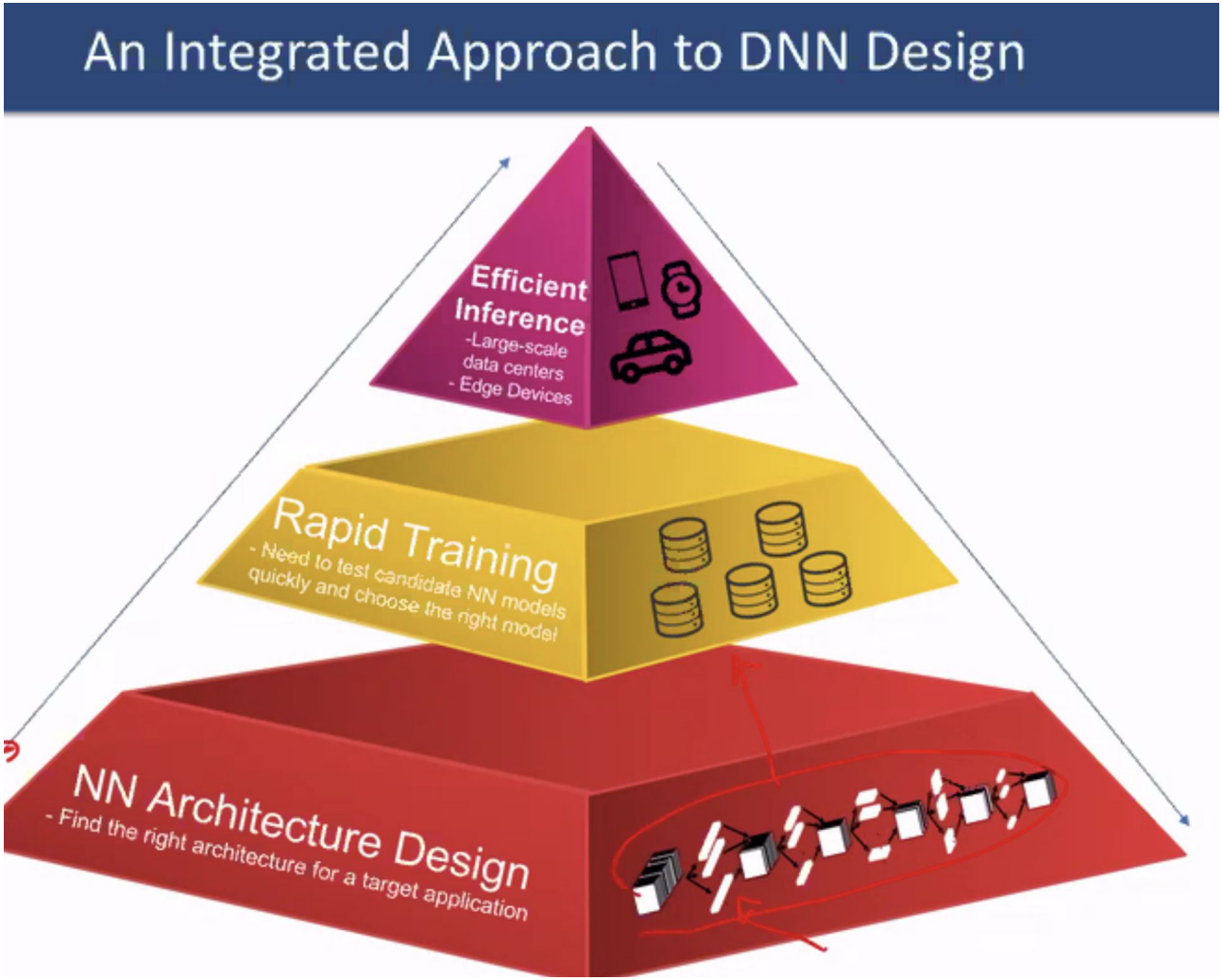
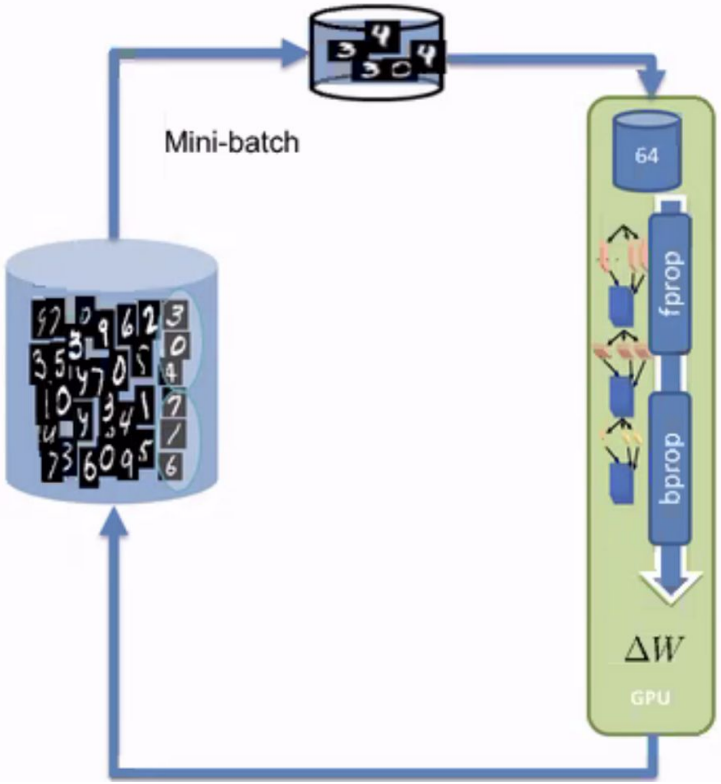


April 6 2020
Zoom Talk @Columbia by Amir Gholani (Quantization, Rapid training, hardware gain aware NN architecture search)

Amir Gholani:
amirgh@berkeley.edu

In every iteration of SGD we load a **random mini-batch of training** data, and compute the gradient.



$$\min_w E(w) = \frac{1}{N} \sum_{i=1}^{N} cost(w, x_i)$$

$$w^1 = w^0 - \frac{\lambda}{B} \sum_{i=1}^{B} \frac{\partial E_i(w^0)}{\partial w}$$

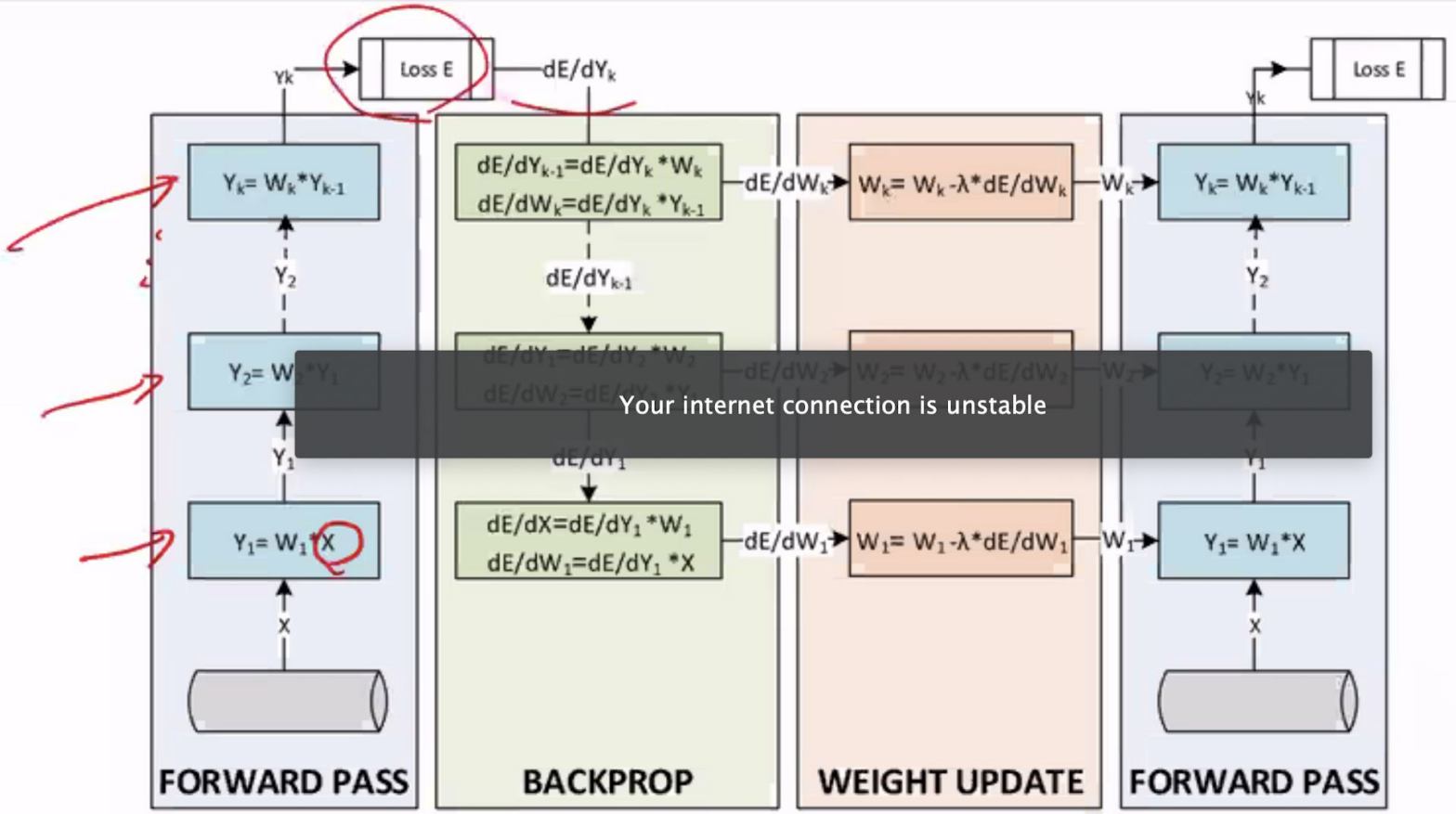


Illustration from Nvidia (B. Ginsburg)

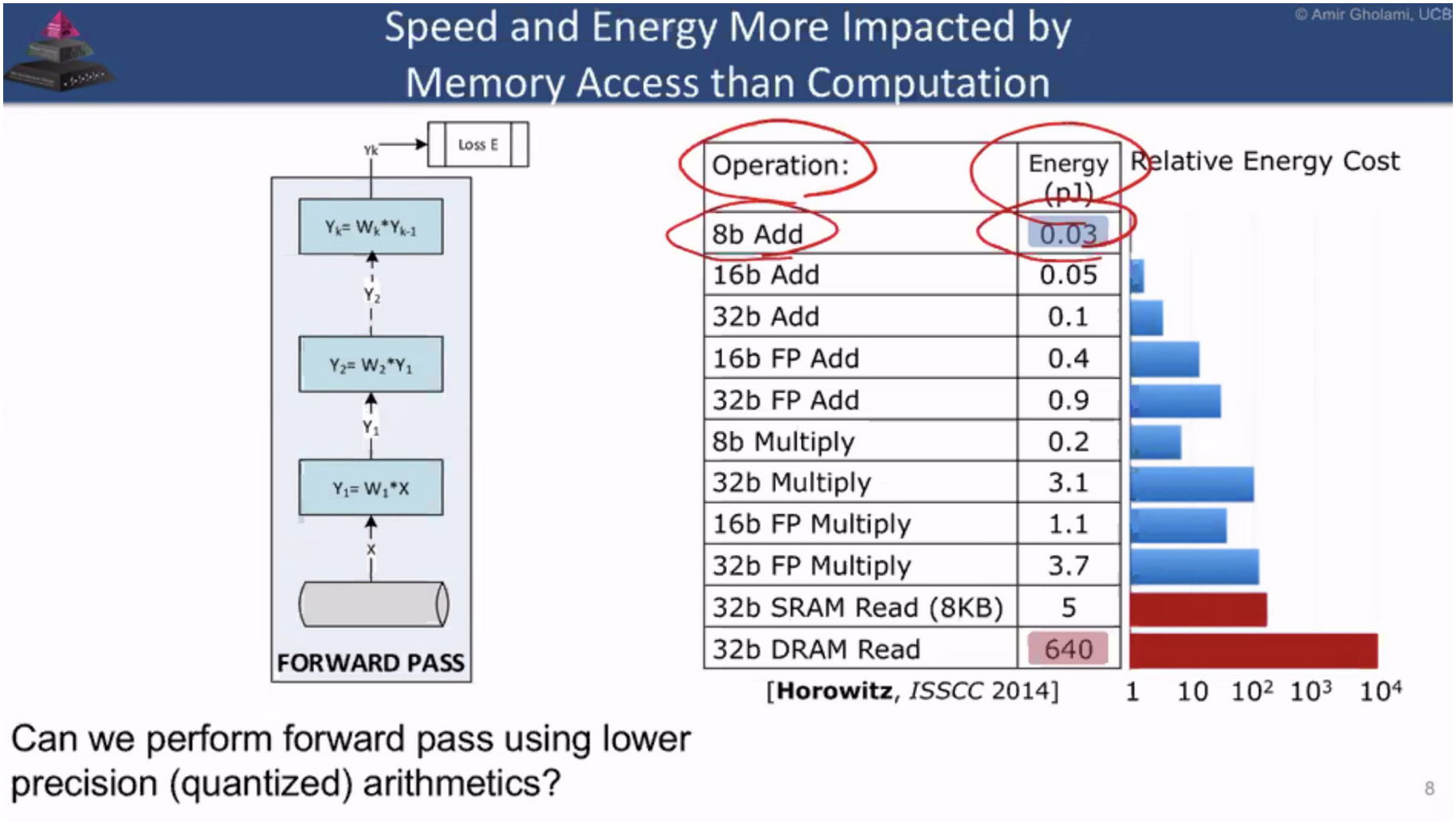
Inference

Problem Description:

