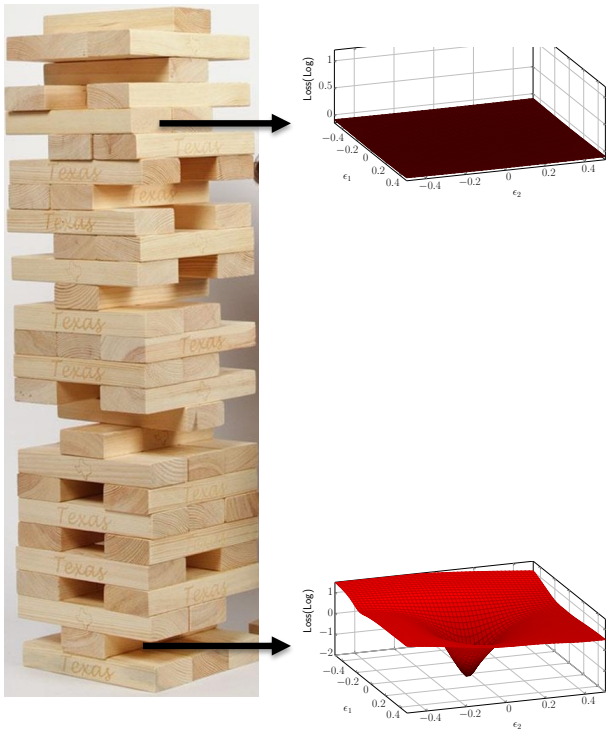
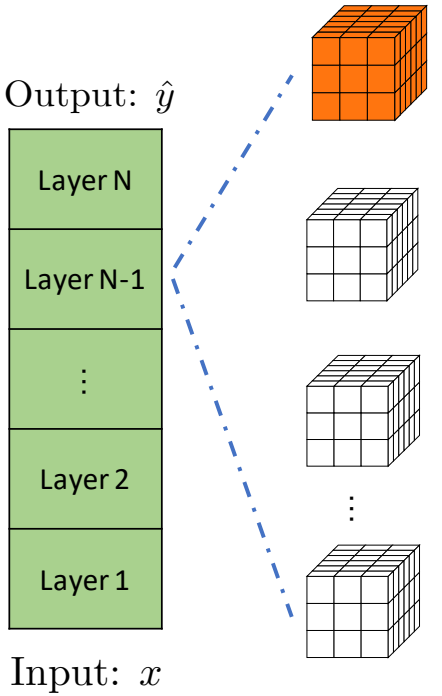
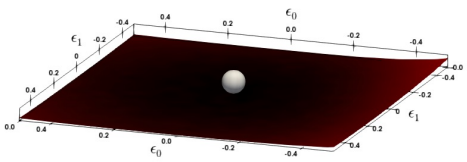


Image from UniversityCoop

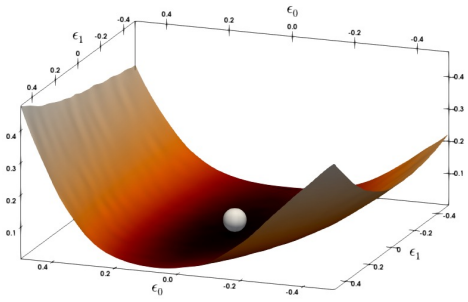


Dong Z, Yao Z, Arfeen D, Gholami A, Mahoney MW, Keutzer K. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. NeurIPS, 2020.
Yu S, Yao Z, Gholami A, Dong Z, Mahoney MW, Keutzer K. Hessian-Aware Pruning and Optimal Neural Implant. WACV, 2022(Accepted).

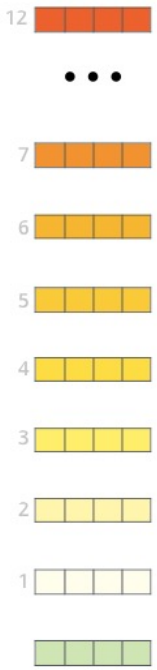
HAWQ Also works on NLP



11th Layer



7th Layer



Named Entity Recognition Task

Method	w-bits	e-bits	F_1	Size
Baseline	32	32	95.00	410.9
Q-BERT	8	8	94.79	102.8
DirectQ	4	8	89.86	62.2
Q-BERT	4	8	**94.90**	62.2
DirectQ	3	8	84.92	52.1
Q-BERT	3	8	**94.78**	52.1
Q-BERT$_{MP}$	2/4 $_{MP}$	8	**94.55**	52.1
DirectQ	2	8	54.50	42.0
Q-BERT	2	8	**91.06**	42.0
Q-BERT$_{MP}$	2/3 $_{MP}$	8	**94.37**	**45.0**

