Quantization Methods for Efficient Neural Networks

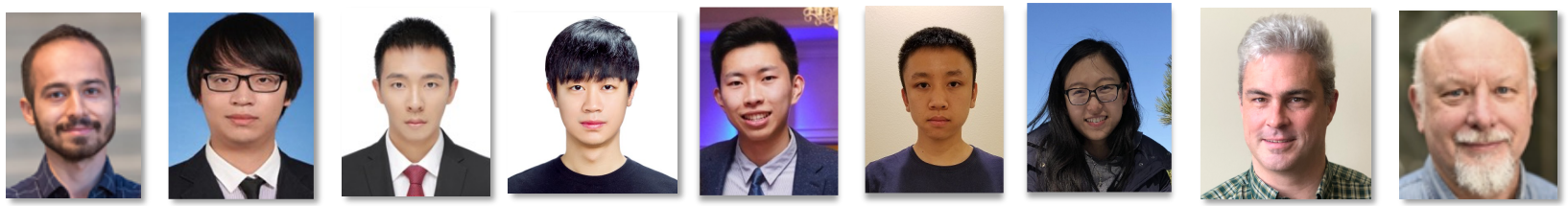
Amir Gholami, and

Z. Yao, Z. Dong, S. Kim, Z. Zheng, E. Tan, Q. Huang, Michael Mahoney, Kurt Keutzer, and Pallas Group

University of California Berkeley

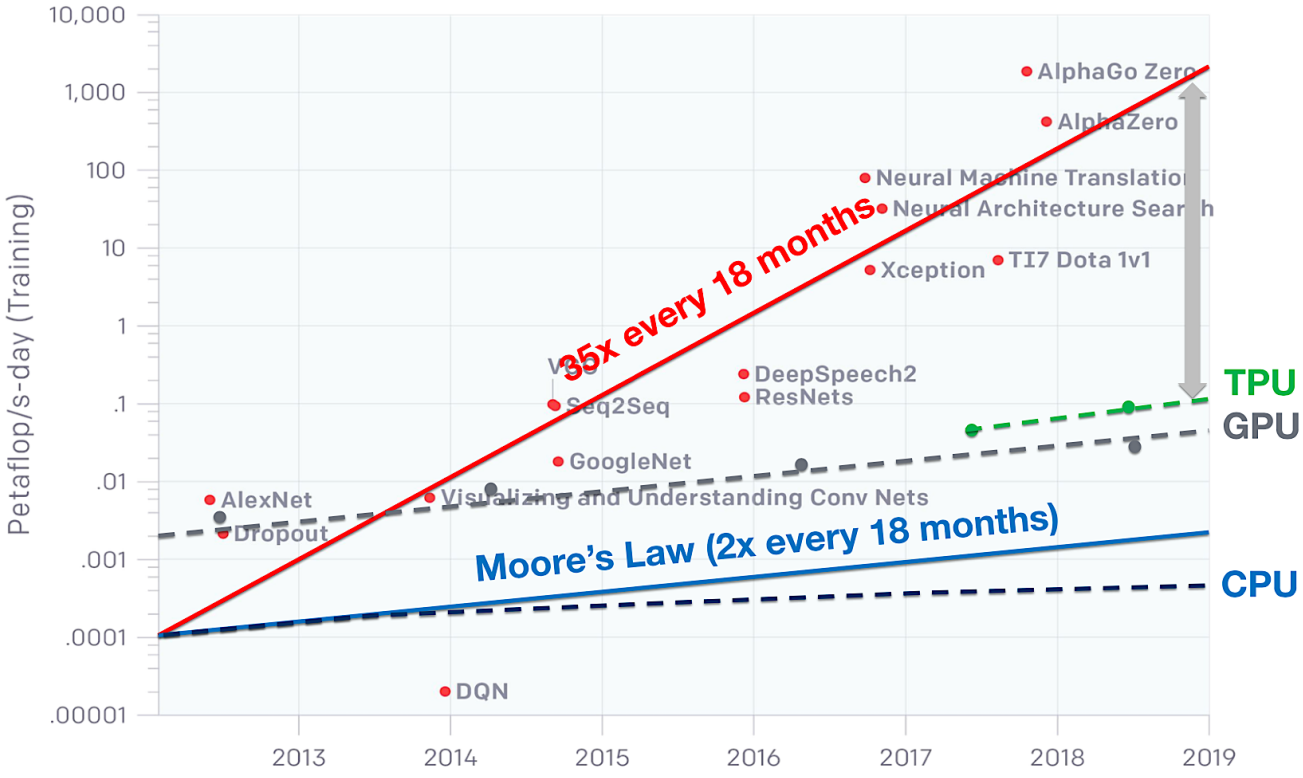
Prof. Shao's class on Hardware for Machine Learning

(EE290, Spring 2021)



AI and Compute

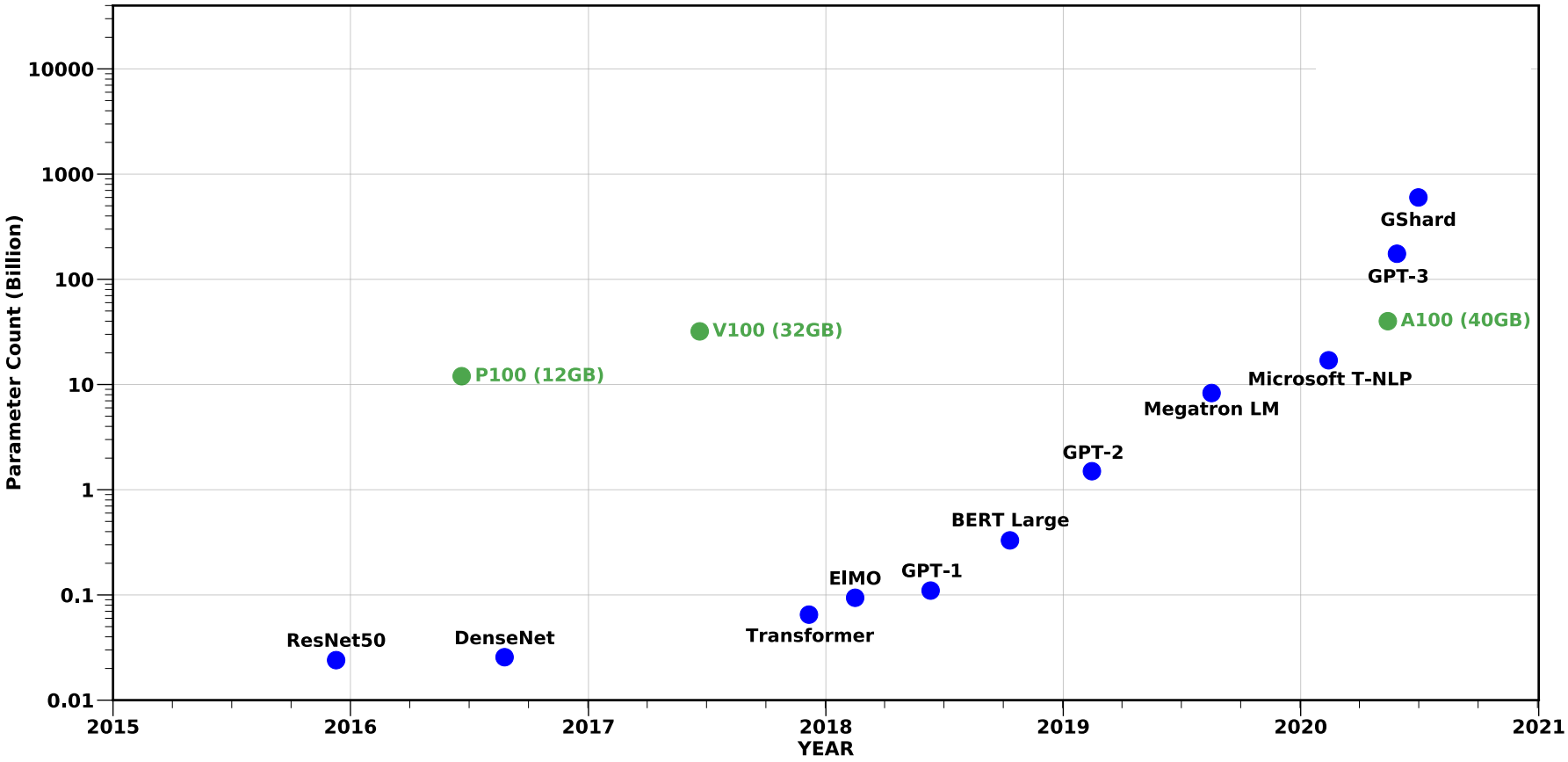
- Quantization can help improve the training the speed



AI and Compute

AI and Memory Wall

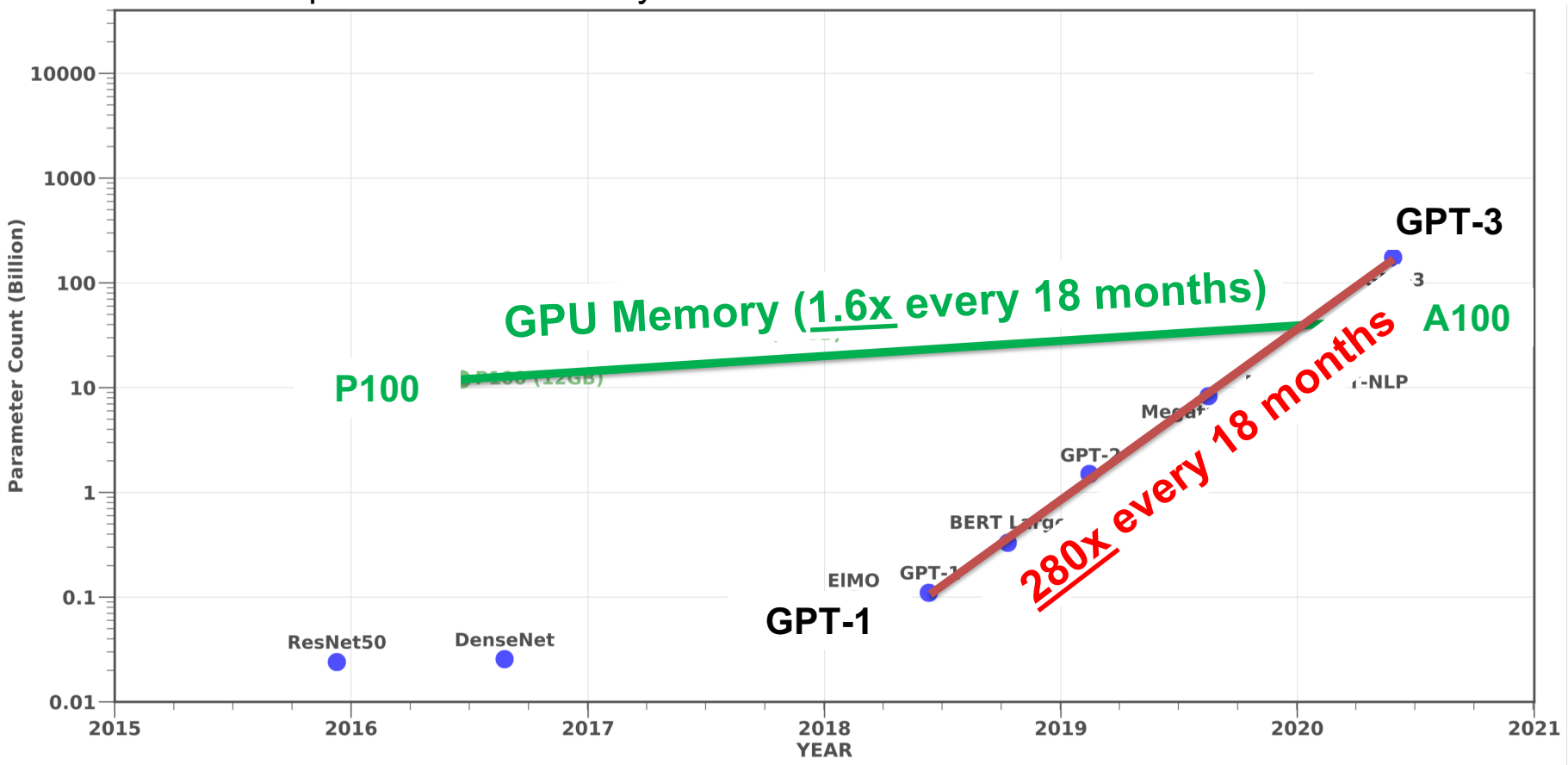
- Quantization can help reduce the memory overhead



Amir Gholami, AI and Memory Wall, Blogpost, 2021.