Modern Neural Networks cannot be efficiently deployed on edge devices

Image Source: StartupBlog

E.g. in smart cities, FGPA at every intersection. On edge devices, power comsumption and latency is important



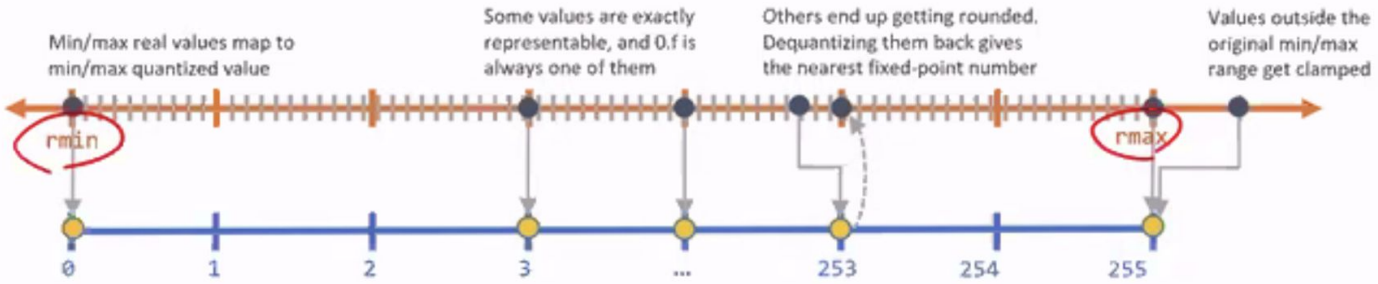
Computing operations (+,-) are much cheaper than data loading operations (e.g. DRAM)

Uniform Quantization

- r: Real value
- r_{max}, r_{min} : max/min range of values
- B: Quantization bits
- S (FP32), z (int): Scale and bias
- q: Fixed point quantized values

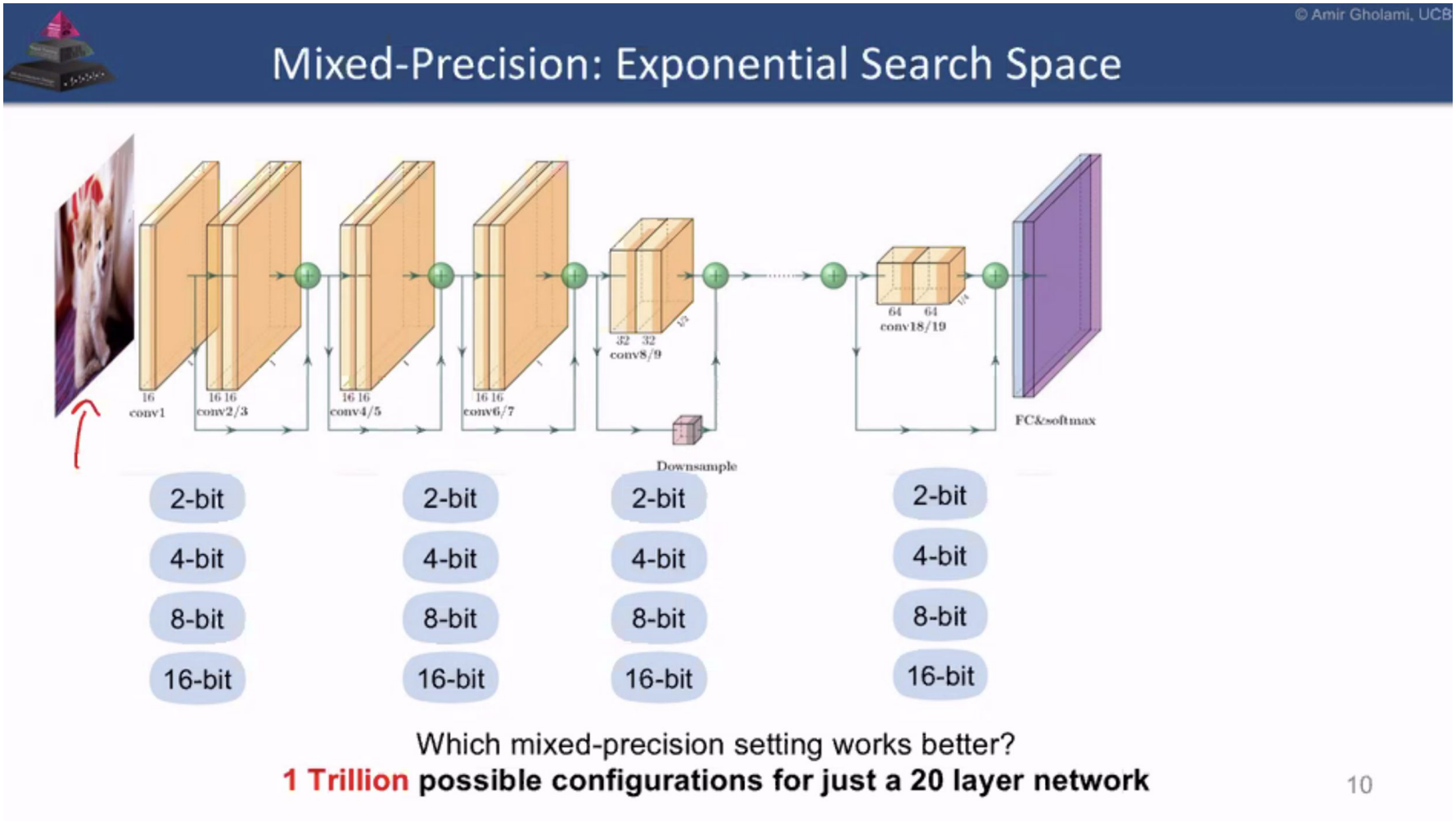
$$r = \frac{r_{max} - r_{min}}{2^B - 1}(q - z)$$

$$r = S(q - z)$$



Uniform 8-bit Quantization

Illustration from Sahni Manas



4 choices for each layer. We want to use keep more sensitive layer matrix multiplications in high precision. Exponential Search

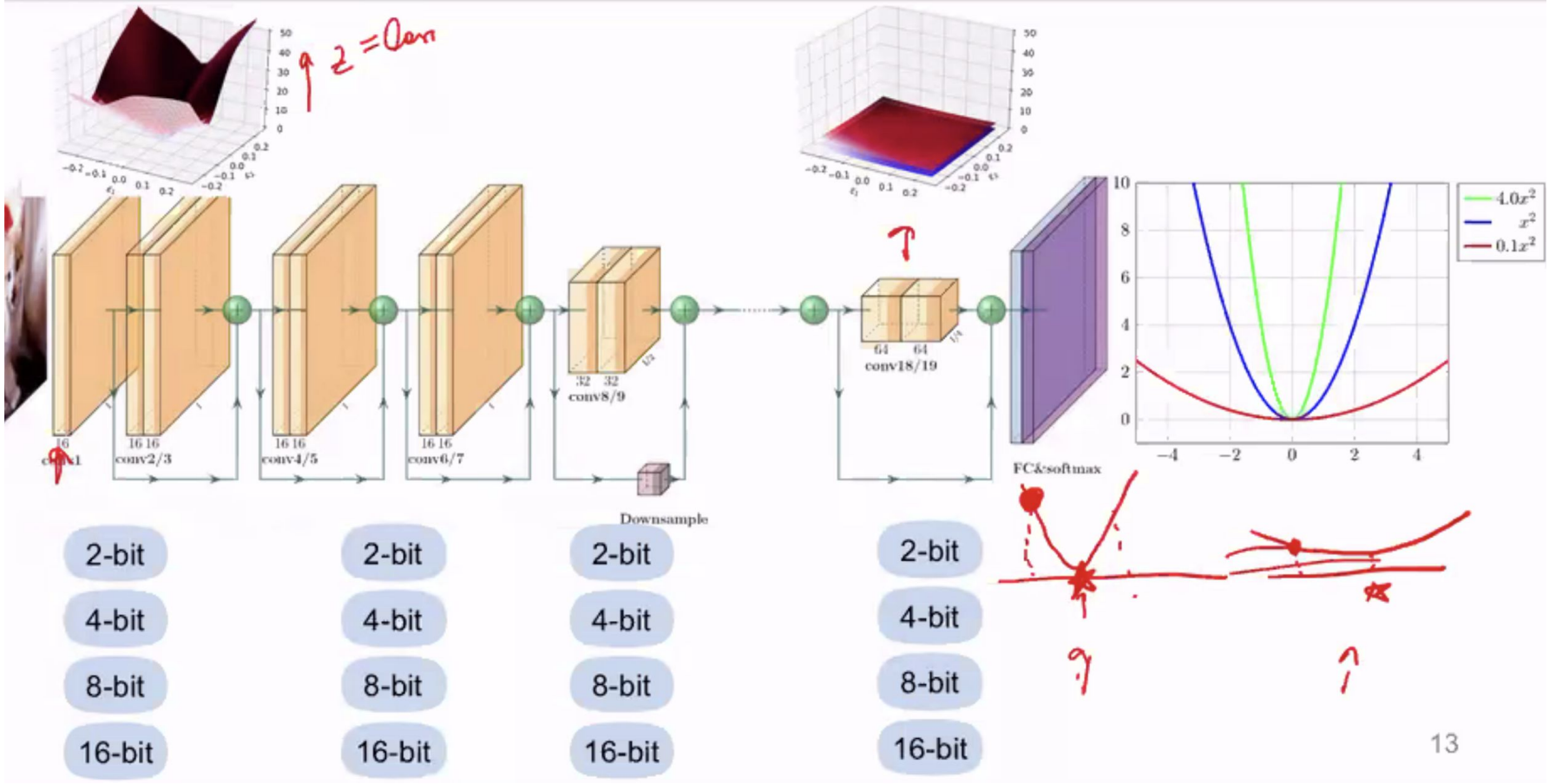
Status of Existing Methods: Only work for some problems

HAQ method works on ResNet50, but fails on other models even after > 50 hours of search time.