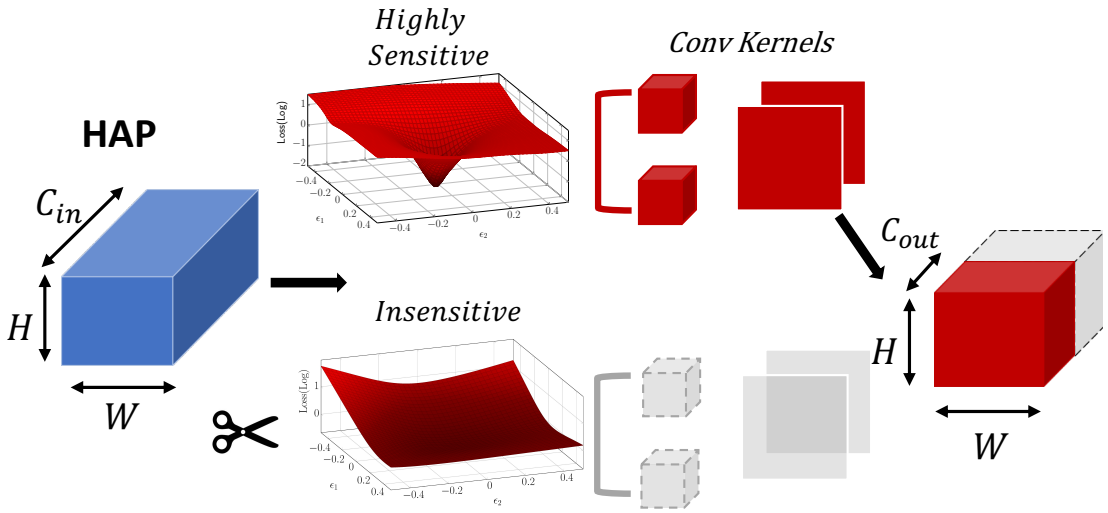
S. Sheng, Z. Dong, J. Ye, L. Ma, Z. Yao, **A. Gholami**, M. Mahoney, K. Keutzer, Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT, **AAAI, 2020.**