AI and Memory Wall

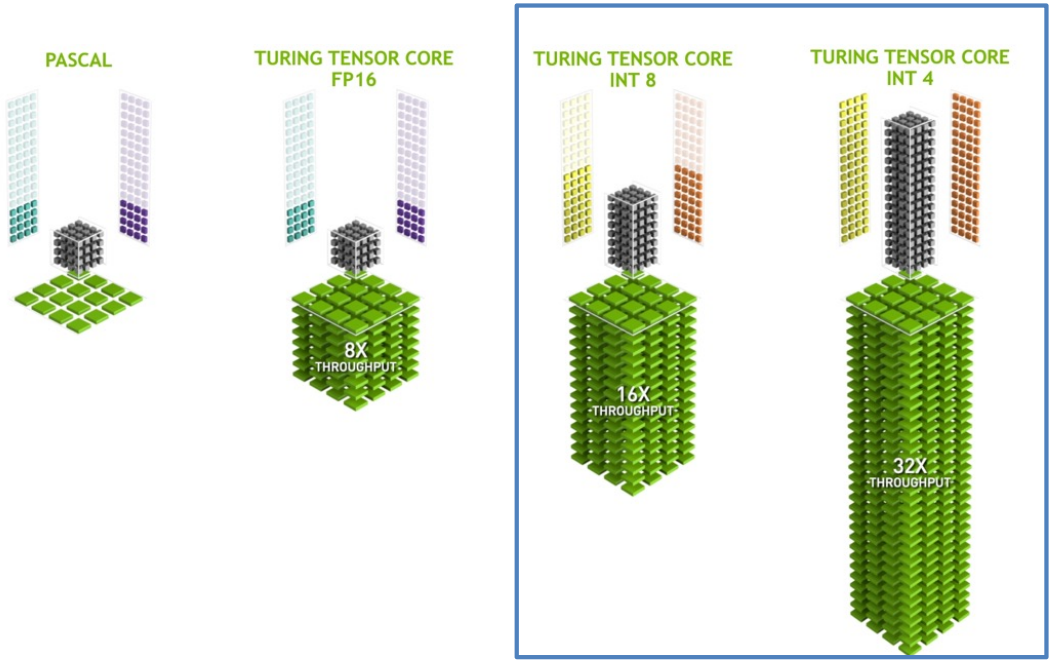
- Quantization can help reduce the memory overhead



Amir Gholami, AI and Memory Wall, Blogpost, 2021.

Lower precision Multiply-Acc Has higher Throughput

- Lower precision weights mean less energy per Multiply-Accumulate
- Also enables putting more MAC units per unit of silicon



Big opportunity to enable lower bit precision inference!

Quantization Produces Lower Latency

Network	Batch Size 1		Batch Size 8	
	FP32	INT8	FP32	INT8
MobileNet v2	1.0	**1.9**	1.0	**4.0**
ResNet50 v1.5	1.0	**3.5**	1.0	**7.3**
VGG-19	1.0	**3.1**	1.0	**7.0**
Inception V4	1.0	**4.4**	1.0	**7.1**

- Quantization improves latency, sometimes **superlinearly**

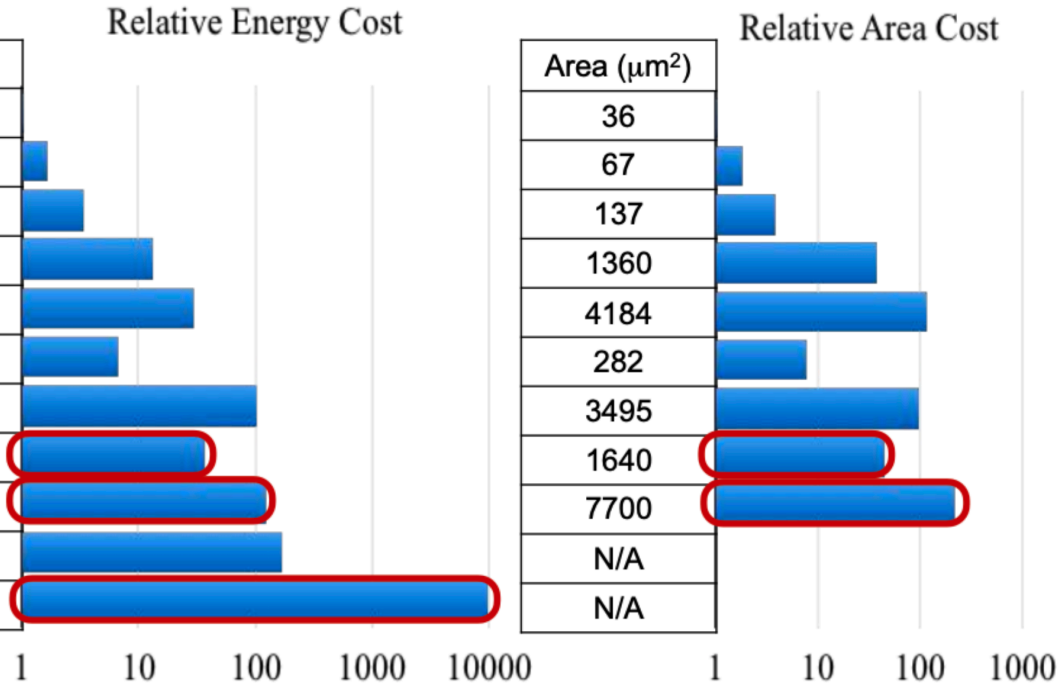
Results from TensorRT 7.0 (credit: P. Judd)

Energy Consumption

Quantized arithmetic as compared to FP arithmetic has order of magnitude less:

- **Energy Cost**
- **Area Cost**

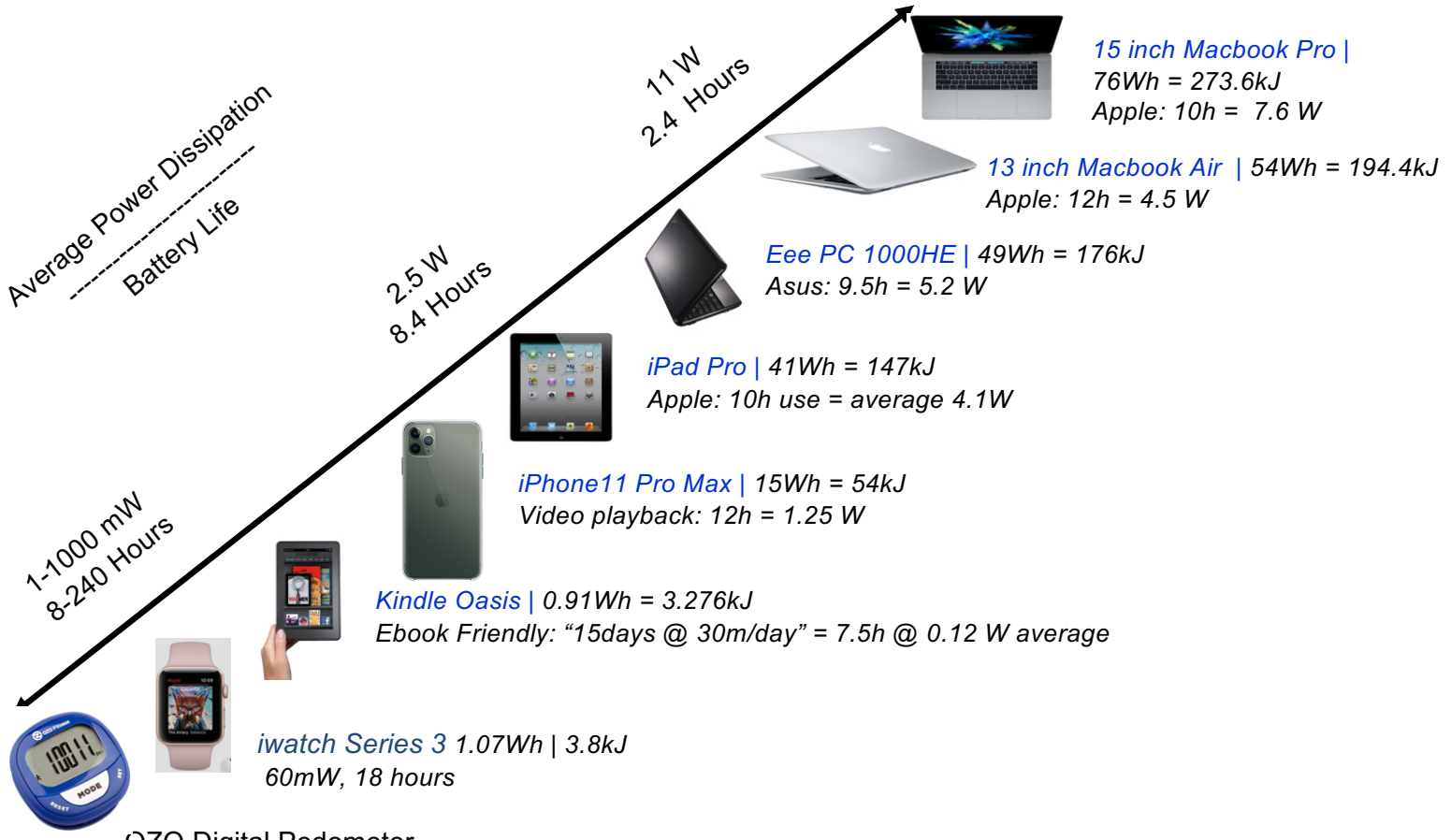
Operation:	Energy (pJ)	Area (μm²)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



"computing's Energy Problem, M. Horowitz, ISSCC, 2014
(Numbers are rough approximations for 45nm)

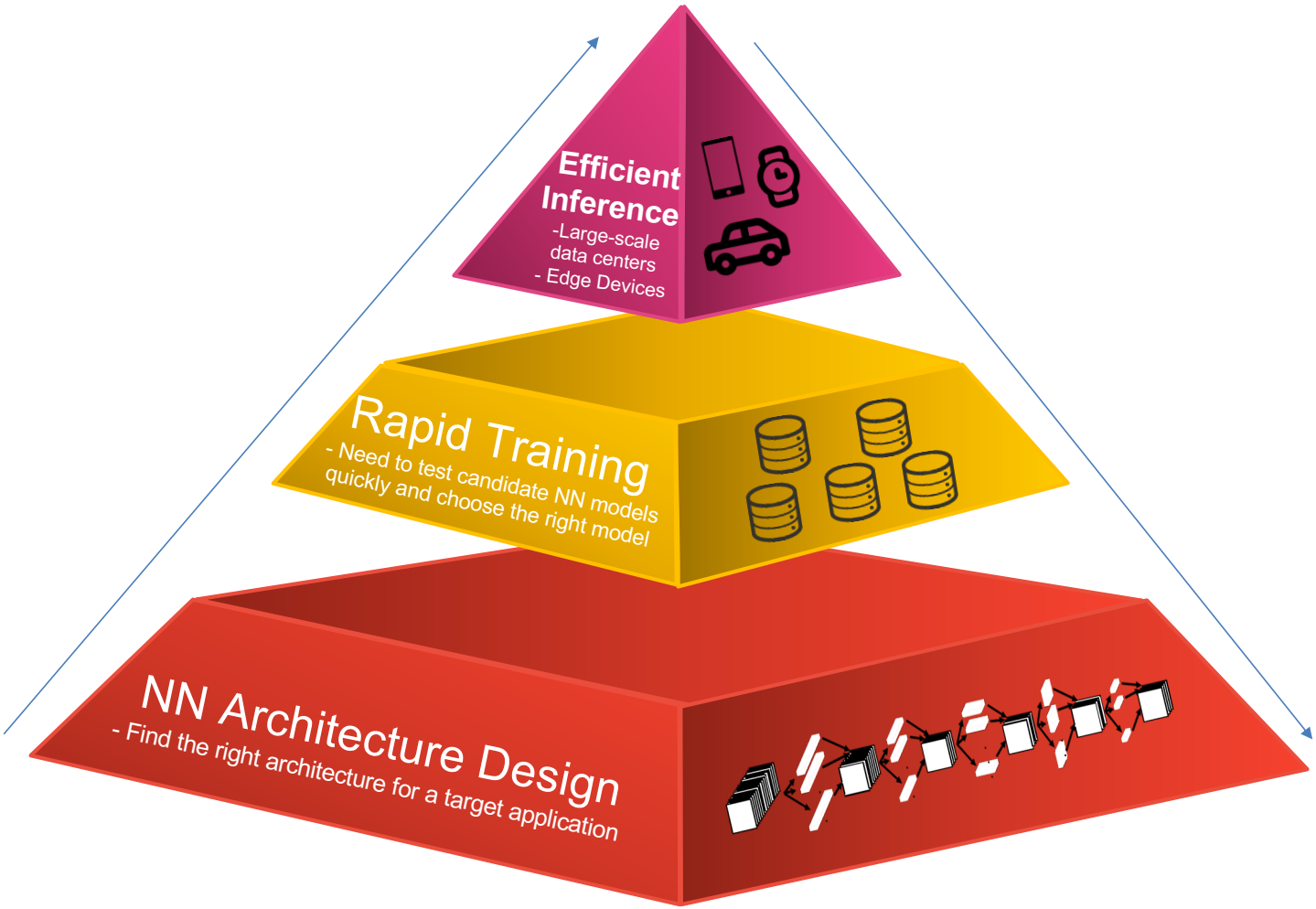
Slide: Courtesy of Prof. Shao

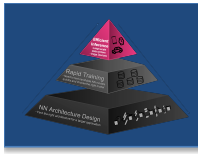
Want to operate in Low Power Regimes to be used in Mobile Devices