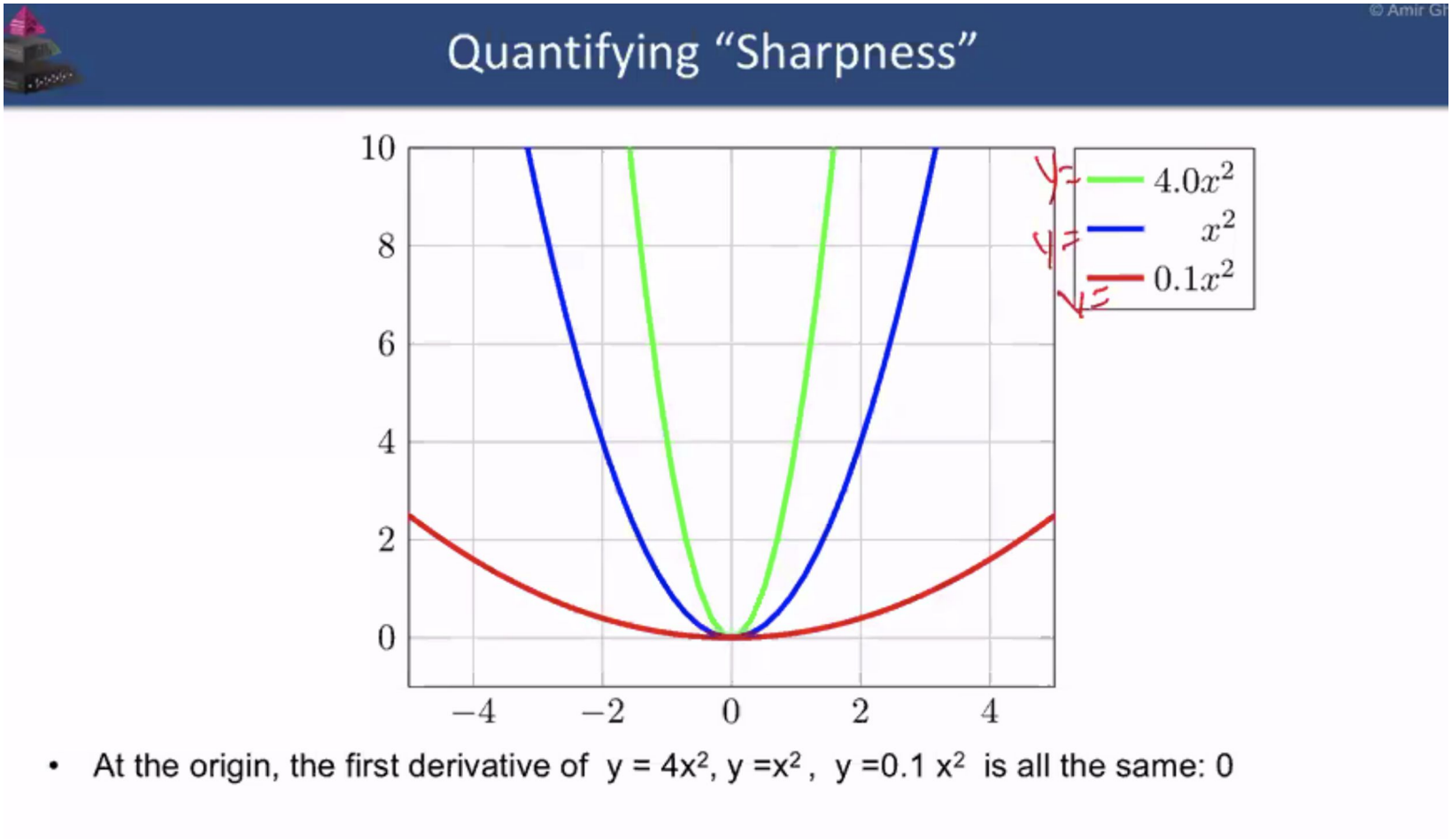
Model	Method	Top-1 Accuracy	W-Comp	Size(MB)	Search Time(hours)
ResNet50	Baseline Model	**77.39**	1.00×	97.8	
ResNet50	HAQ [22]	**75.30**	10.57×	9.22	**10**
InceptionV3	Baseline Model	**77.45**	1.00×	91.2	
InceptionV3	HAQ [22]	**71.60**	10.00×	9.12	**50**
SqueezeNext	Baseline Model	**69.38**	1.00×	10.1	
SqueezeNext	HAQ [22]	**65.87**	10.00×	1.01	**50**

[22] Wang K, Liu Z, Lin Y, Lin J, Han S. HAQ: Hardware-aware automated quantization with mixed precision. CVPR'19, 2019.

Used RL based search on ResNet, approach didn't generalize well to other architectures



Plot the loss at each layer on z axis with layer parameters on x axis, keeping other laters frozen. Flatter layers are less sensitive to aggressive quantization



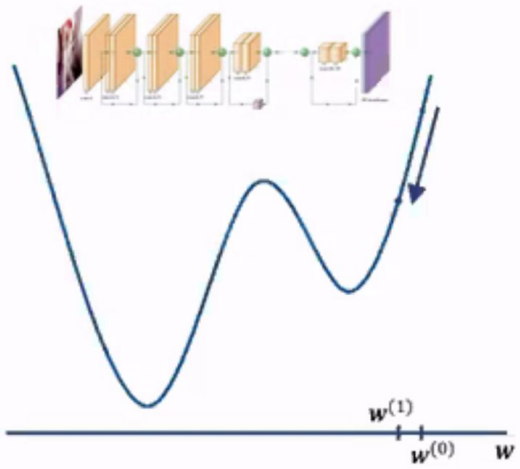
If second derivative is smaller, the loss landscape is flatter

Second Derivative (Hessian)

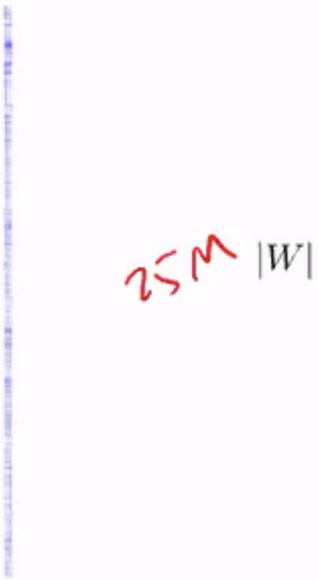
$$\min_w E(w) = \frac{1}{N}\sum_{i=1}^{N} cost(w, x_i)$$

Gradient: $\frac{\partial E}{\partial w} \in \mathcal{R}^{|W|}$

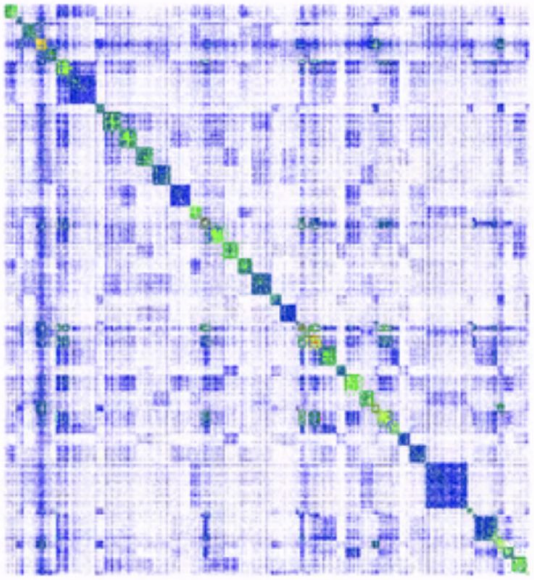
Hessian: $\frac{\partial^2 E}{\partial w^2} \in \mathcal{R}^{|W|\times|W|}$



25M = $|W|$



25M $|W|$



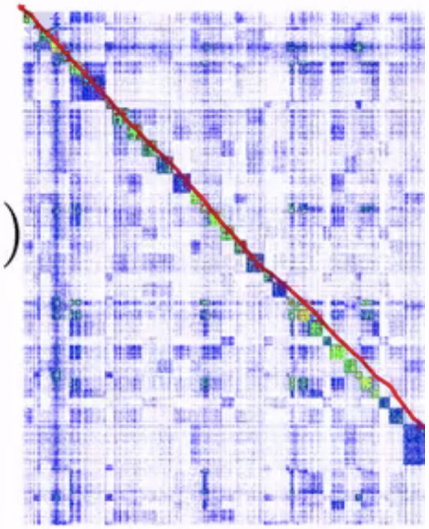
25M $|W|$

Eigenvalues of the Hessian are the second derivative gradients

Hessian Eigenvalues

Theorem 1 *After quantizing the model to same precision, fine tuning layers that have smaller average t*
of Hessian can achieve a smaller loss, compared to layers with larger ***average trace of Hessian****.*

$$\text{average } tr(H) = \frac{1}{n}\sum_i H_{ii} = \frac{1}{n}\sum_i \lambda_i(H)$$



Z. Dong, Z. Yao, D. Arfeen, Y. Cai, **A. Gholami**, M. Mahoney, K. Keutzer, Trace Weighted Hessian Aware Quantization, **Spotlight at NeurIPS'19 workshop** on Beyo

Heissian doesn't need to be computed (e.g. 24M for Inception) which is huge. Instead we use the Hutchinson's algorithm

Theorem: Average Trace (sum of diagonal elements) will give me average eigenvalue, i.e. it'll give me a measure of average second derivative

Hutchinson's Algorithm

Algorithm 4: Hutchinson Method for Trace Computation

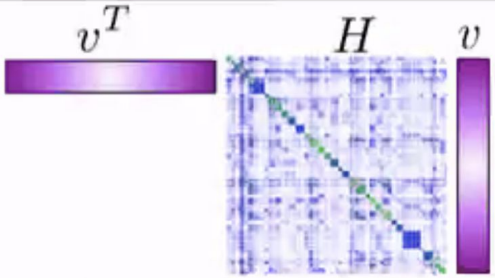
Input: Parameter: θ.

Compute the gradient of θ by backpropagation, *i.e.*, compute $g_\theta = \frac{dL}{d\theta}$.

for $i = 1, 2, \ldots$ **do** `// Hutchinson Steps`

- Draw a random vector v from Rademacher distribution (same dimension as θ).
- Compute $gv = g_\theta^T v$
- Compute Hv by backpropagation, $Hv = \frac{d(gv)}{d\theta}$
- Compute and record $v^T H v$

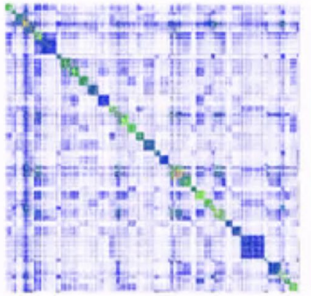
Return the average of all computed $v^T H v$.



Z. Yao*, **A. Gholami***, Q. Lei, K. Keutzer, M. Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'18, 2018.
Z. Yao*, **A. Gholami***, K. Keutzer, M. Mahoney, PyHessian: Neural Networks Through the Lens of the Hessian, arxiv:1912.07145
Code: https://github.com/amirgholami/PyHessian

How to Compute Sharpness?

- How to compute Hessian matrix vector multiply without forming Hessian?



$$g = \frac{dL}{d\theta}$$

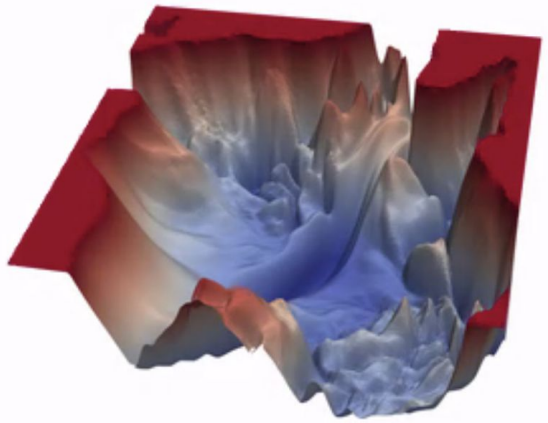
$$Hv = \frac{dg^T v}{d\theta} = \frac{dg^T}{d\theta} v + g^T \frac{dv}{d\theta} = \frac{dg^T}{d\theta} v$$

Z. Yao*, **A. Gholami***, Q. Lei, K. Keutzer, M. Mahoney, Hessian-based Analysis of Large Batch Training and Robustness to Adversaries, NeurIPS'18, 2018.
Code: https://github.com/amirgholami/PyHessian

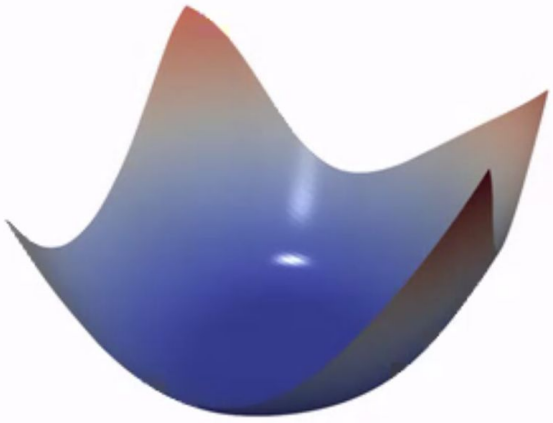
Github:
PyHessian

Important Point:
One Size Does Not Fit All

- Quantizability of a model depends on loss landscape which is a function of:
  - Model Architecture
  - Dataset
- We can compute the quantizability of the model using higher order statistics



Less Robust to Quantization



More Robust to Quantization

Li H, Xu Z, Taylor G, Studer C, Goldstein T. Visualizing the loss landscape of neural nets. NeurIPS'18, 2018

Left: Sensitive

Right: Can do more aggressive quantization

HAWQ-V2: ResNet50 on ImageNet

Precisions for all layers as well as block-wise fine-tuning orders are 100% automatically selected.