Outline

- Introduction
- Quantization
- **Pruning**
- Conclusions and Future Work

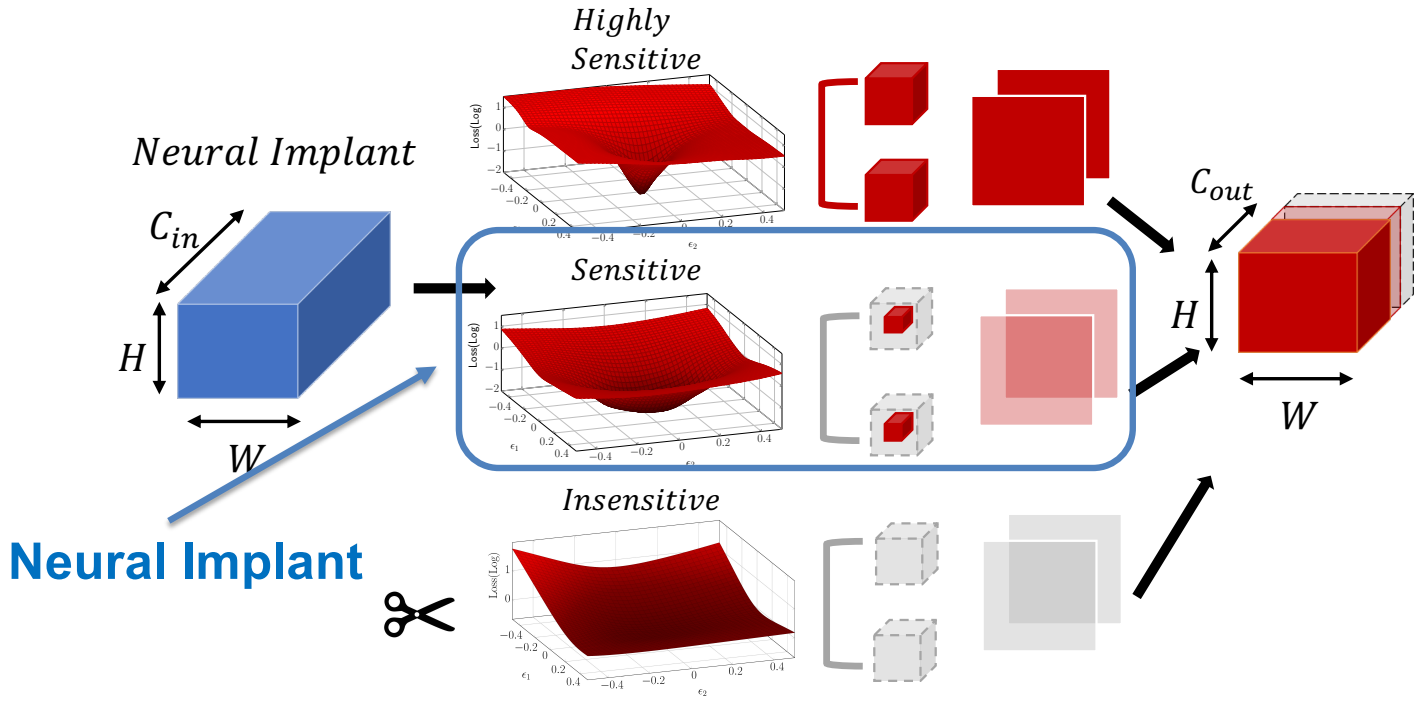
Pruning is Quantization with 0 bits

- Pruning is a special case of quantization with 0 bits, and all the previous results apply to pruning as well



LeCun Y, Denker J, Solla S. Optimal brain damage. Advances in neural information processing systems. 1989;2:598-605.
Yu S*, Yao Z*, Gholami A*, Dong Z*, Kim S, Mahoney MW, Keutzer K. Hessian-Aware Pruning and Optimal Neural Implant. WACV, 2022 (accepted).

Neural Implant



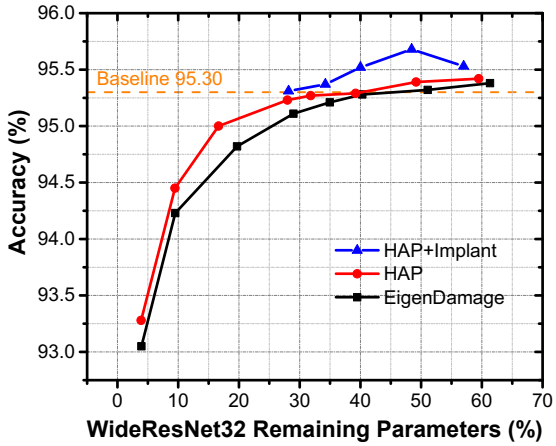
- **Completely removing** some of the neuron results in **irrecoverable accuracy loss**
- **=> We can insert a small low dimensional implant instead of complete removal**

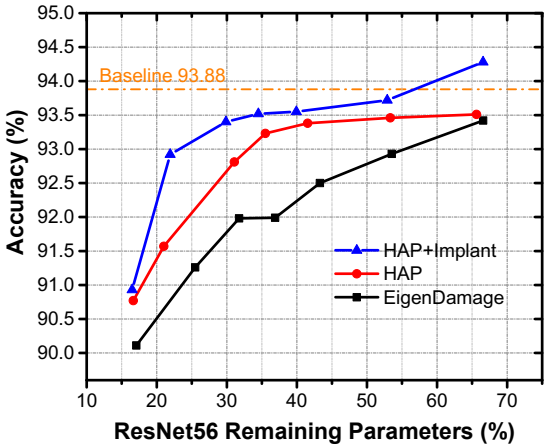
Yu S*, Yao Z*, Gholami A*, Dong Z*, Kim S, Mahoney MW, Keutzer K. Hessian-Aware Pruning and Optimal Neural Implant. WACV, 2022 (accepted).

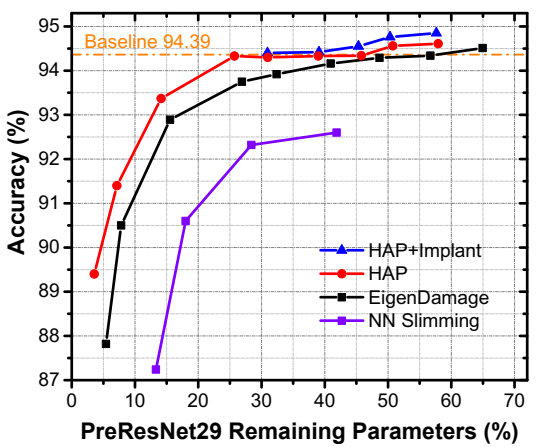
HAP with Neural Implant Achieves SOTA

- HAP with **Neural Implant** consistently achieves higher performance

Higher is better





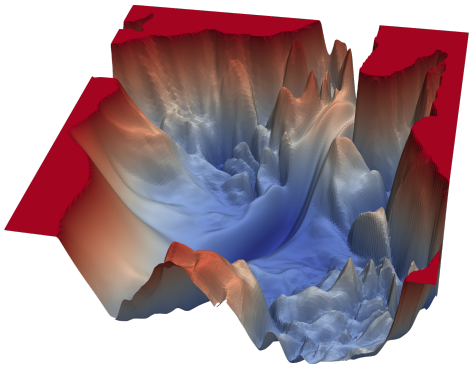


Pruning results on Cifar-10. Y-axis is accuracy, and x-axis is remaining parameters after pruning.