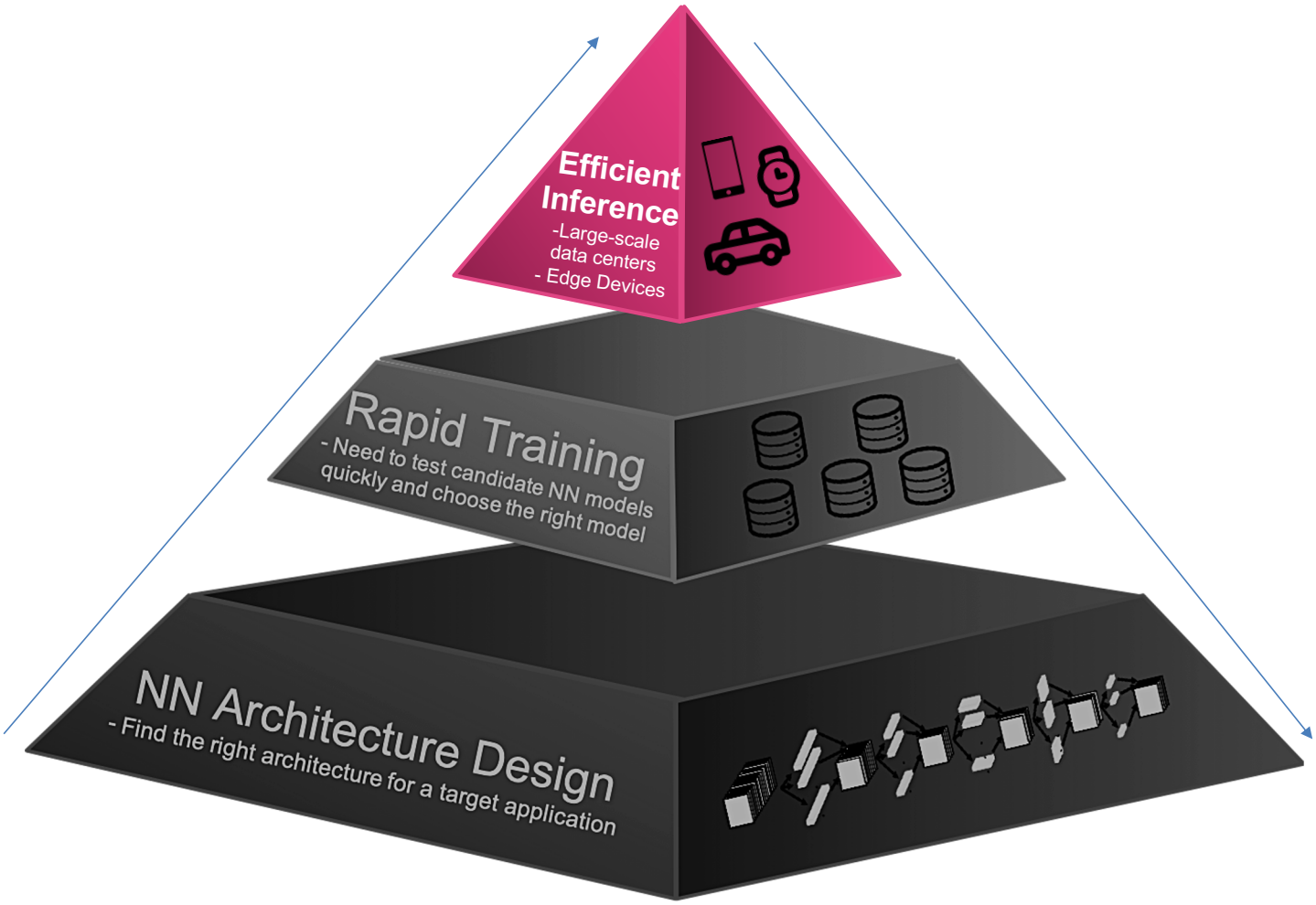
Slide: Courtesy of Xiangyu Yu, and Kurt Keutzer

An Integrated Approach to DNN Design





An Integrated Approach to DNN Design



Outline

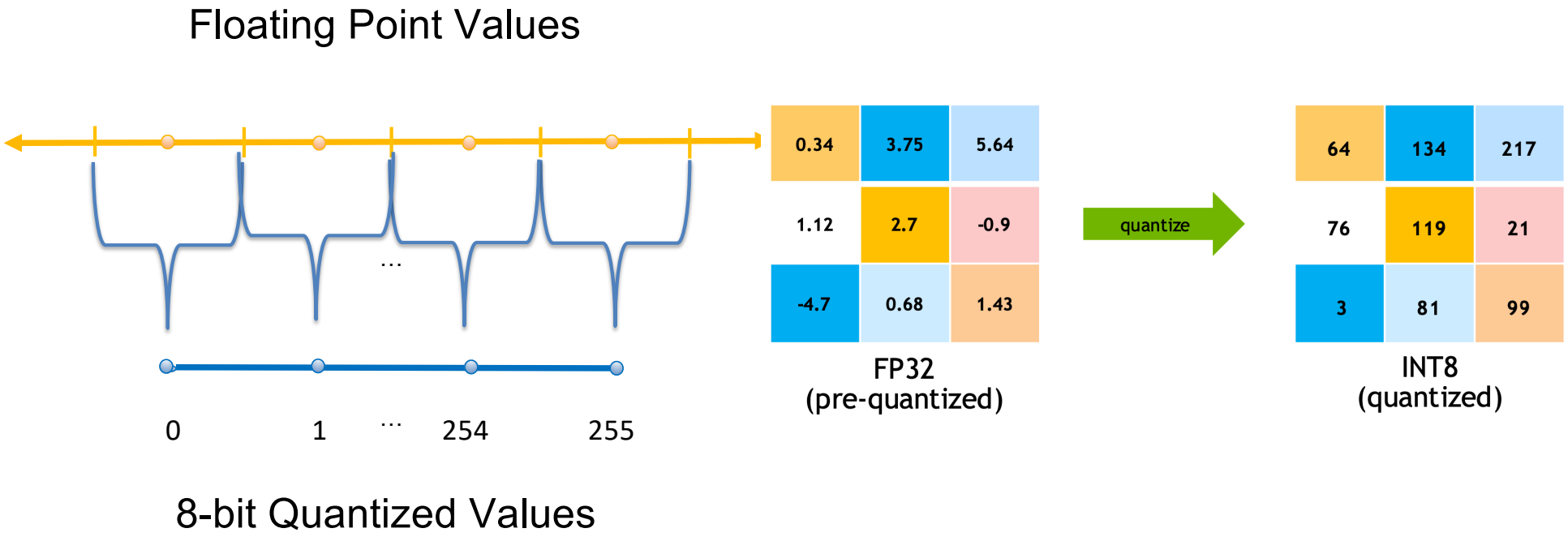
- Basic Concepts of Quantization
- Advanced Concepts of Quantization
- Co-design of Neural Network and Hardware

Outline

- **Basic Concepts of Quantization**
  - Uniform vs Non-Uniform Quantization
  - Symmetric vs Asymmetric Quantization
  - Quantization Granularity: Layer-wise vs Channel-wise
  - Dynamic vs Static Quantization
  - Post Training Quantization vs Quantization Aware Training

Quantization: Workhorse for Efficient Inference

- Uniform quantization is a linear mapping from floating point values to quantized integer values

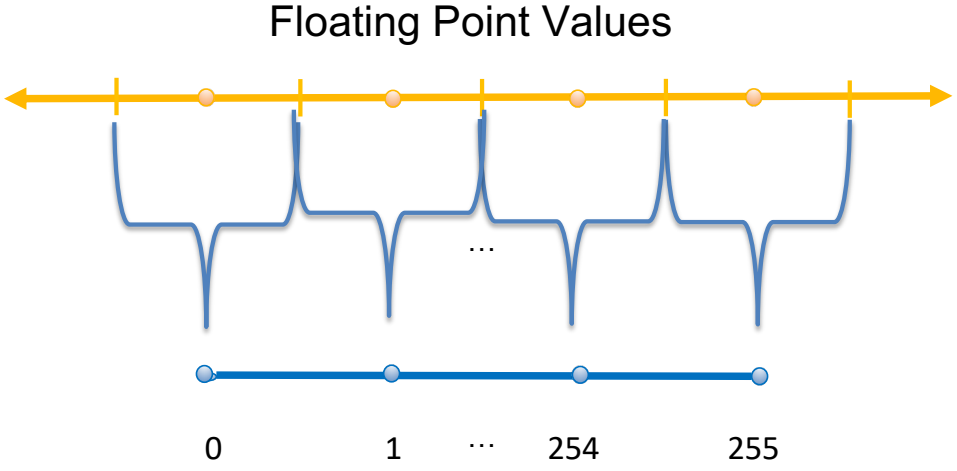


Uniform Quantization

- 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

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

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



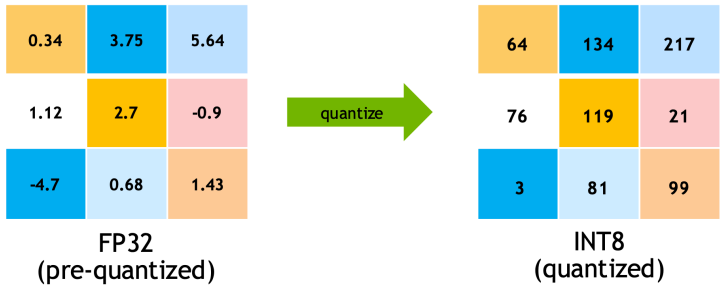
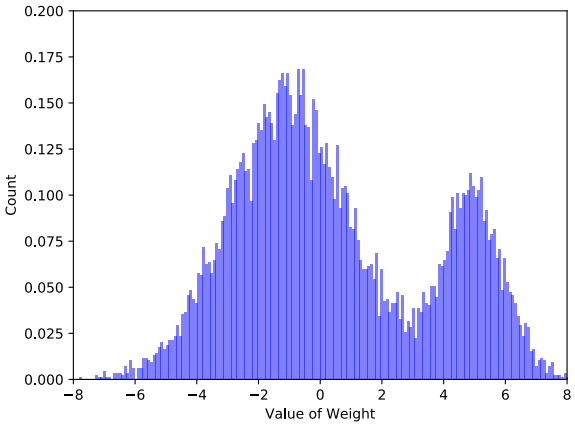
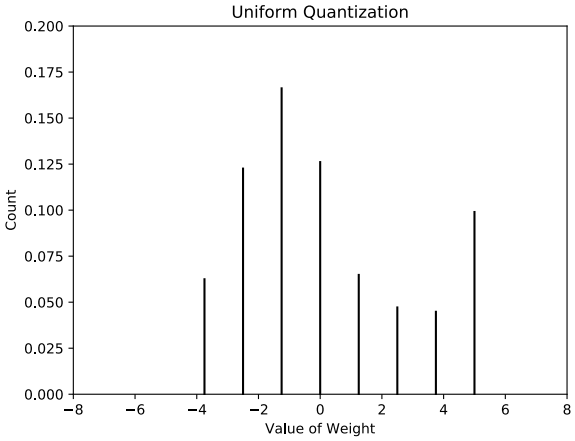
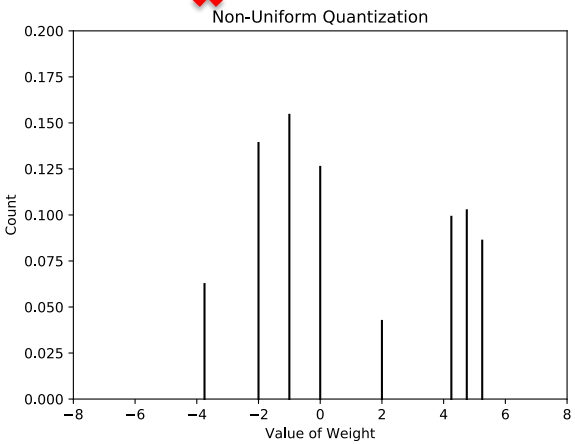


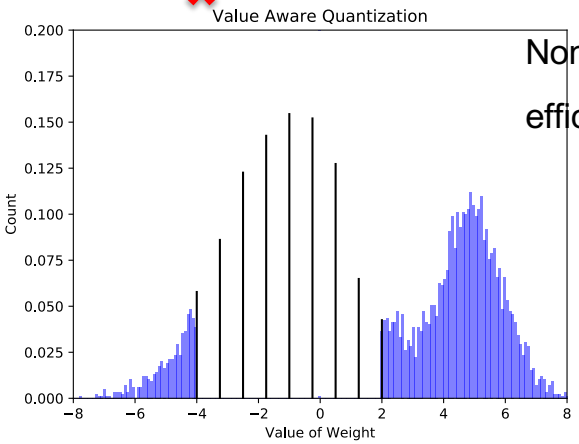
Illustration from Sahni Manas

Non-Uniform Quantization









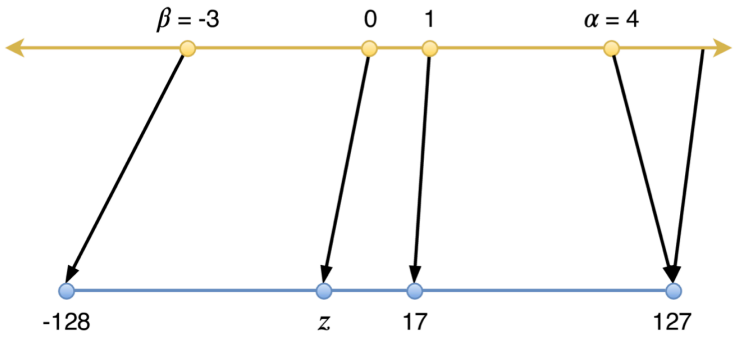
Non-uniform quantization is not efficient for hardware deployment

Park, Eunhyeok, Sungjoo Yoo, and Peter Vajda. "Value-aware quantization for training and inference of neural networks." ECCV, 2018.