Method	w-bits	a-bits	Top-1	W-Comp	Size(MB)
Baseline	32	32	77.39	1.00×	97.8
Dorefa [43]	2	2	67.10	16.00×	6.11
Dorefa [43]	3	3	69.90	10.67×	9.17
PACT [2]	2	2	72.20	16.00×	6.11
PACT [2]	3	3	75.30	10.67×	9.17
LQ-Nets [40]	3	3	74.20	10.67×	9.17
Deep Comp. [8]	3	MP	75.10	10.41×	9.36
HAQ [35]	MP	MP	75.30	10.57×	9.22
HAWQ	2 MP	4 MP	75.48	12.28×	**7.96**
HAWQ-V2	2 MP	4 MP	**75.56**	12.25×	**7.98**

Z. Dong*, Z. Yao*, **A. Gholami***, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19, 2019.
Z. Dong, Z. Yao, D. Arfeen, Y. Cai, **A. Gholami**, M. Mahoney, K. Keutzer, Trace Weighted Hessian Aware Quantization, **Spotlight at NeurIPS'19 workshop** on Bey
First-Order Optimization Methods in Machine Learning, 2019.

HAWQ-V2: End to end automatic, precision choices based on the Hessian

Performance of HAWQ-V@ on other architectutes

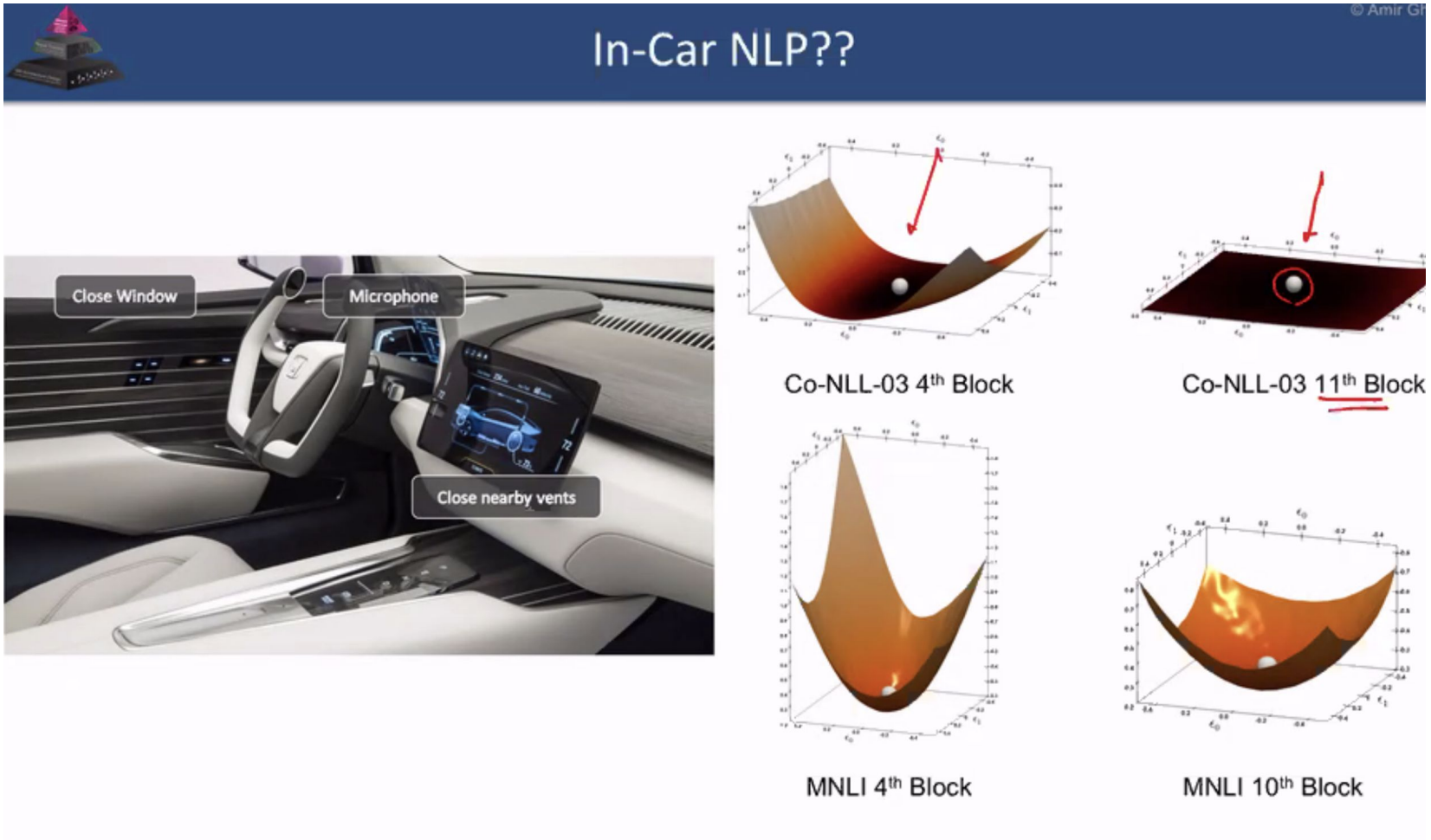
Example Comparison with RL Based AutoML

Comparison between HAWQ-V2 and HAQ on ResNet50, InceptionV3 and SqueezeNext models

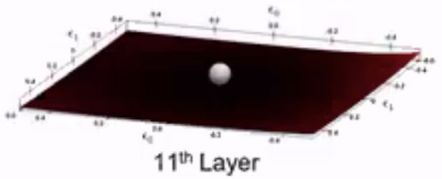
Model	Method	Top-1 Accuracy	W-Comp	Size(MB)	Search Time(hours)	Speed Up
ResNet50	Baseline Model	**77.39**	1.00×	97.8		
ResNet50	HAQ [22]	**75.30**	10.57×	9.22	10	
ResNet50	HAWQ-V2	**75.56**	**12.24**×	7.99	**0.5**	**> 20**×
InceptionV3	Baseline Model	**77.45**	1.00×	91.2		
InceptionV3	HAQ [22]	**71.60**	10.00×	9.12	50	
InceptionV3	HAWQ-V2	**75.68**	**12.04**×	7.57	**0.4**	**> 125**×
SqueezeNext	Baseline Model	**69.38**	1.00×	10.1		
SqueezeNext	HAQ [22]	**65.87**	10.00×	1.01	50	
SqueezeNext	HAWQ-V2	**68.38**	9.40×	1.07	**0.9**	**> 55**×

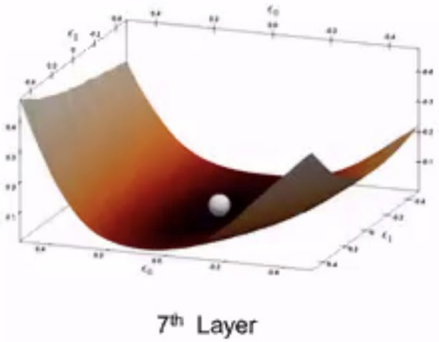
[1] Wang K, Liu Z, Lin Y, Lin J, Han S. HAQ: Hardware-Aware Automated Quantization with Mixed Precision. CVPR'19, 2019.
[2] Z. Dong*, Z. Yao*, **A. Gholami***, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19, 2019.
[3] Z. Dong, Z. Yao, Y. Cai, D. Arfeen, **A. Gholami**, M. Mahoney, K. Keutzer HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neur Networks. arXiv:1911.03852, 2019.

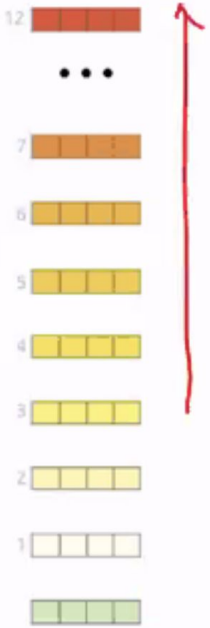
In-Car NLP (will require transformers, BERT, etc)



Named Entity Recognition (NER)







(c) CoNLL-03

Method	w-bits	e-bits	F_1	Size	Size-w/o-e
Baseline	32	32	95.00	410.9	324.5
Q-BERT	8	8	94.79	102.8	81.2
DirectQ	4	8	89.86	62.2	40.6
Q-BERT	4	8	**94.90**	62.2	40.6
DirectQ	3	8	84.92	52.1	30.5
Q-BERT	3	8	**94.78**	52.1	30.5
Q-BERT$_{MP}$	2/4 $_{MP}$	8	**94.55**	52.1	30.5
DirectQ	2	8	54.50	42.0	20.4
Q-BERT	2	8	**91.06**	42.0	20.4
Q-BERT$_{MP}$	2/3 $_{MP}$	8	**94.37**	**45.0**	**23.4**

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'20, 2020

14x smaller model vs baseline

ZeroQ: Zero Shot Quantization

RetinaNet-ResNet50 Quantization

Method	W-bit	A-bit	Size (MB)	mAP
Baseline	32	32	145.10	36.4
ZEROQ	8	8	36.25	**36.4**
FQN	4	4	18.13	32.5
ZEROQ	MP	8	18.13	**33.7**

[FQN]: Li R, Wang Y, Liang F, Qin H, Yan J, Fan R. Fully Quantized Network for Object Detection. CVPR'19, 2019.
[ZeroQ]: Y. Cai, Z. Yao, Z. Dong, **A. Gholami**, M. Mahoney, and K. Keutzer. ZeroQ: A Novel Zero Shot Quantization Framework, CVPR'20, arxiv:2001.00281, 2020.
http://wwww.github.com/amirgholami/ZeroQ
Video is from Keras-RetinaNet (source: https://www.youtube.com/watch?v=51ujDJ-01oc)

Zero-shot quantization, takes just 2 minutes on 4 V-100 GPUs

AWS: Delivery robots
Facebook: Recommendation AI
Ford: On-car quantized processing