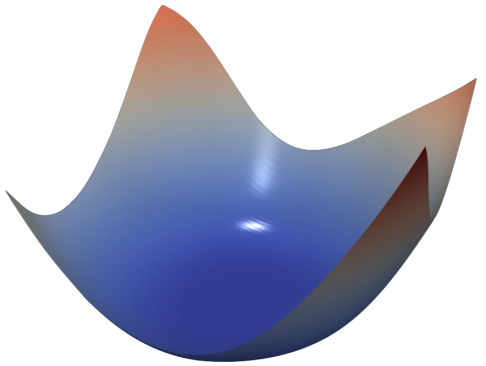
Yu S*, Yao Z*, Gholami A*, Dong Z*, Kim S, Mahoney MW, Keutzer K. Hessian-Aware Pruning and Optimal Neural Implant. WACV 2022 (accepted).
Code: **https://github.com/yaozhewei/HAP**

Summary:
One Size Does Not Fit All

- Quantization and Pruning of a model depends on the loss landscape which is a function of:
  - Model Architecture
  - Dataset
- We can quantify the sensitivity to quantization and pruning efficiently using Hessian



Less Robust to Quantization



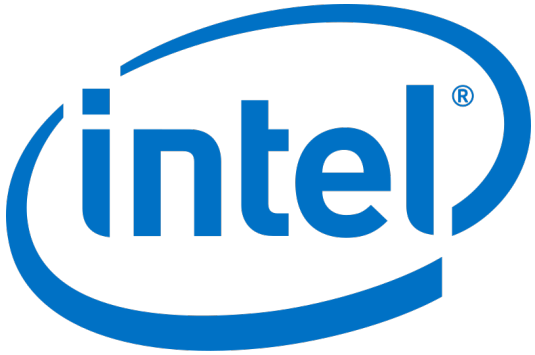
More Robust to Quantization

https://github.com/Zhen-Dong/HAWQ

Z. Yao*, Z. Dong*, Z. Zheng*, A. Gholami*, E. Tan, J. Li, L. Yuan, Q. Huang, Y. Wang, M. W. Mahoney, K. Keutzer, HAWQ-V3: Dyadic Neural Network Quantization in Mixed Precision, **ICML, 2021**.
Dong Z, Yao Z, Cai Y, Arfeen D, Gholami A, Mahoney MW, Keutzer K. Hawq-V2: Hessian aware trace-weighted quantization of neural networks. **NeurIPS, 2019.**
Dong Z*, Yao Z*, Gholami A*, Mahoney MW, Keutzer K. HAWQ: Hessian AWare Quantization of neural networks with mixed-precision. **ICCV, 2019.**
Yao Z*, Gholami A*, Shen S, Keutzer K, Mahoney MW. ADAHESSIAN: An adaptive second order optimizer for machine learning**. AAAI, 2020.**
Yao Z*, Gholami A*, Lei Q, Keutzer K, Mahoney MW. Hessian-based analysis of large batch training and robustness to adversaries. **NeurIPS, 2018.**
Yao Z*, Gholami A*, Xu P*, Keutzer K, Mahoney MW. Trust region based adversarial attack on neural networks. **CVPR, 2018.**
Images from Li H et al.

Widely Adopted by Industry

- Incorporated in Intel OpenVino Library and used for low precision quantization on FPGAs

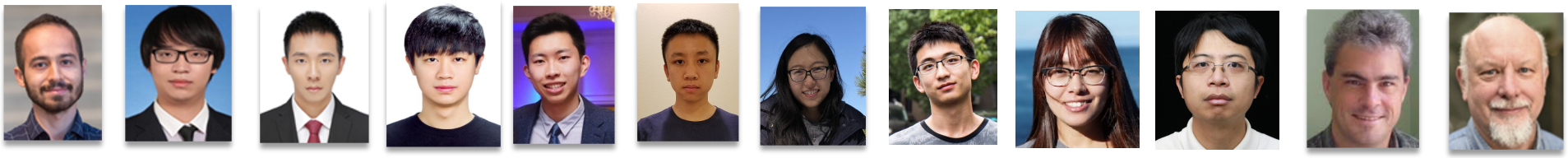


Thank You for Listening!

amirgh@berkeley.edu

Slides will be available on amirgholami.org

Hessian Code:	https://github.com/amirgholami/PyHessian
HAWQV3 Code:	https://github.com/Zhen-Dong/HAWQ
I-BERT Code:	https://github.com/kssteven418/I-BERT
Neural Implant Code:	https://github.com/yaozhewei/HAP



Berkeley UNIVERSITY OF CALIFORNIA

riselab UC Berkeley