Symmetric vs Asymmetric Quantization

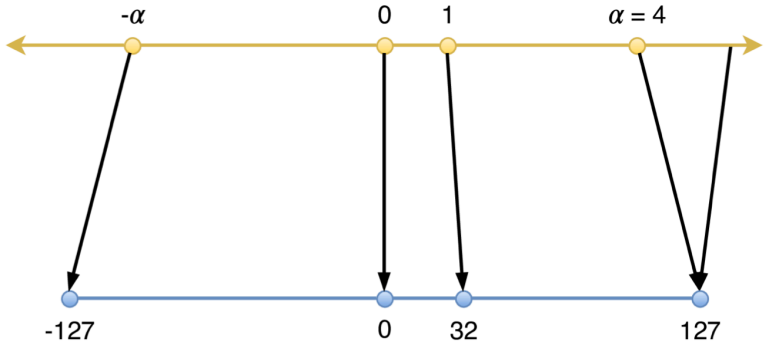
- How do we choose the range? In general, the range does not have to $[r_{min}, r_{max}]$

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



Asymmetric Quantization: no constraint on β, α

$$r = \frac{\beta - \alpha}{2\hat{}B - 1}(q - z)$$



Symmetric Quantization: $\beta = -\alpha$

$$r = \frac{-2\alpha}{2\hat{}B - 1}(q - z)$$

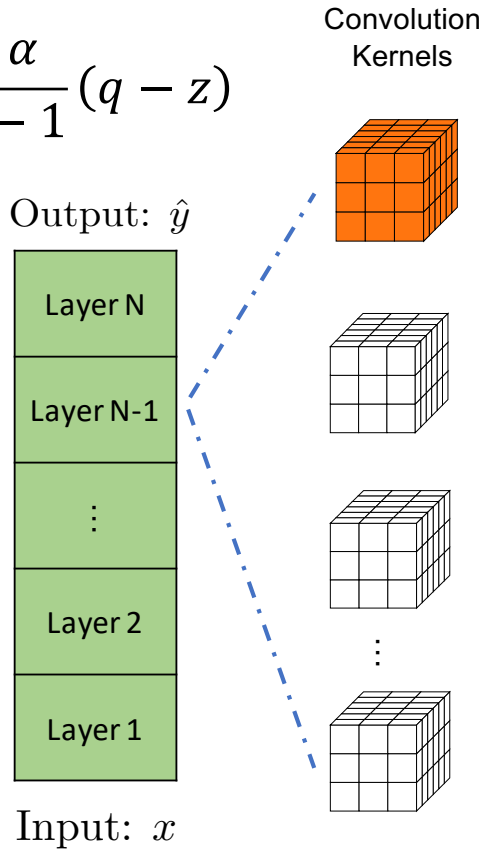
- For both cases the range can be different than min/max. A good example, is percentile where we choose to keep a percentage of the values instead of all of them (say 98%)

Quantization Granularity: Layer-wise vs Channel-wise

$$r = \frac{\beta - \alpha}{2\text{^}B - 1}(q - z)$$

- Layer-wise Quantization:
  - Use the **same clipping range** $[\beta, \alpha]$ for all the convolution kernels in a layer
  - Usually sub-optimal since the weights in a layer can have different ranges

- Channel-wise Quantization:
  - Use the **different clipping range** $[\beta, \alpha]$ for all the convolution kernels in a layer
  - Usually results in better accuracy as the range can be better captured for each output channel
  - Has negligible overhead



Dynamic vs Static Quantization

- How do we choose the clipping range $[\beta, \alpha]$? $r = \frac{\beta - \alpha}{2\hat{}B - 1}(q - z)$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \\ w_{40} & w_{41} & w_{42} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\boldsymbol{y} = W \cdot \boldsymbol{x}$$

  - For weights, we know the values **statically,** since weights are fixed during inference
  - **But what about activations? We can either use static or dynamic quantization:**

- Static Quantization: Choose pre-determined static range for activations (x,y), independent of input
  - Very fast, low overhead, but typically not accurate since each input can have a different range

- Dynamic Quantization: Determine range for each activation separately during the runtime
  - Typically very slow due to the cost of computing mix/max or percentile
  - But very accurate as it exactly detects the correct range for quantization

Post Training Quantization (PTQ) vs Quantization Aware Training (QAT)

- There is also another classification:

- Post Training Quantization (PTQ):
  - Input is a model trained at higher precision after training is finished
  - Quantization is performed by analyzing weights/activations **without fine-tuning**

- Quantization Aware Training (QAT):
  - Input is a model trained at higher precision after training is finished
  - Quantization is performed along with possibly **several epochs of fine-tuning**

Post Training Quantization (PTQ) vs Quantization Aware Training (QAT)

PTQ	QAT
Usually fast	Slow
No re-training of the model	Model needs to be trained/finetuned
Plug and play of quantization schemes	Plug and play of quantization schemes (requires re-training)
Less control over final accuracy of the model	More control over final accuracy since *q-params* are learned during training.

Slide from D. Peri, J. Patel, J.Park, Deploying Quantization-Aware Trained Networks Using TensorRT, 2020.

Review

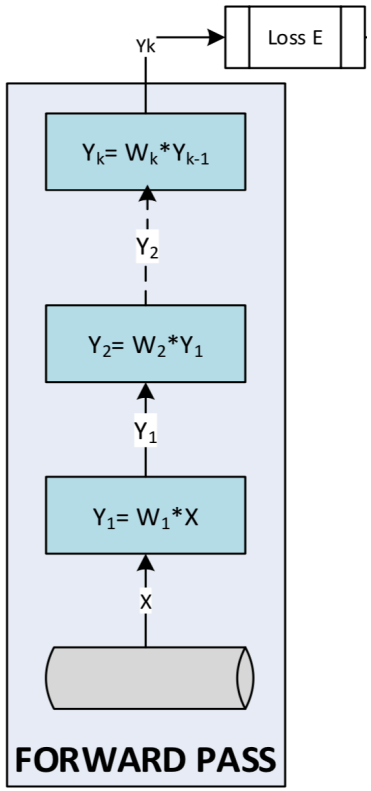
- **Basic Concepts of Quantization**
  - Uniform vs Non-Uniform Quantization
  - Symmetric vs Asymmetric Quantization
  - Quantization Granularity: Layer-wise vs Channel-wise
  - Dynamic vs Static Quantization
  - Post Training Quantization vs Quantization Aware Training

Outline

- Basic Concepts of Quantization
- **Advanced Concepts of Quantization**
  - Fake Quantization vs Integer-only Quantization
  - Uniform vs Mixed-Precision Quantization
- Co-design of Neural Network and Hardware

Fake (Simulated) Quantization vs Integer-only/Fixed-Point/Dyadic Quantization

Let's take a closer look at a layer



$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \\ w_{40} & w_{41} & w_{42} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\boldsymbol{y} = W \cdot \boldsymbol{x}$$

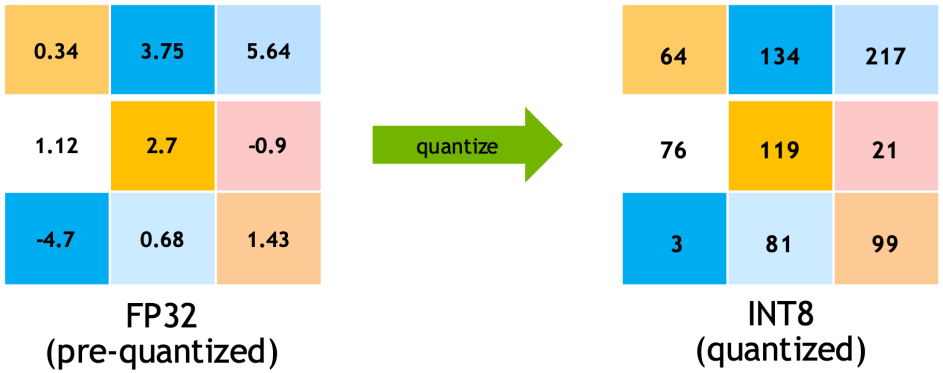


Illustration from Sahni Manas

Closer Look at One Layer

For simplicity, let's consider symmetric quantization => z=0

We first need to accumulate results in INT32 and then rescale back to INT4

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \\ w_{40} & w_{41} & w_{42} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\boldsymbol{y} = W \cdot \boldsymbol{x}$$

$$y^{i32} = W^{i4} x^{i4}$$

$$y^{i4} = \frac{\alpha_x \alpha_w}{\alpha_y} y^{i32}$$

$$y^{i4} = Round(S y^{i32})$$

$$(\alpha_y, y^{i4}) = (\alpha_w, W^{i4}) \cdot (\alpha_x, x^{i4})$$

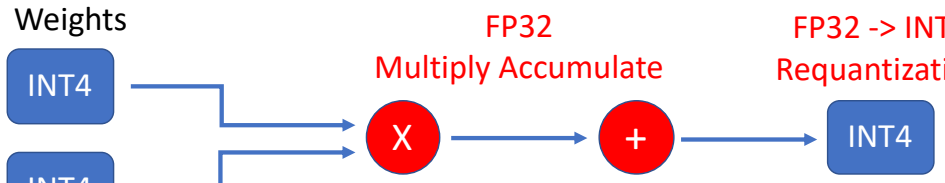
This is Integer-only (aka fixed-point/Dyadic quantization)

But not all quantization algorithms use this method

Fake Quantization!

Fake Quantization:

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



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

docs.oracle.com

What Every Computer Scientist Should Know About Floating-Point Arithmetic

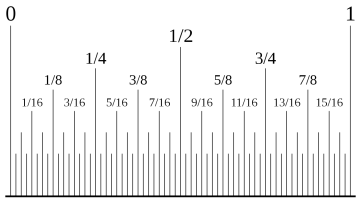
D Goldberg · 1991

Integer-only Quantization

$$(S_w, W^{i4}) \qquad a^{\text{i32}} = W^{\text{i4}} h^{\text{i4}} \qquad a^{i4} = Int(\frac{S_w S_h}{S_a} a^{\text{i32}})$$

$$(S_h, h^{i4})$$

$$\frac{S_w S_h}{S_a} = \frac{\alpha}{2^{\beta}}$$

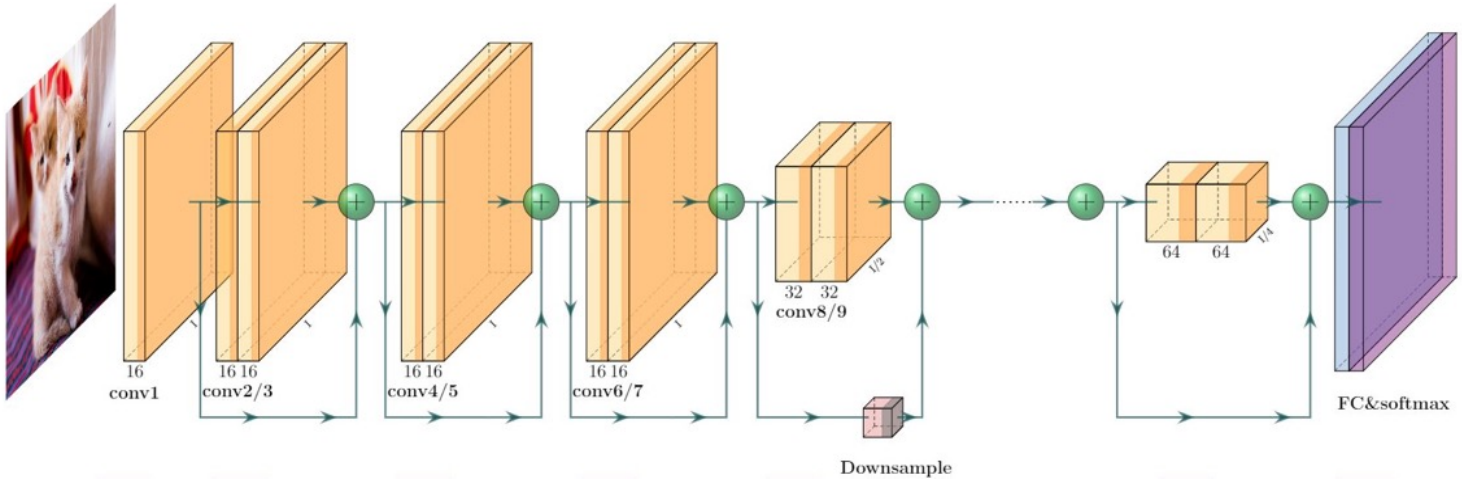


Dyadic Number

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, arxiv:2011.10680, 2020.

Fake Quantization!

- What is the impact of fake quantization? Are the errors considerable?
- Yes. The errors become quite large for low bit precision.
- Below we report the relative L2 error between integer-only and fake quantization for ResNet50 on ImageNet with INT4 precision



Layer 4: 1.05% Layer 24: 58.60% Layer 50: 97.56% Last Layer/Logits: 65.50%

$$\text{Relative L2 Error: } \frac{\|a-b\|_2}{\|a\|_2}$$

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, arxiv:2011.10680, 2020.