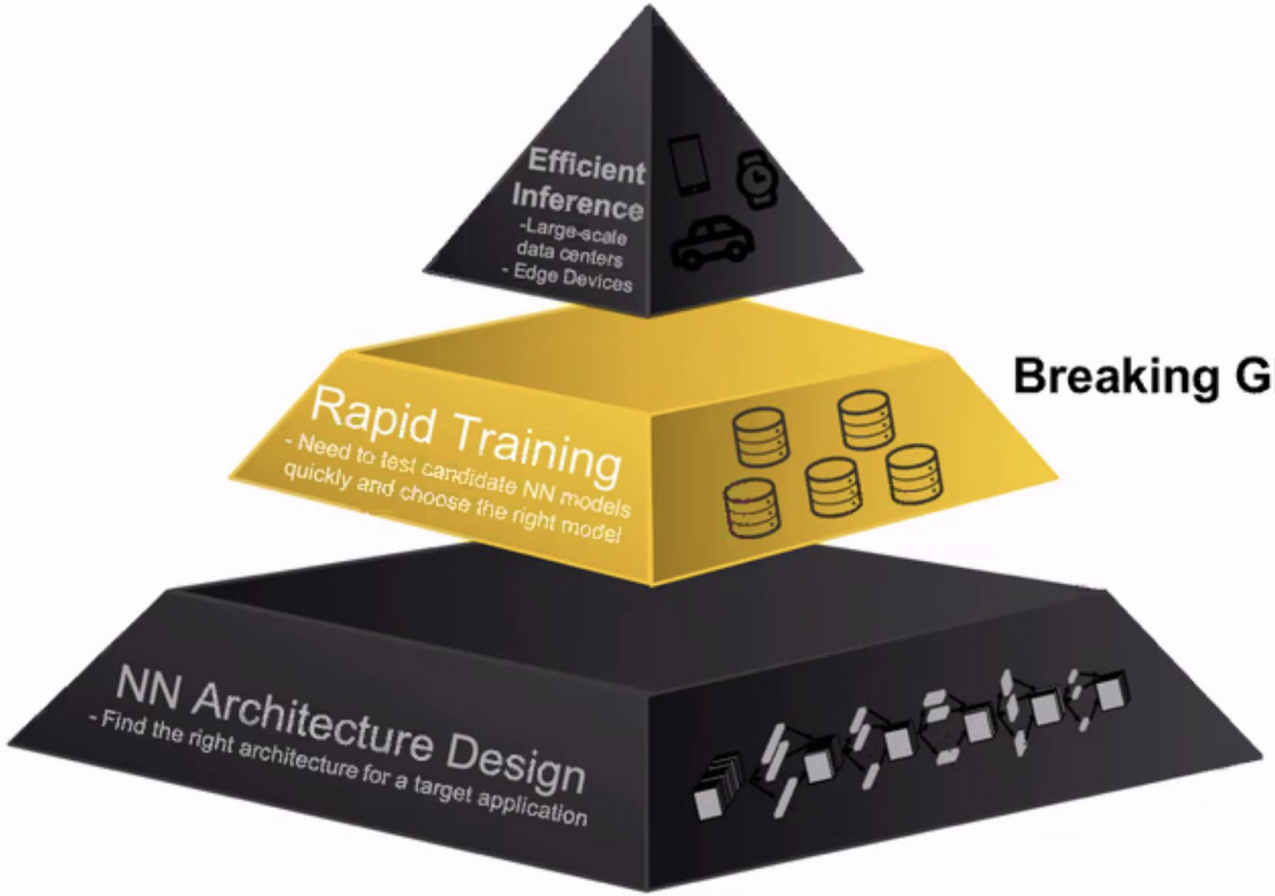
Summary

- A systematic, **second-order** quantization framework **without ad-hoc tricks**
- Novel compression results exceeding **all existing state-of-the-art** for:
  - Image Classification
  - Object Detection
  - Natural Language Processing
- Extension to Zero Shot Quantization

Z. Dong, Z. Yao, **A. Gholami**, M. Mahoney, and K. Keutzer, 2019. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. ***ICCV'19,*** 2019.
Z. Dong, Z. Yao, D. Arfeen, Y. Cai, **A. Gholami**, M. Mahoney, and K. Keutzer. Trace weighted hessian- aware quantization. **Spotlight at NeurIPS'19** workshop on Beyond First-Order Optimization Methods in Machine Learning, 2019.
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-20**, 2020.
Y. Cai, Z. Yao, Z. Dong, **A. Gholami**, M. Mahoney, and K. Keutzer. ZeroQ: A Novel Zero Shot Quantization Framework, **CVPR'20** (arxiv:2001.00281), 2020.

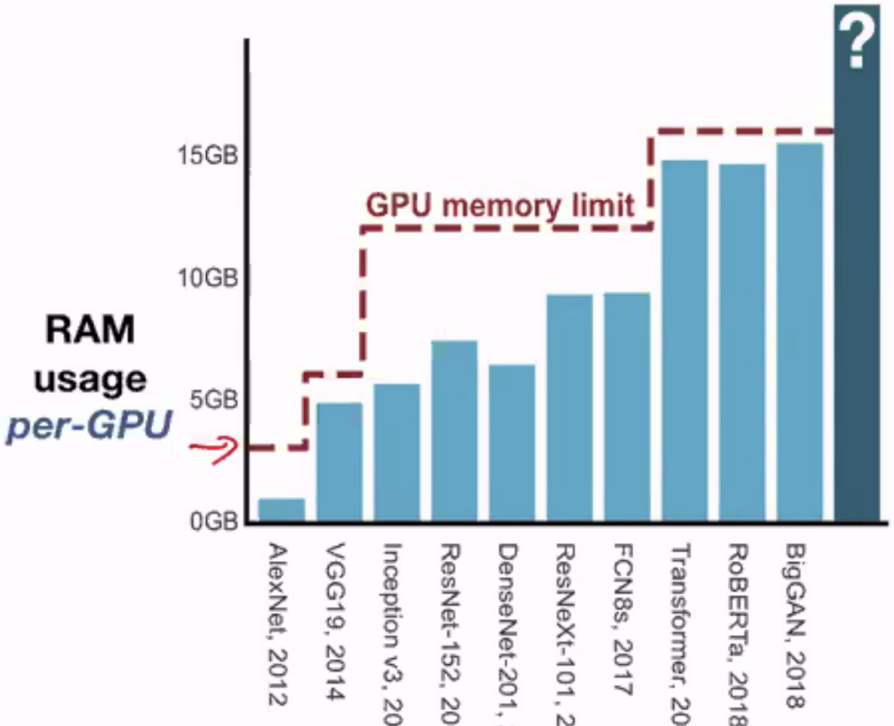
http://wwww.github.com/amirgholami/ZeroQ
http://wwww.github.com/amirgholami/PyHessian

Rapid Training of NNs



Breaking GPU Memory Wall, MLSys'20

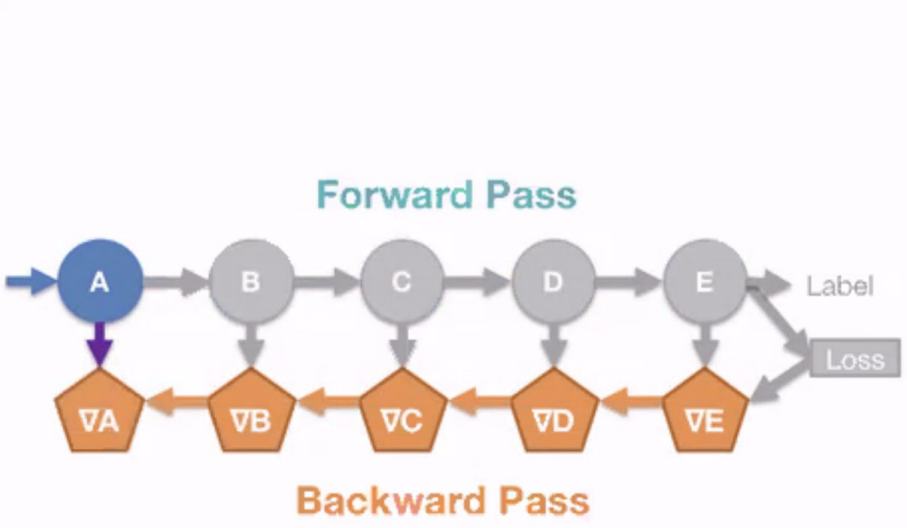
GPU Memory Wall Limit!

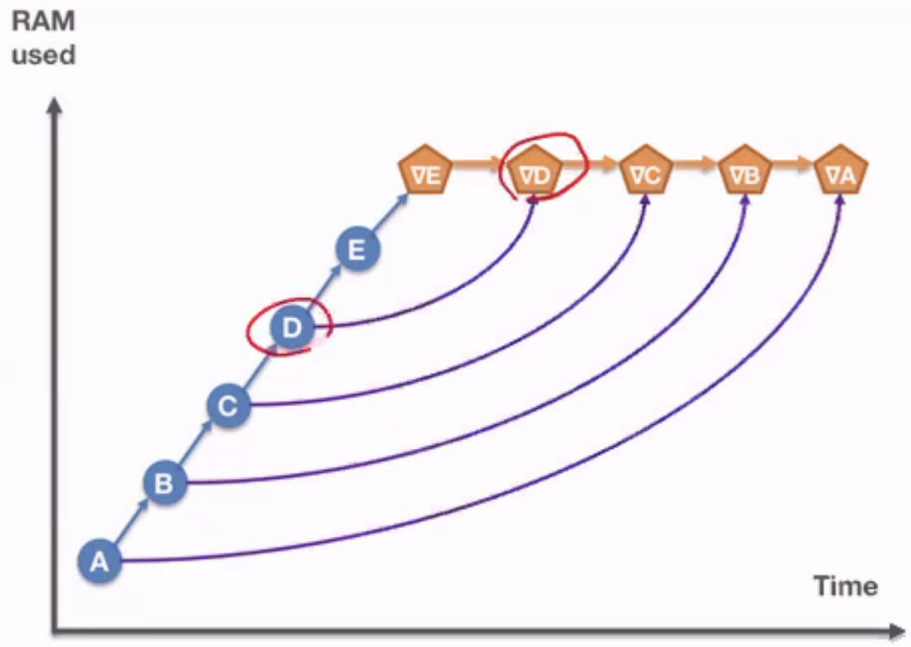


At a per-GPU level, recent state-of-the-art models have hit a **memory capacity wall**.

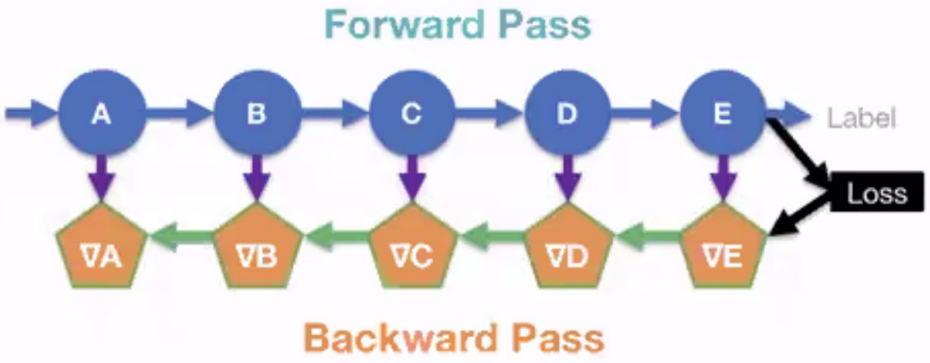
Limited GPU memory is slowing progress in new deep learning models!

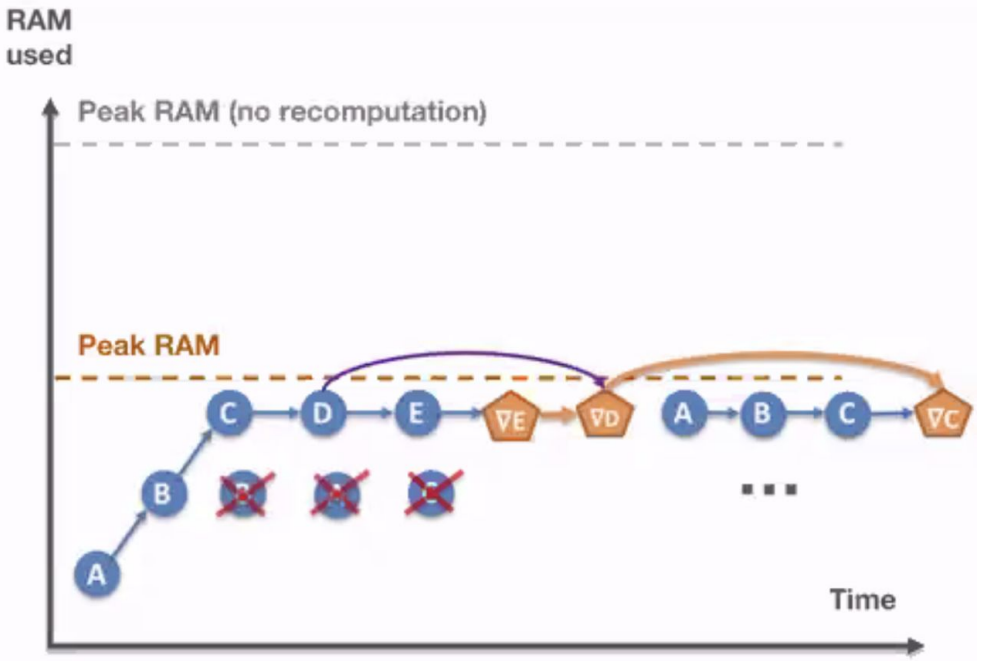
RAM Hungry BackProp: Keep All Layers in RAM





Compute Hungry BackProp: Recompute All Layers





How can we use less memory?