Fake Quantization

- In fake quantization weights/activations are converted to FP32 and inference is performed using floating point arithmetic
  - This creates a mismatch which can become significant for low bit integer-only quantization
- Another important difference is that fake quantization uses FP32 for batch normalization layer
  - This is because BatchNorm is sensitive to quantization

We avoid both of these in integer-only quantization (HAWQV3):

- The entire inference is performed with integer-only quantization
- BatchNorm layer is quantized and folded into the convolution weights

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, arxiv:2011.10680, 2020.
https://github.com/zhen-dong/hawq
Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." CVPR'18, 2018.

Integer-only quantization works at 8-bits!

(a) *ResNet18*

Method	IntOnly	Uniform	Open Source	Precision	Size (MB)	BOPS (G)	Top-1
Baseline	✗	–	✓	W32A32	44.6	1858	71.47
RVQuant [39]	✗	✗	✗	W8A8	11.1	116	70.01
HAWQV3	✓	✓	✓	W8A8	11.1	116	**71.56**

(b) *ResNet50*

Method	IntOnly	Uniform	Open Source	Bit Precision	Size (MB)	BOPS (G)	Top-1
Baseline	✗	–	✓	W32A32	97.8	3951	77.72
Integer Only [29]	✓	✓	✗	W8A8	24.5	247	74.90
RVQuant [39]	✗	✗	✗	W8A8	24.5	247	75.67
HAWQV3	✓	✓	✓	W8A8	24.5	247	**77.58**

(a) *InceptionV3*

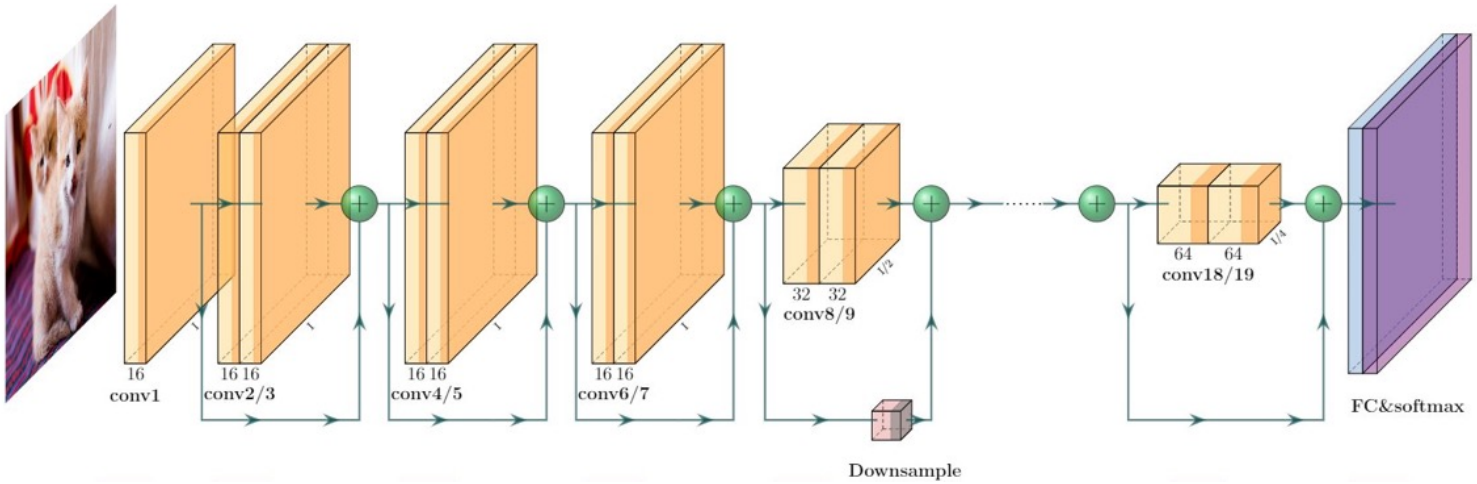
Method	IntOnly	Uniform	Open Source	Bit Precision	Size (MB)	BOPS (G)	Top-1
Baseline	✗	–	✓	W32A32	90.9	5850	78.88
Integer Only [29]	✓	✓	✗	W8A8	22.7	366	74.20
RVQuant [39]	✗	✗	✗	W8A8	22.7	366	74.22
HAWQV3	✓	✓	✓	W8A8	22.7	366	**78.76**

Can we go even further and perform lower 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, arxiv:2011.10680, 2020.
https://github.com/zhen-dong/hawq

Low Precision Quantization

Can we go even further and perform lower precision quantization?

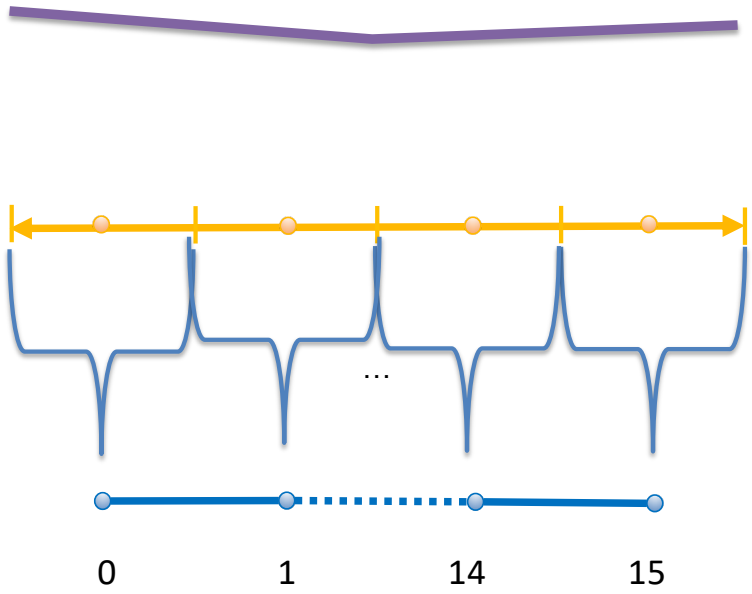


Uniform low precision does not work as it can significantly degrade accuracy

- **Use mixed-precision**

Flat Loss Landscape → Low Bit Precision

- Uniform quantization is a linear mapping from floating point values to quantized integer values



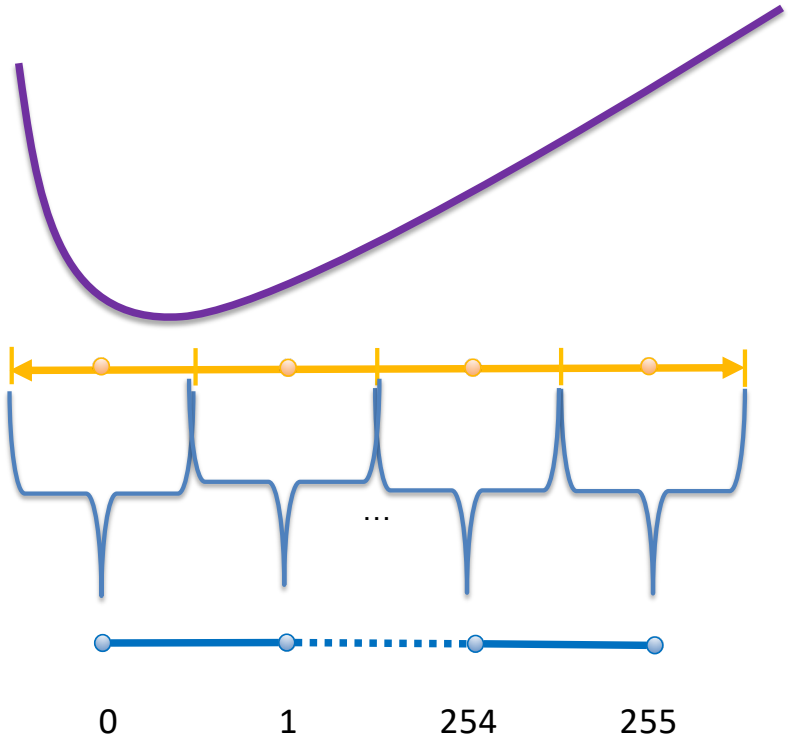
Flat Loss Landscape

Floating Point values

4-bit Quantization

Sharp Loss Landscape → High Bit Precision Needed

- Uniform quantization is a linear mapping from floating point values to quantized integer values

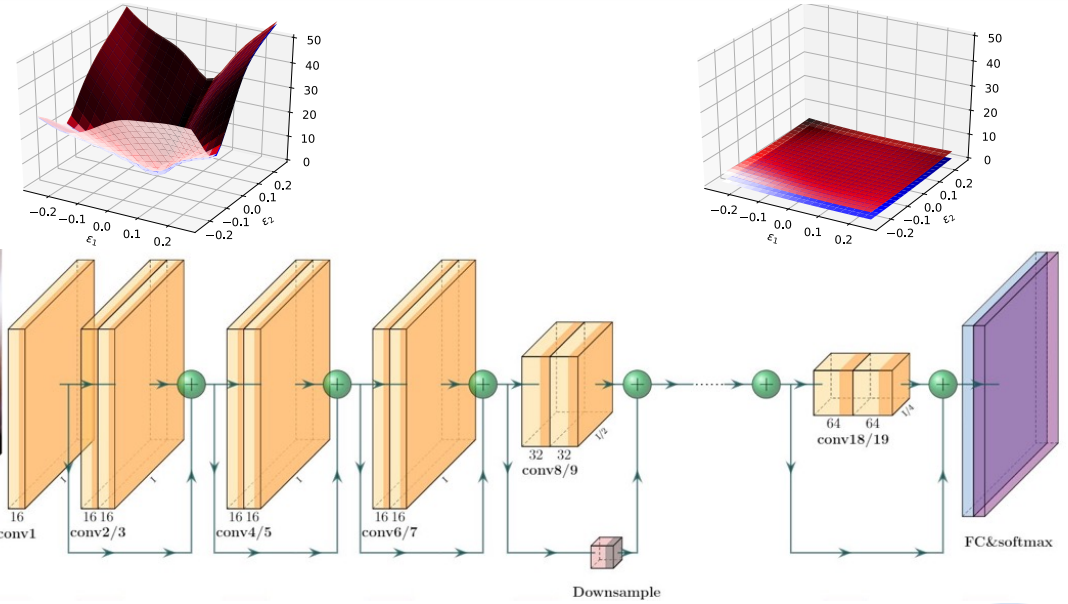


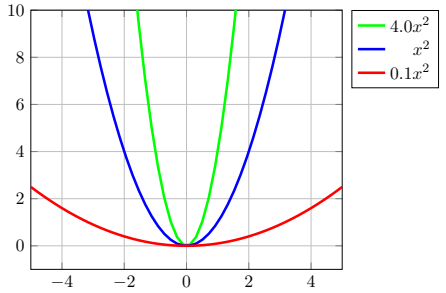
Sharp Loss Landscape

Floating Point values

8-bit Quantization

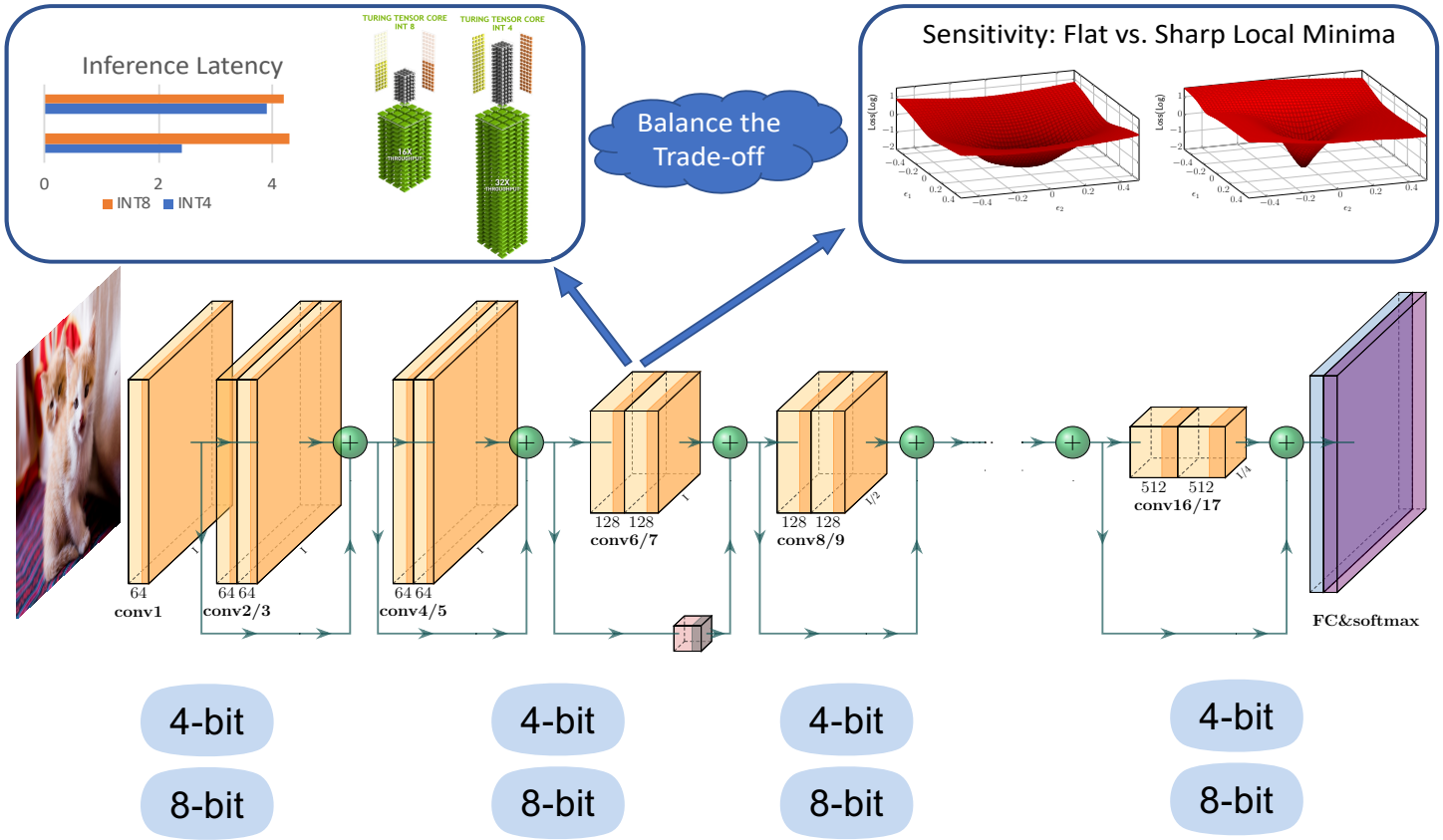
Using Hessian to Guide Choice of Bit Precision Layer by Layer





This picture misses the hardware. How can we find the right trade-off for a given latency/power constraint for a given hardware platform?

Hardware and Hessian Aware 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, arxiv:2011.10680, 2020.

Optimal Trade-Off with Integer Linear Programming

We find the best bit precision configuration such that:

- Minimally perturbs the model
- Meets application specific:
  - Model size constraint
  - Total bit operations for inference
  - Inference Latency

$$\text{Objective: } \min_{\{b_i\}_{i=1}^{L}} \sum_{i=1}^{L} \Omega_i^{(b_i)},$$

$$\text{Subject to: } \sum_{i=1}^{L} M_i^{(b_i)} \leq \text{ Model Size Limit},$$

$$\sum_{i=1}^{L} G_i^{(b_i)} \leq \text{ BOPS Limit},$$

$$\sum_{i=1}^{L} Q_i^{(b_i)} \leq \text{ Latency Limit}.$$

Results: ResNet18

(a) *ResNet18*

Method	IntOnly	Uniform	Open Source	Precision	Size (MB)	BOPS (G)	Top-1
Baseline	✗	–	✓	W32A32	44.6	1858	71.47
RVQuant [39]	✗	✗	✗	W8A8	11.1	116	70.01
HAWQV3	✓	✓	✓	W8A8	11.1	116	**71.56**
PACT [12]	✗	✓	✗	W5A5	7.2	50	69.80
LQ-Nets [54]	✗	✗	✓	W4A32	5.8	225	70.00
HAWQV3	✓	✓	✓	W4/8A4/8	6.7	72	70.22
HAWQV3+DIST	✓	✓	✓	W4/8A4/8	6.7	72	**70.38**
CalibTIB[26]	✗	✓	✓	W4A4	5.8	34	67.50
HAWQV3	✓	✓	✓	W4A4	5.8	34	**68.45**

Results: ResNet50

(b) *ResNet50*

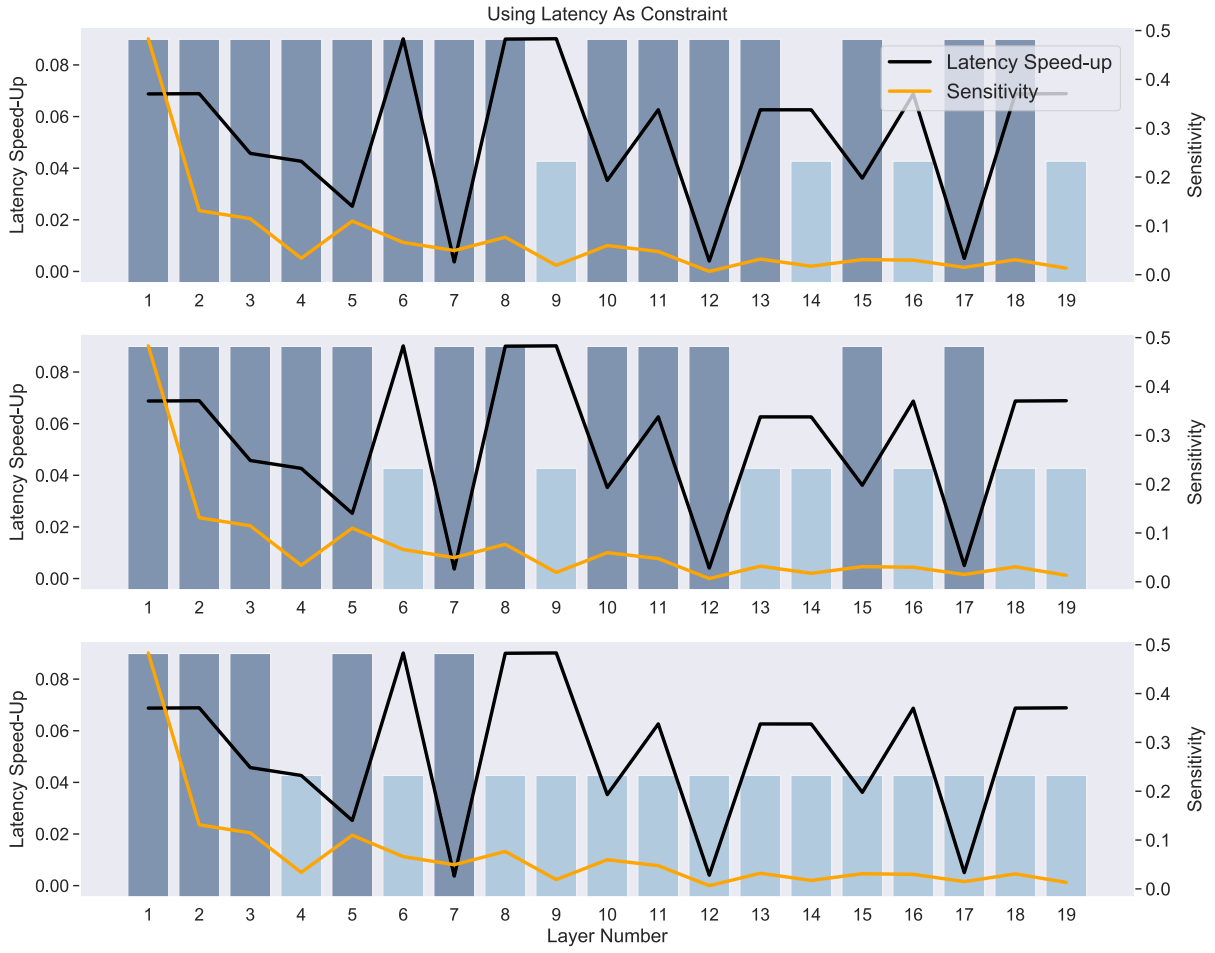
Method	IntOnly	Uniform	Open Source	Bit Precision	Size (MB)	BOPS (G)	Top-1
Baseline	✗	–	✓	W32A32	97.8	3951	77.72
Integer Only [29]	✓	✓	✗	W8A8	24.5	247	74.90
RVQuant [39]	✗	✗	✗	W8A8	24.5	247	75.67
HAWQV3	✓	✓	✓	W8A8	24.5	247	**77.58**
PACT [12]	✗	✓	✗	W5A5	16.0	101	76.70
LQ-Nets [54]	✗	✗	✓	W4A32	13.1	486	76.40
RVQuant [39]	✗	✗	✗	W5A5	16.0	101	75.60
HAQ [48]	✗	✗	✓	WMPA32	9.62	520	75.48
OneBitwidth [11]	✗	✓	✗	W1*A8	12.3	494	76.70
HAWQV3	✓	✓	✓	W4/8A4/8	18.7	154	75.39
HAWQV3+DIST	✓	✓	✓	W4/8A4/8	18.7	154	**76.73**

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, arxiv:2011.10680, 2020.
https://github.com/zhen-dong/hawq

How Does ILP Find the Trade-Off?

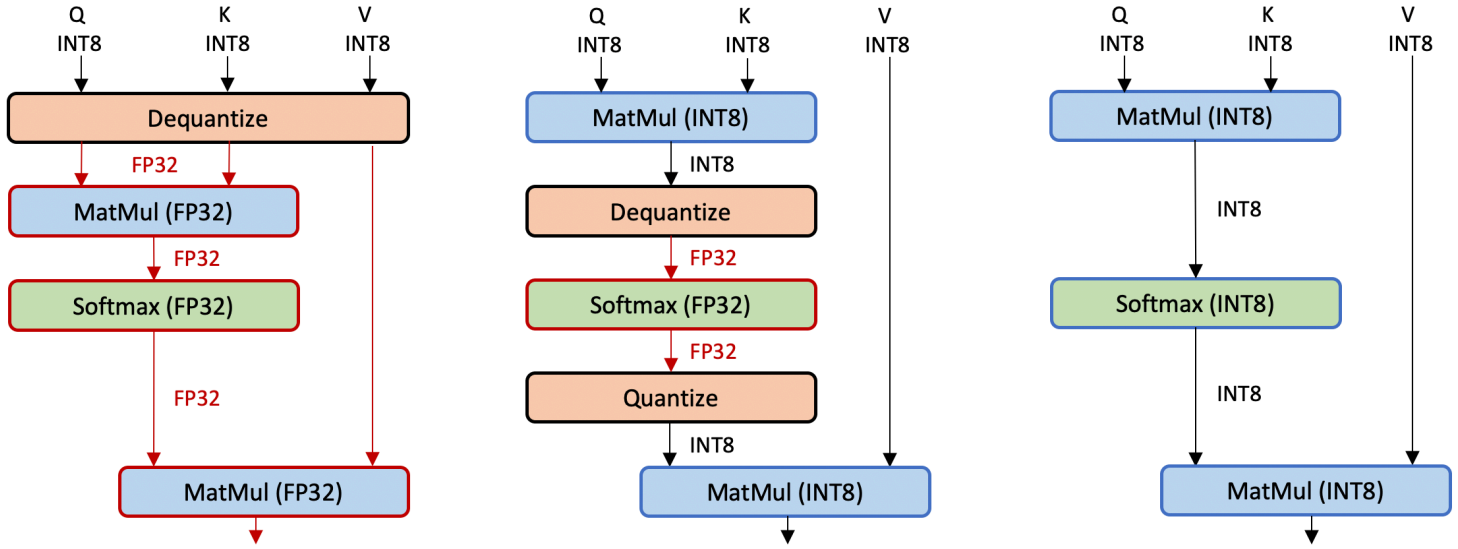
(a) *ResNet18*

	Level	Size (MB)	BOPS (G)	Speed	Top-1
INT8	–	11.2	114	1x	71.56
Latency	High	8.7	92	**1.12x**	70.40/71.05
	Medium	7.2	76	**1.19x**	70.34/70.55
	Low	6.1	54	**1.35x**	68.56/69.72
INT4	–	5.6	28	1.48x	68.45



Does this only work for CV tasks? Integer-only BERT

- We can also use integer-only quantization for BERT
  - Similar to vision, current BERT quantization uses fake quantization
  - Parameters are stored as integer, However operations are processed in floating point



Simulated Quantization (Q-BERT) **Simulated Quantization (Q8BERT)** **Integer-only Quantization (I-BERT)**

Shen S, Dong Z, Ye J, Ma L, Yao Z, Gholami A, Mahoney MW, Keutzer K. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. In AAAI 2020.
Zafrir O, Boudoukh G, Izsak P, Wasserblat M. Q8bert: Quantized 8bit BERT. arXiv preprint arXiv:1910.06188. 2019.
S. Kim, A. Gholami, Z. Yao, M.Mahoney, K. Keutzer, **I-BERT: Integer-only BERT Quantization**, work in progress.

Integer-only BERT

- Dyadic quantization works for 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

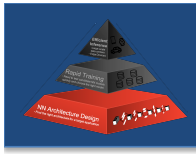
S. Kim*, A. Gholami*, Z. Yao*, M. Mahoney, K. Keutzer, **I-BERT: Integer-only BERT Quantization**, arxiv:2101.01321, 2021.