Free early & recompute

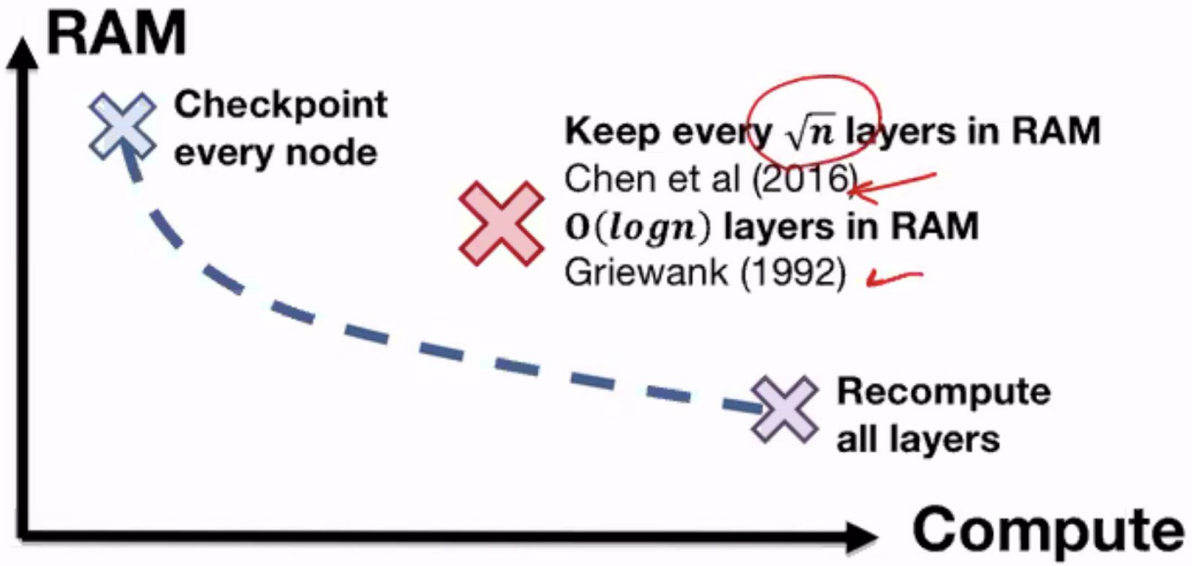
How To Trade Off RAM with Optimal Additional Compute?

Challenges:

1. Variable runtime per layer

2. Variable RAM usage per layer

3. Real DNNs are non-linear

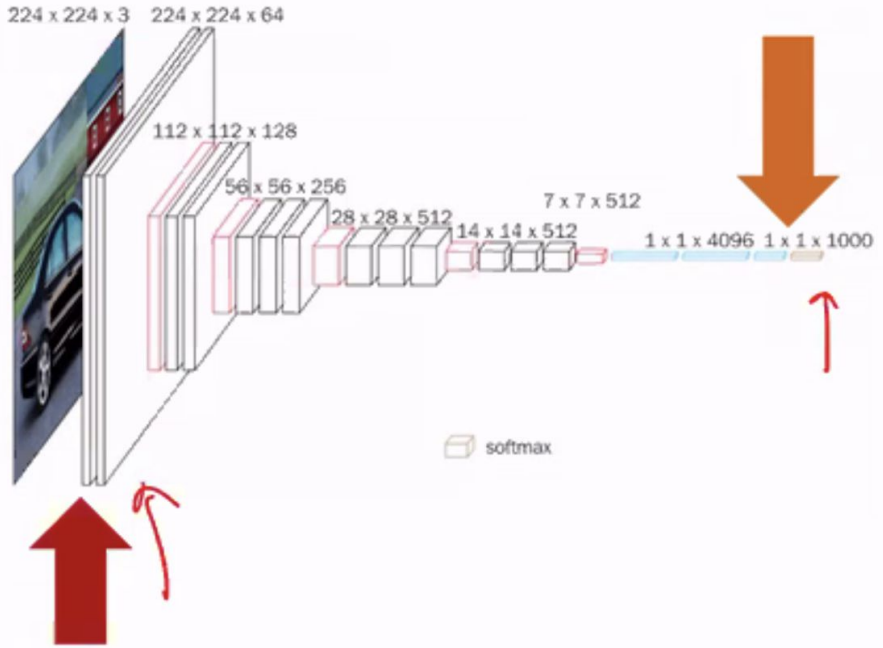


T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training Deep Nets with Sublinear Memory Cost. Apr. 2016. arXiv: 1604.06174.
Griewank A. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optimization Methods and software.,1992.

1. Variable Runtime per layer

1. Latency is not constant between layers

10^6x compute gap between biggest and smallest layer in VGG19



x10^6

CheckMate: A system for optimal tensor rematerialization

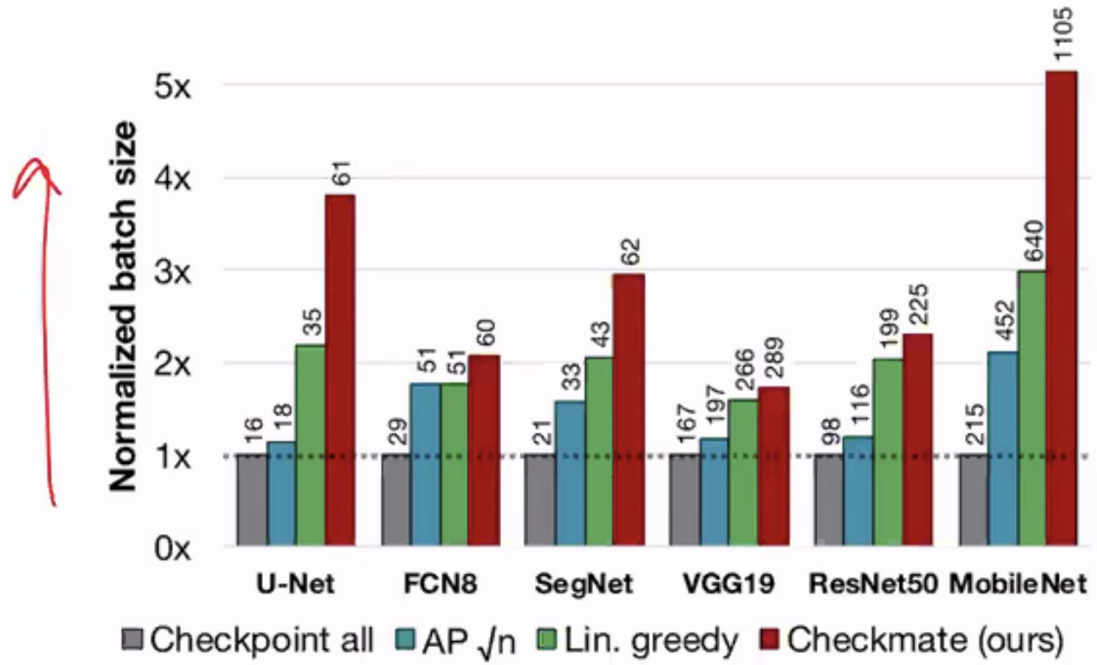
Profile layers

Solve integer LP

Rewrite TF2.0 graph

Checkmate explores optimal trade-off
5x larger models w/ 2x cost

CheckMate!

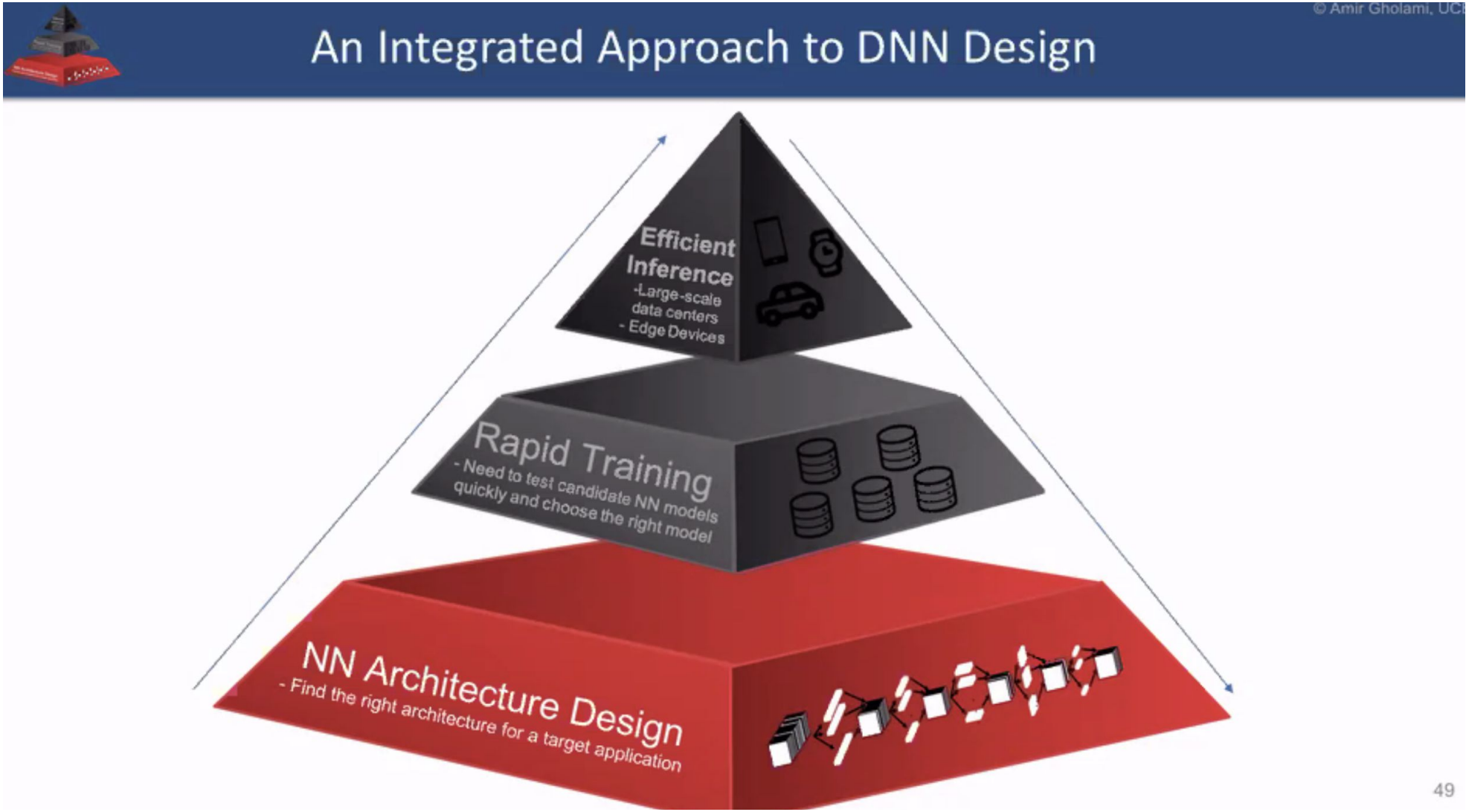


Maximum batch size possible on a single NVIDIA Tesla V100 GPU

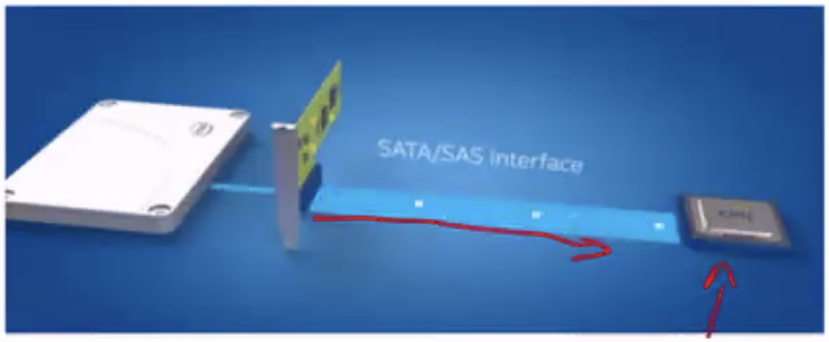
P. Jain, A. Jain, A. Nrusimha , **A. Gholami**, P. Abbeel, K. Keutzer, I. Stoica, J. Gonzalez, CheckMate: Breaking the Memory Wall with Optimal Tensor Rematerialization, MLSys'20, 2020.

Checkmate thus enables significantly larger batch sizes

NN Architecture design



Memory/Compute Bound Operations



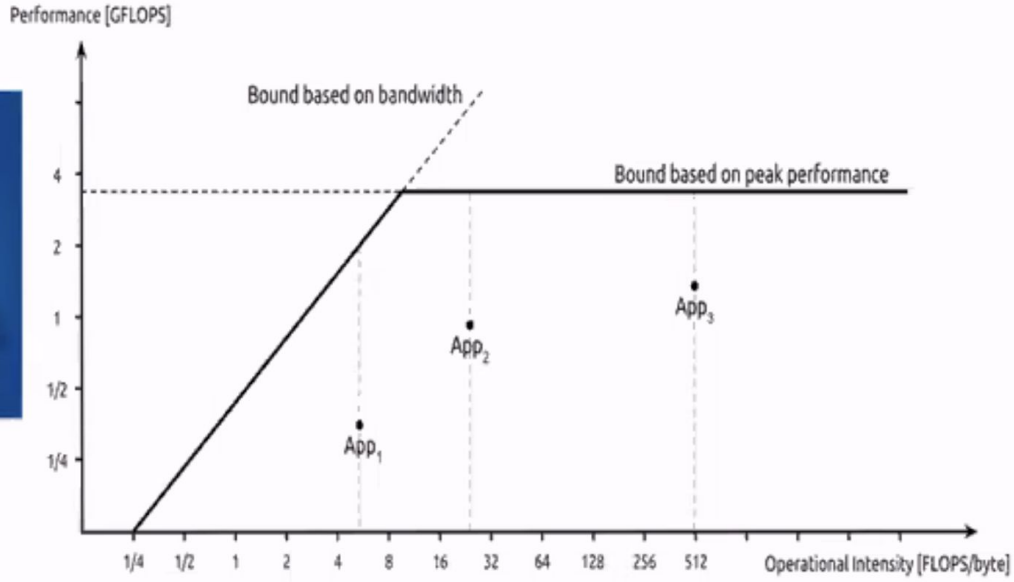


Image Sources: Futurology

On x-axis, we have FLOPS/Number of operations

Memory/Compute Bound Operations

- #Params and MAC could be misleading
  - Ignores memory movement cost
- The right metric is **Arithmetic Intensity**

AI = FLOP / memory operations

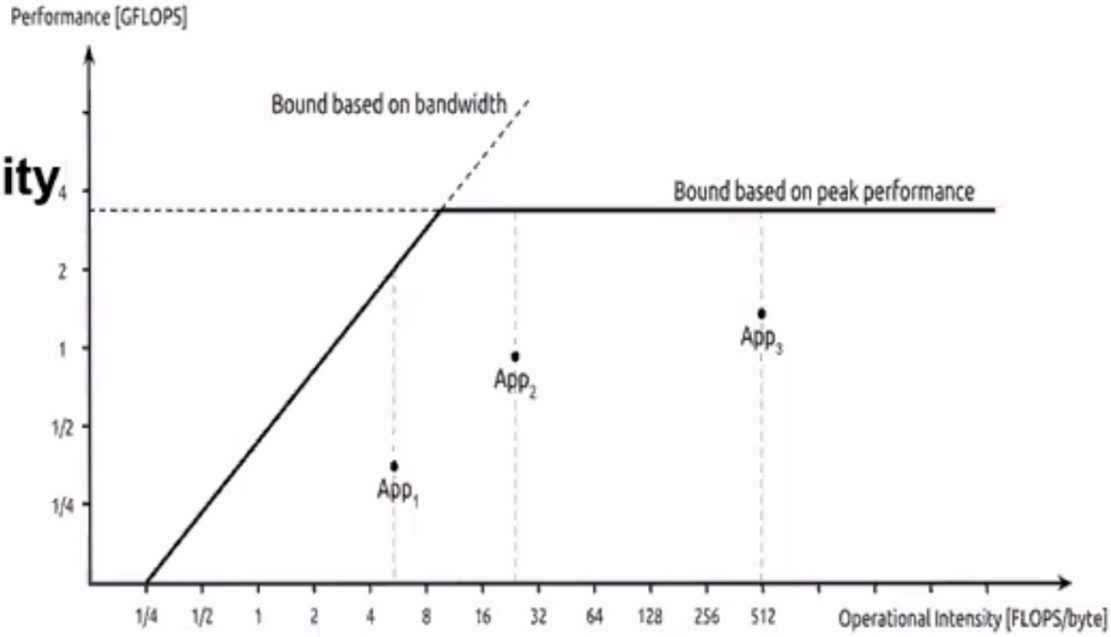
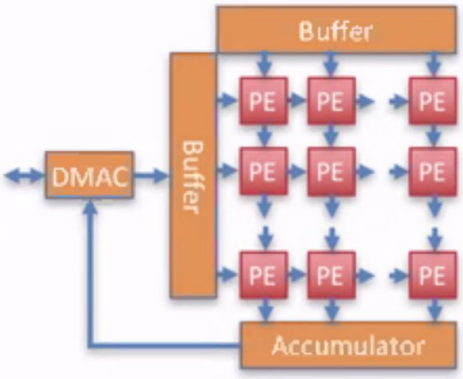
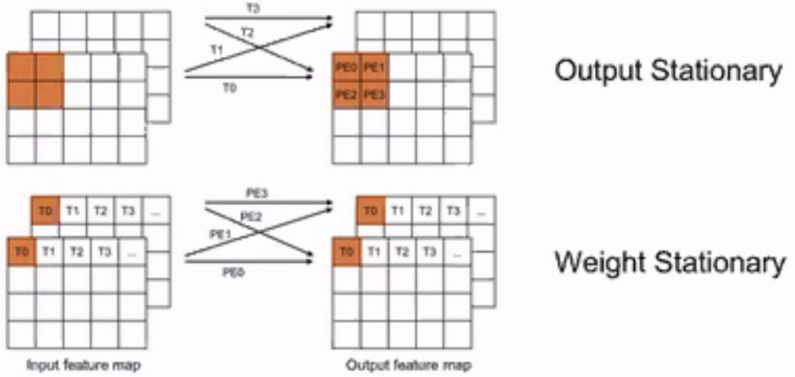


Image Sources: Supermicro X6QT8 Quad Core CPU System, and Wikipedia

SqueezeNext: Hardware Aware Neural Network Design

- Squeezelerator: Hybrid WS and OS data flow
  - Optimal execution flow statically based on trained weights (one time setup cost per network)





A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, K. Keutzer, SqueezeNext: Hardware-Aware Neural Network Design, ECV Workshop at CVPR'18, 2018.
K. Kwon, A. Amid, **A. Gholami**, B. Wu, K. Keutzer, Co-Design of Deep Neural Nets and Neural Net Accelerators for Embedded Vision Applications, Design Automation Conference (DAC'18), 2018.

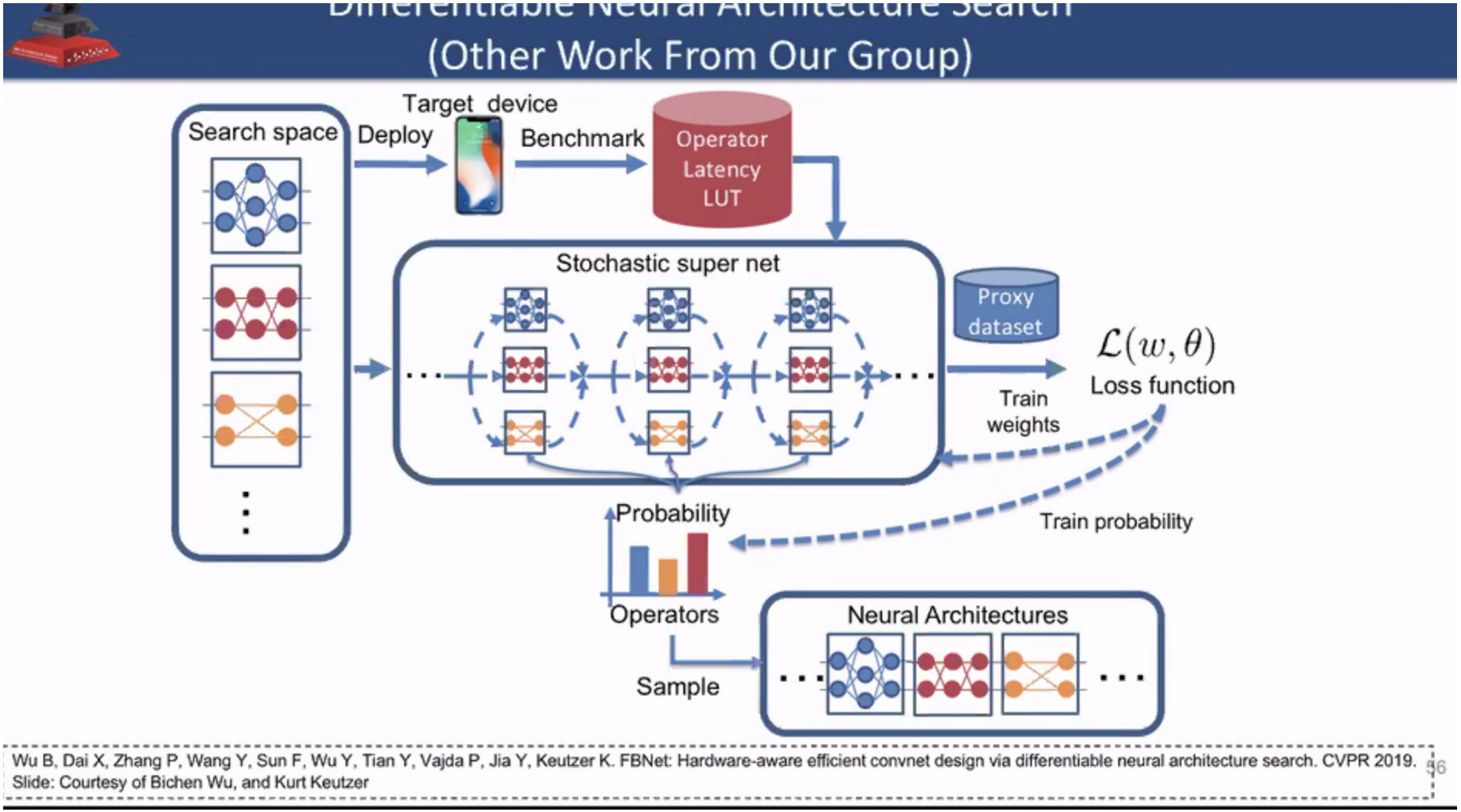
SqueezeNext: Hardware Aware Neural Network Design

- #Params and MAC could be misleading
- Smaller FLOP does not mean smaller Energy and faster inference
  - Memory movement could become the bottleneck
- **Important Point:** Even though SqueezeNext has higher MAC as compared to MobileNet but it uses less energy
  - The reason for this is that **Arithmetic Intensity** is the right metric, not MAC

Model	Params	MAC	Energy(xE9)
MobileNetV1	**4.2M**	**574M**	**5.8**
SqueezeNext	**3.2M**	**708M**	**5.4**

A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, K. Keutzer, SqueezeNext: Hardware-Aware Neural Network Design, ECV Workshop at CVPR'18, 2018.

MAC: Multiply-Accumulate Operation



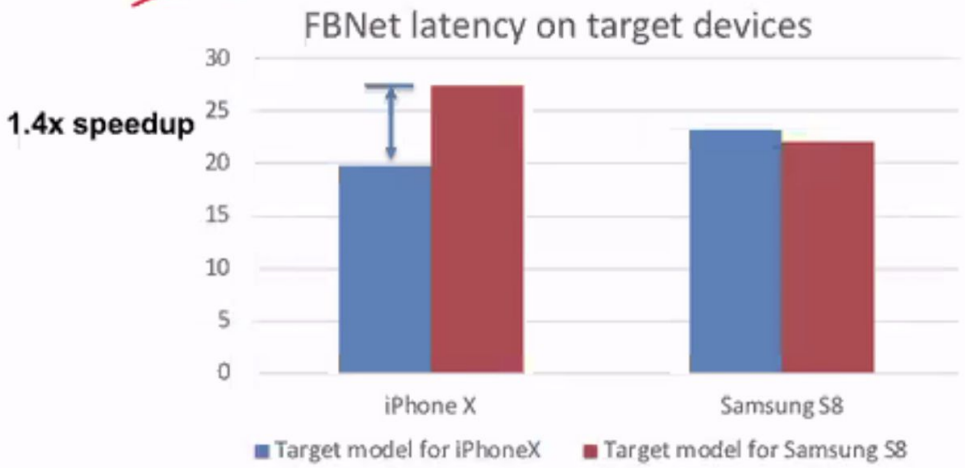
Total loss incorporates both accuracy as well as push towards low latency to output architectures

Result: FBNet for different target devices (Other Work from Our Group)

- Apple A11
- Big: 2 ARMv8 @ 2.5 GHz
- Little: 4 ARMv8 @ 1.4 GHz
- Vectorization: 4-wide 32-bit MAC
- LPDDR4x memory (30 GB/s)
- GPU + Neural Processing Engine

- Snapdragon 835
- Big: 4 ARMv8 @ 2.4 GHz
- Little: 4 ARMv8 @ 1.9 GHz
- Vectorization: 4-wide 32-bit MAC
- LPDDR4x memory (30 GB/s)
- Adreno 540 GPU

- Under similar accuracy constraint (73.27% vs 73.20%), FBNet optimized for iPhone-X achieves 1.4x speedup over the Samsung optimized model



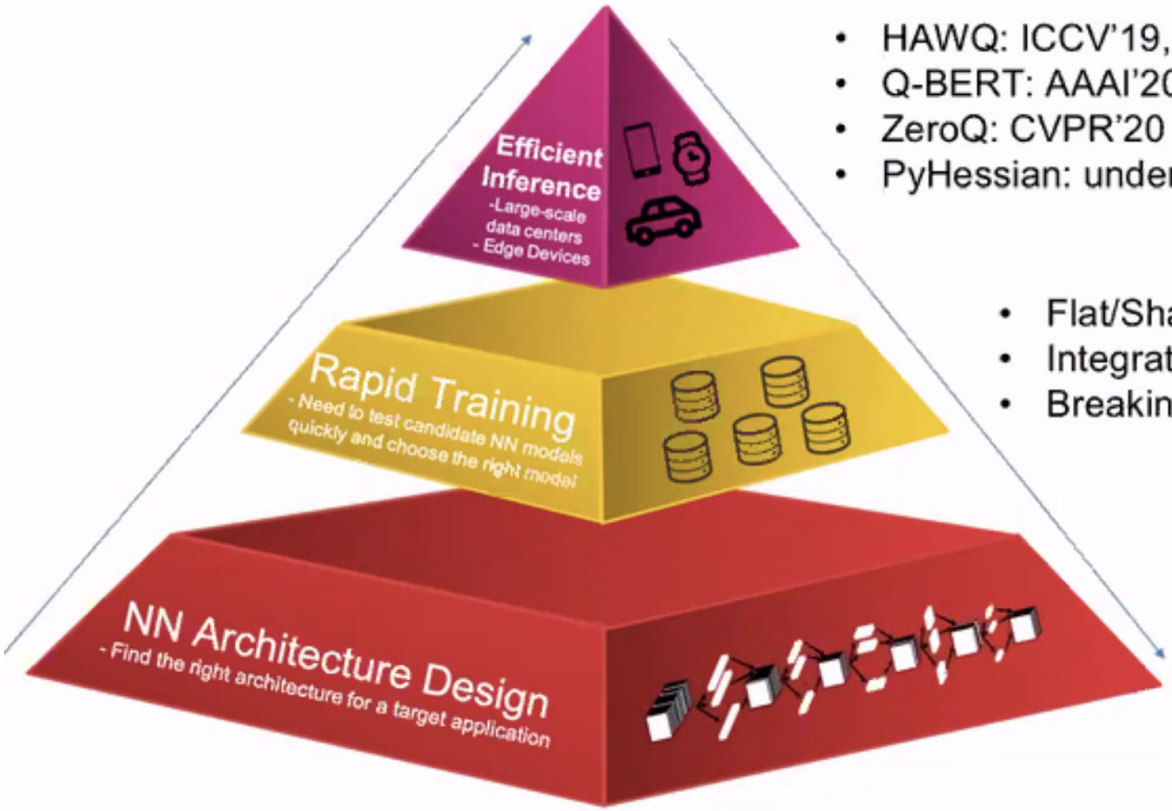
Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. CVPR 2019.
Slide: Courtesy of Bichen Wu, and Kurt Keutzer

Summary

- FLOPS or #Params is not the correct metric to measure efficiency
  - The right metric is **Arithmetic Intensity** which is **hardware specific**

- The next milestone is to co-design NN architecture and hardware
  - SqueezeNext was an early work followed by automated DNAS but much more work is left to do

An Integrated Approach to DNN Design



- HAWQ: ICCV'19, NeurIPS'19
- Q-BERT: AAAI'20
- ZeroQ: CVPR'20
- PyHessian: under review

- Flat/Sharp Minima with Large Batches: NeurIPS'18
- Integrated Parallelism: SPAA'18
- Breaking GPU Memory Wall: MLSys'20

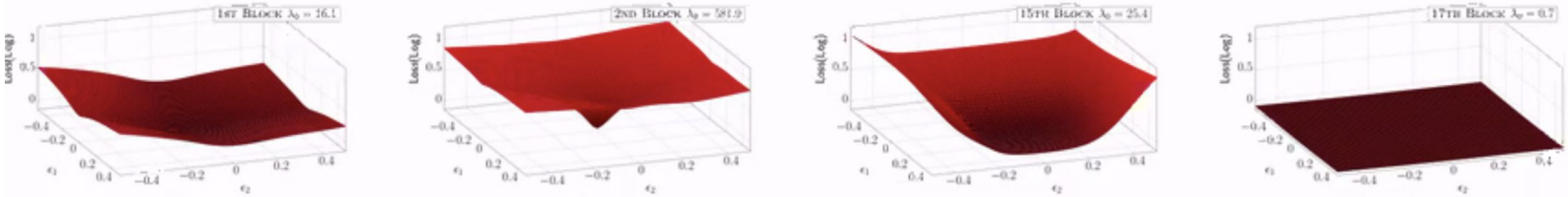
- Shift: CVPR'18
- SqueezeNext: DAC'18; ECV CVPR'18

Future Work: Adaptive Mixed Precision Training

- The precision used to train the model **must depend** on the **loss landscape**
  - Solutions that ignore this are not generalizable

- **Adaptive precision** is needed for **different layers**

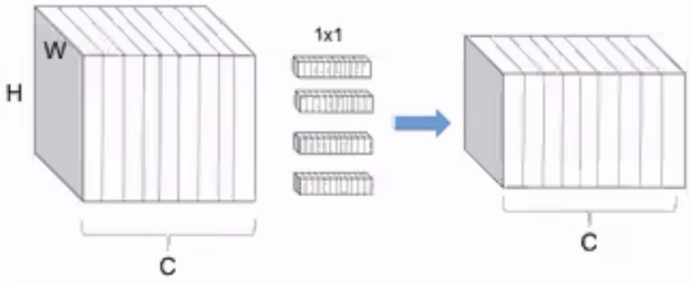
$$\min_{w} \mathcal{J}(w) = \frac{1}{N} \sum_{i=1}^{N} cost(w, x_i)$$

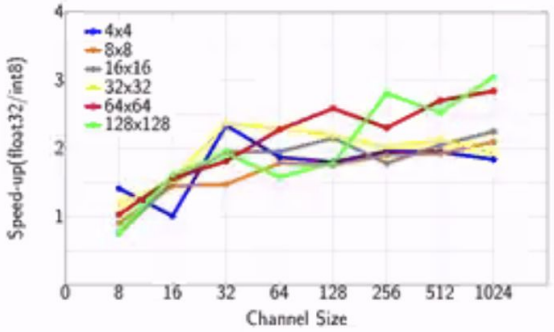


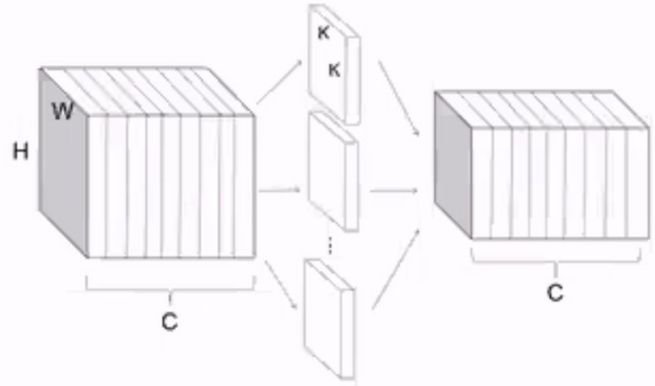
Zhang T, Lin Z, Yang G, De Sa C. QPyTorch: A Low-Precision Arithmetic Simulation Framework. arXiv preprint arXiv:1910.04540. 2019.
Jianyu Huang James Reed Daya Khudia, Jongsoo Park. Fbgemm. 2018. URL https: //github.com/dskhudia/FBGEMM.

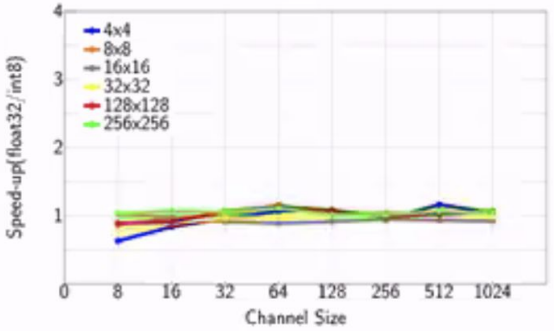
Future Work: Platform Aware Quantization

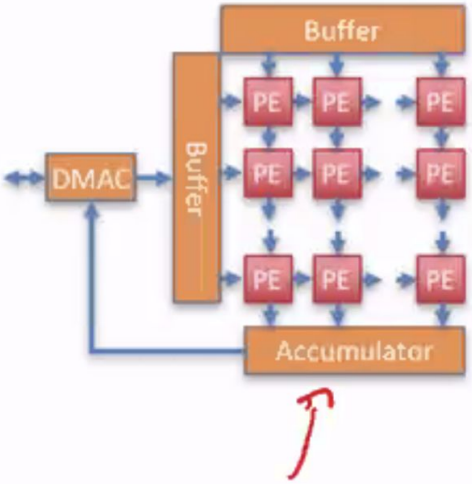
- Gains from quantization/compression are platform specific
- Need to include this information along with the Hessian for quantization for optimal speed ups





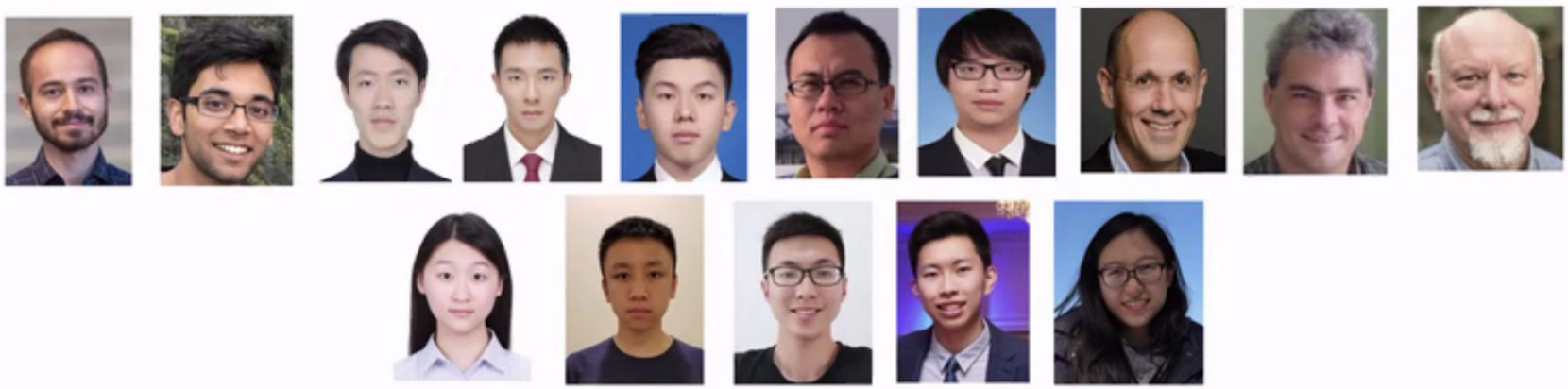






Thank You for Listening!

amirgh@berkeley.edu



Berkeley
UNIVERSITY OF CALIFORNIA