Review

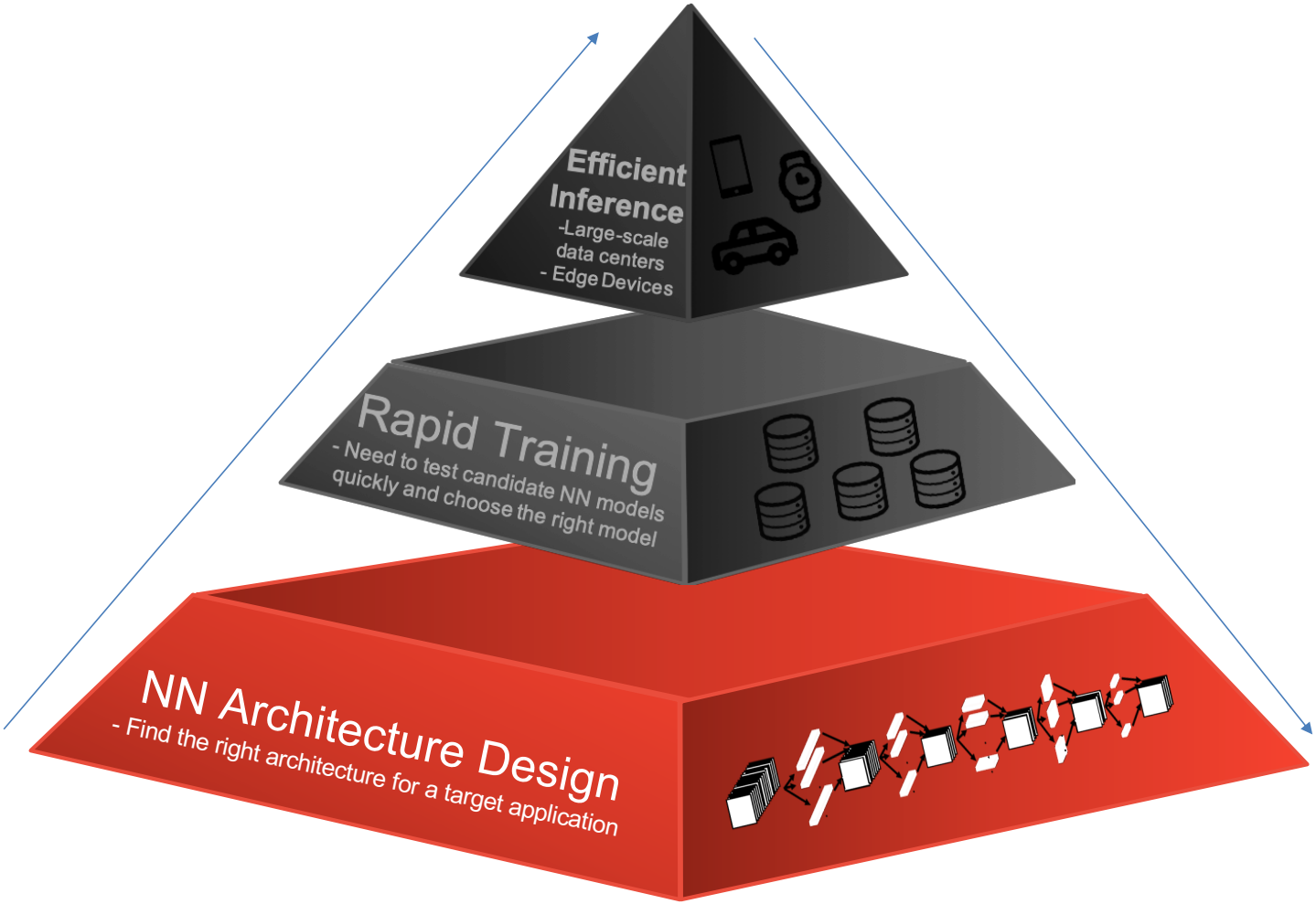
- Basic Concepts of Quantization
- **Advanced Concepts of Quantization**
  - Fake Quantization vs Integer-Only Quantization
  - Uniform vs Mixed-Precision Quantization
- Co-design of Neural Network and Hardware

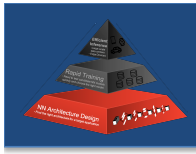
Outline

- Basic Concepts of Quantization
- Advanced Concepts of Quantization
- **Co-design of Neural Network and Hardware**



An Integrated Approach to DNN Design

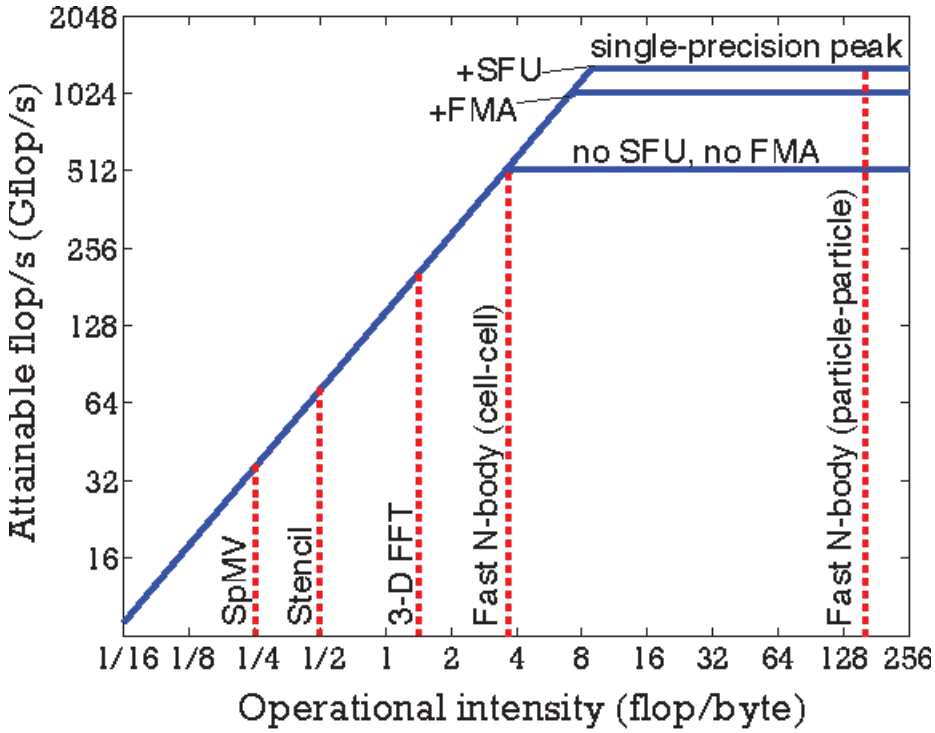




Arithmetic Intensity (AI)

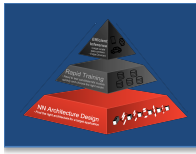
- #Params and MAC could be misleading
  - Ignores memory movement cost
- The right metric is **Arithmetic Intensity**

AI = FLOP / memory operations

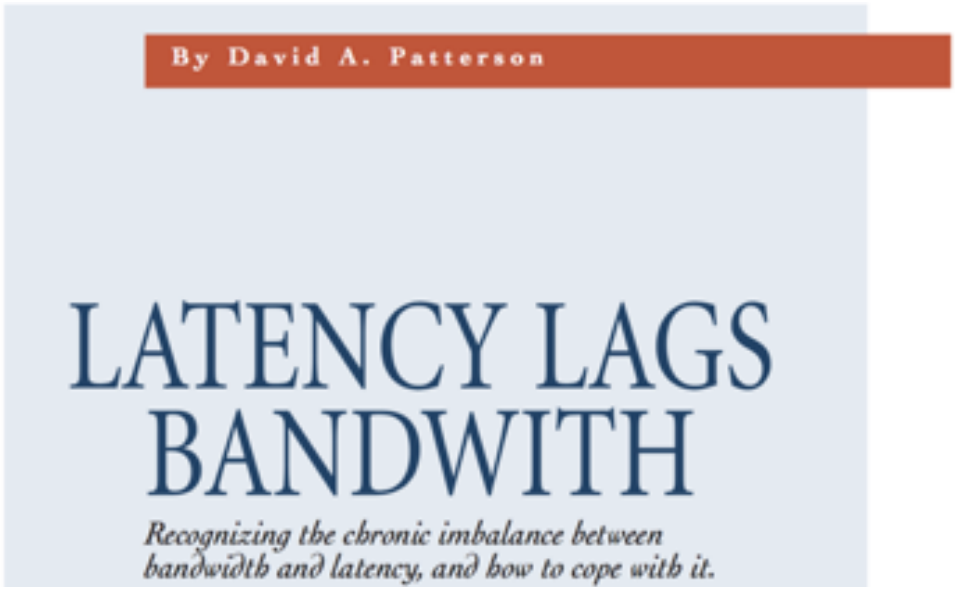


(Image credit Rio Yokota)

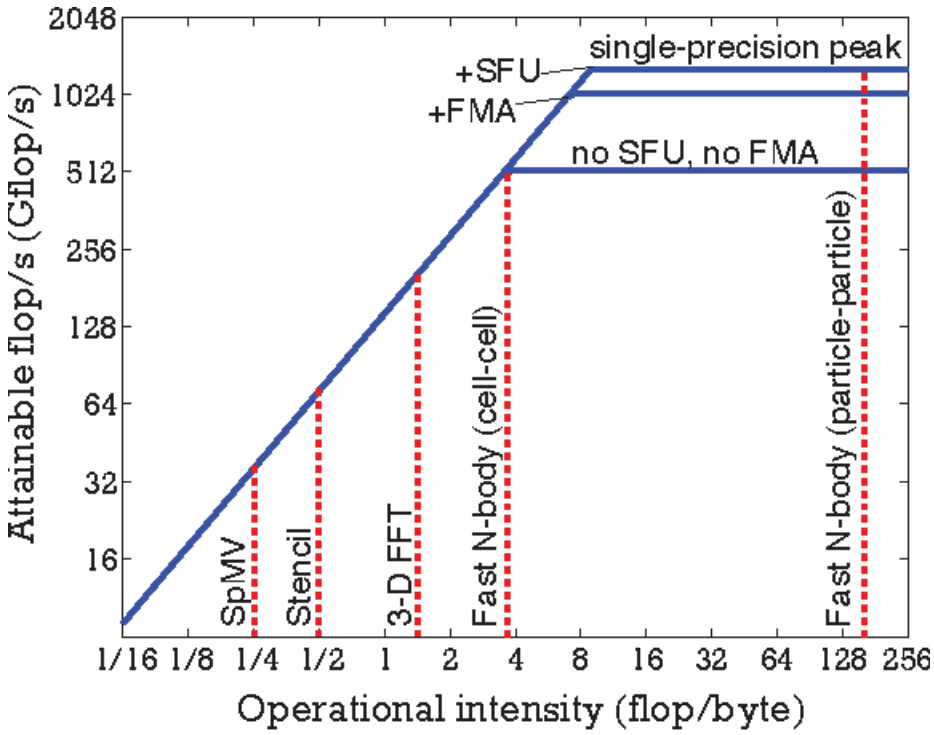
Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM. 2009 Apr 1;52(4):65-76.
Good video by Sam Williams: https://www.youtube.com/watch?v=hX8KjB3fJ3M



Arithmetic Intensity (AI)



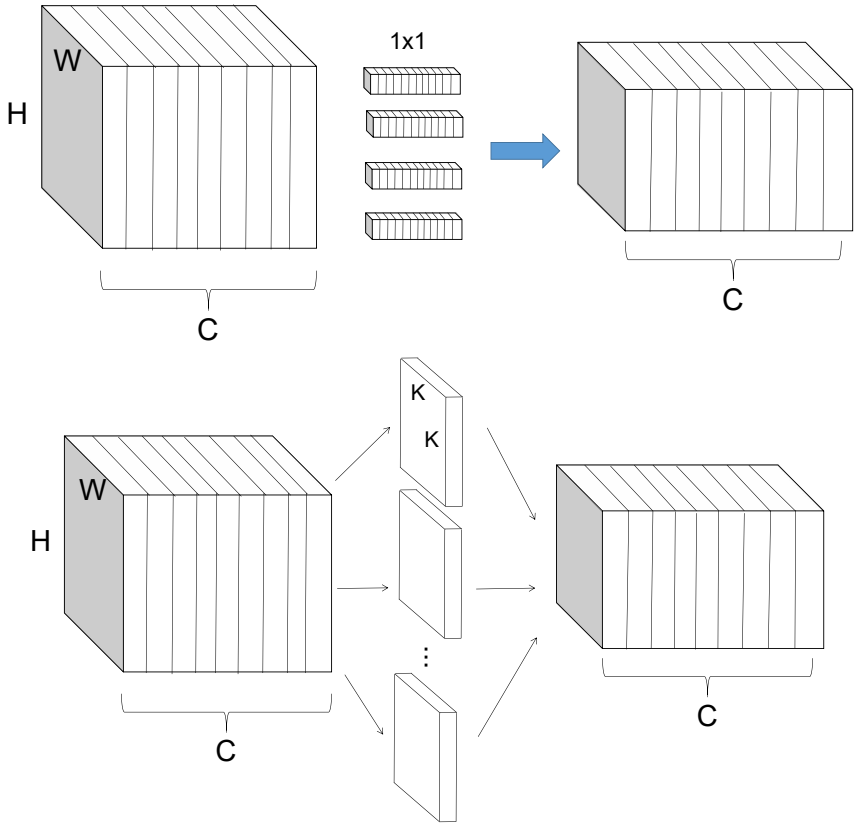
D. Patterson, CACM October 2004

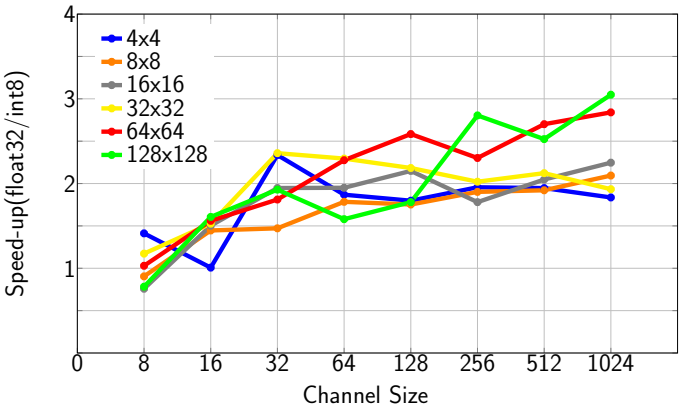


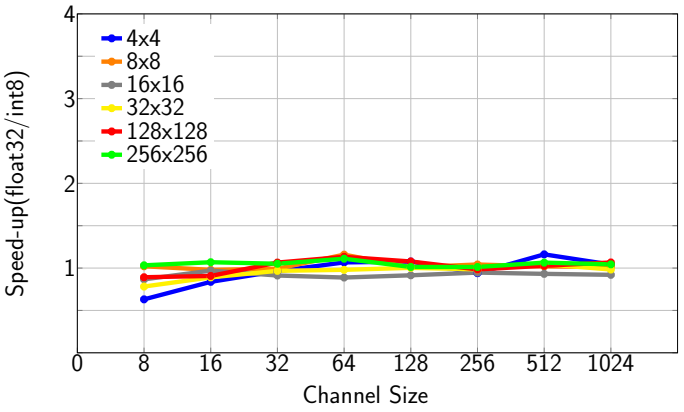
(Image credit Rio Yokota)

Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM. 2009 Apr 1;52(4):65-76.
Good video by Sam Williams: https://www.youtube.com/watch?v=hX8KjB3fJ3M

Not All Layers Have the Same Arithmetic Intensity



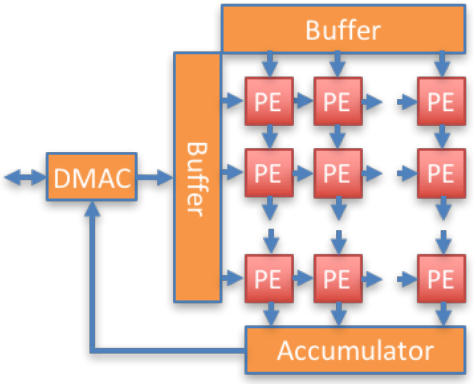


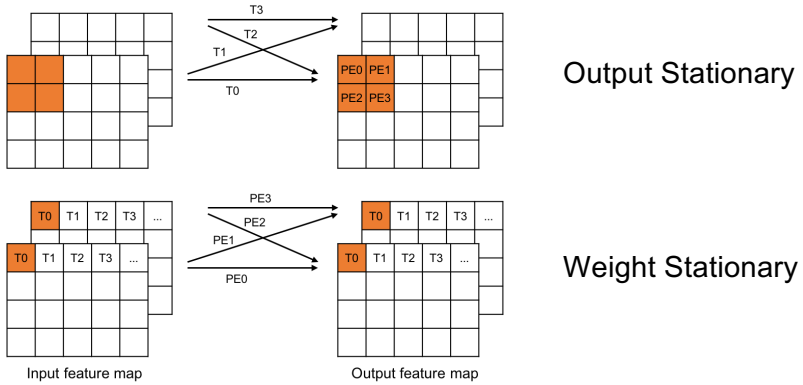


SqueezeNext: Hardware Aware Neural Network Design

- In collaboration with Samsung, we were able to get HW metrics for each layer so that we can also change the NN architecture so that it would run optimally on the HW

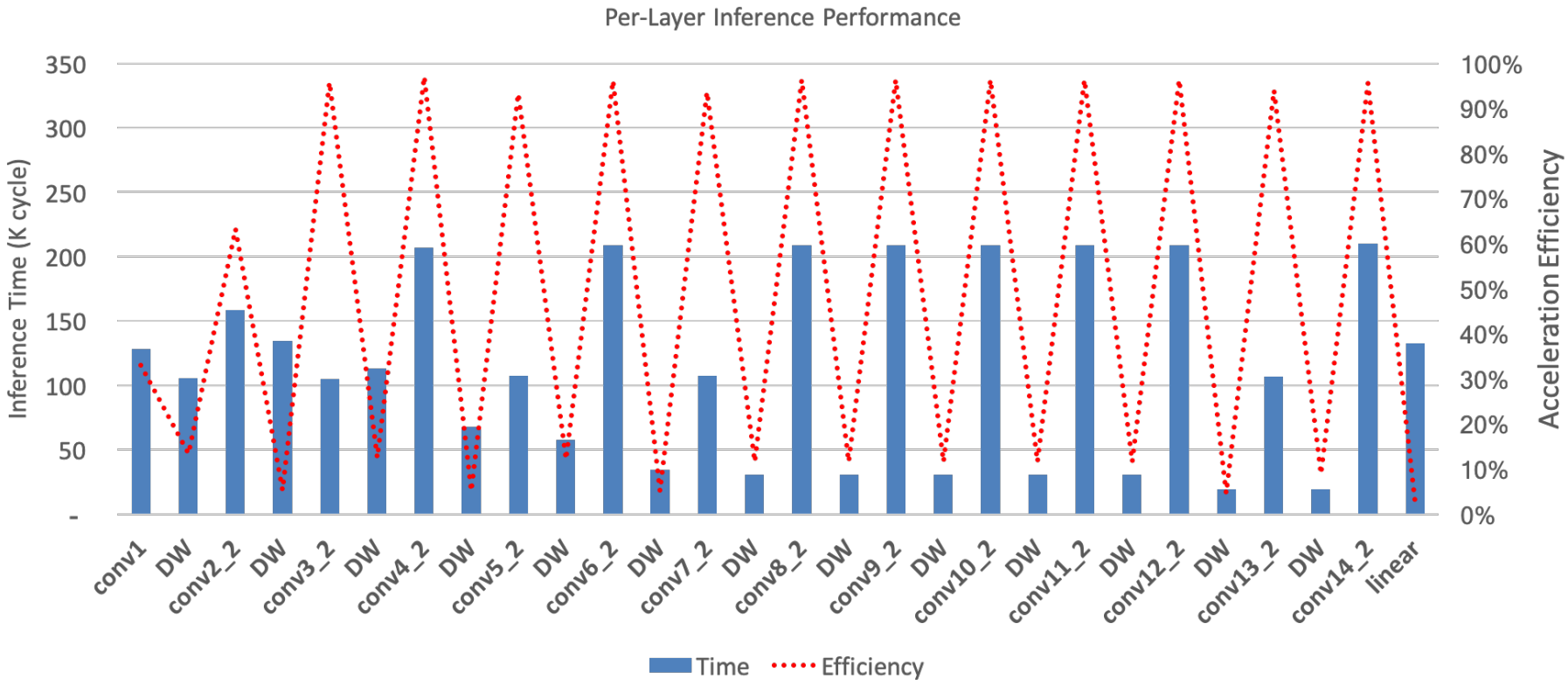
  => **Hardware-aware NN Design**





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.

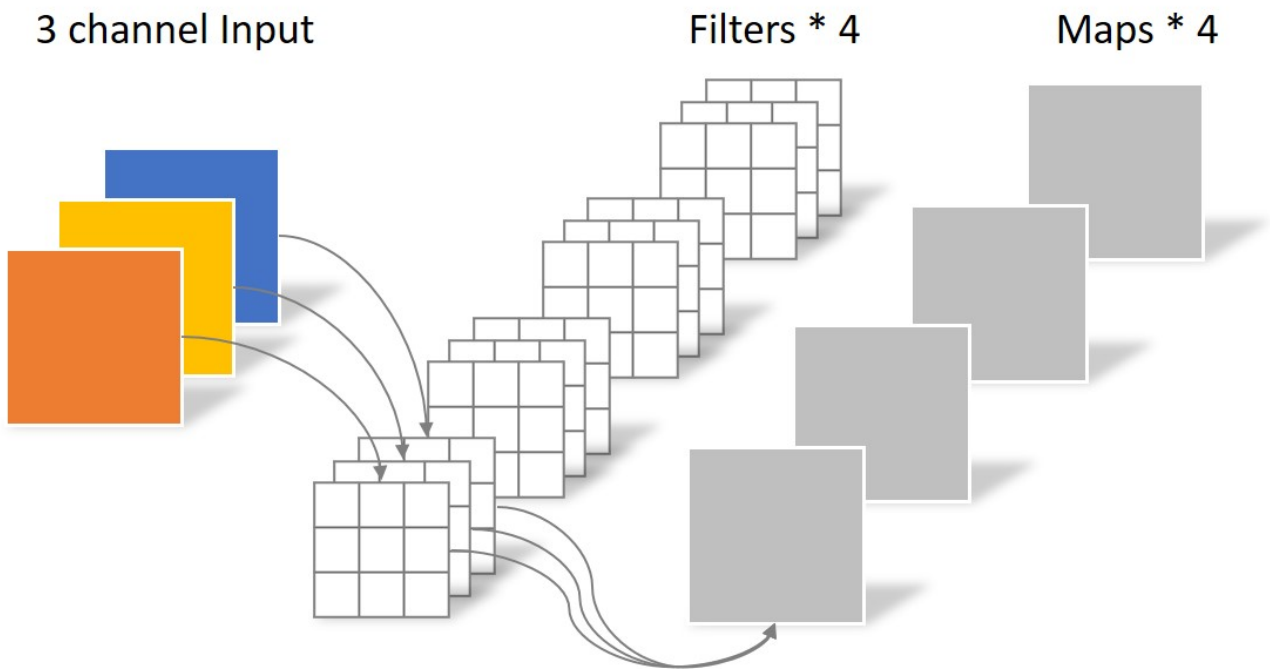
MobileNet



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.

DepthWise Separable Convolution

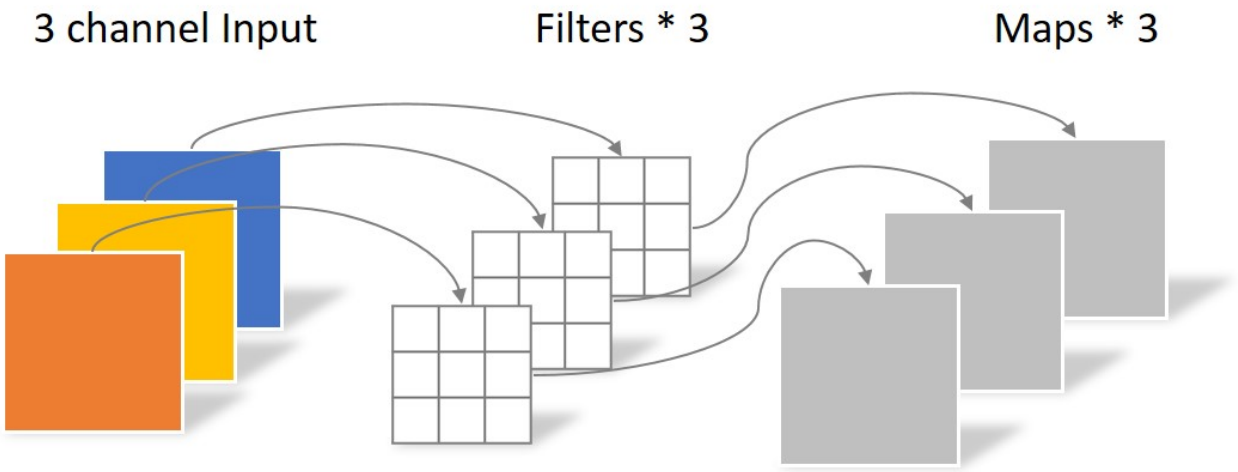
- Dense 3x3 convolution



(Image credit Yin Guobing)

DepthWise Separable Convolution

- DepthWise (DW) 3x3 convolution
  - **AI reduced by a factor of 3**

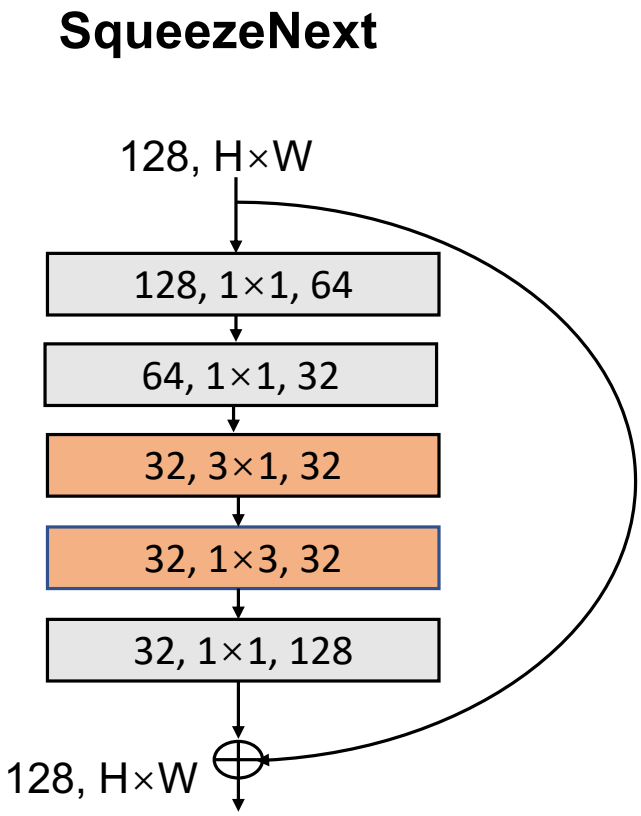


(Image credit Yin Guobing)

SqueezeNext Micro-Architecture

Idea:

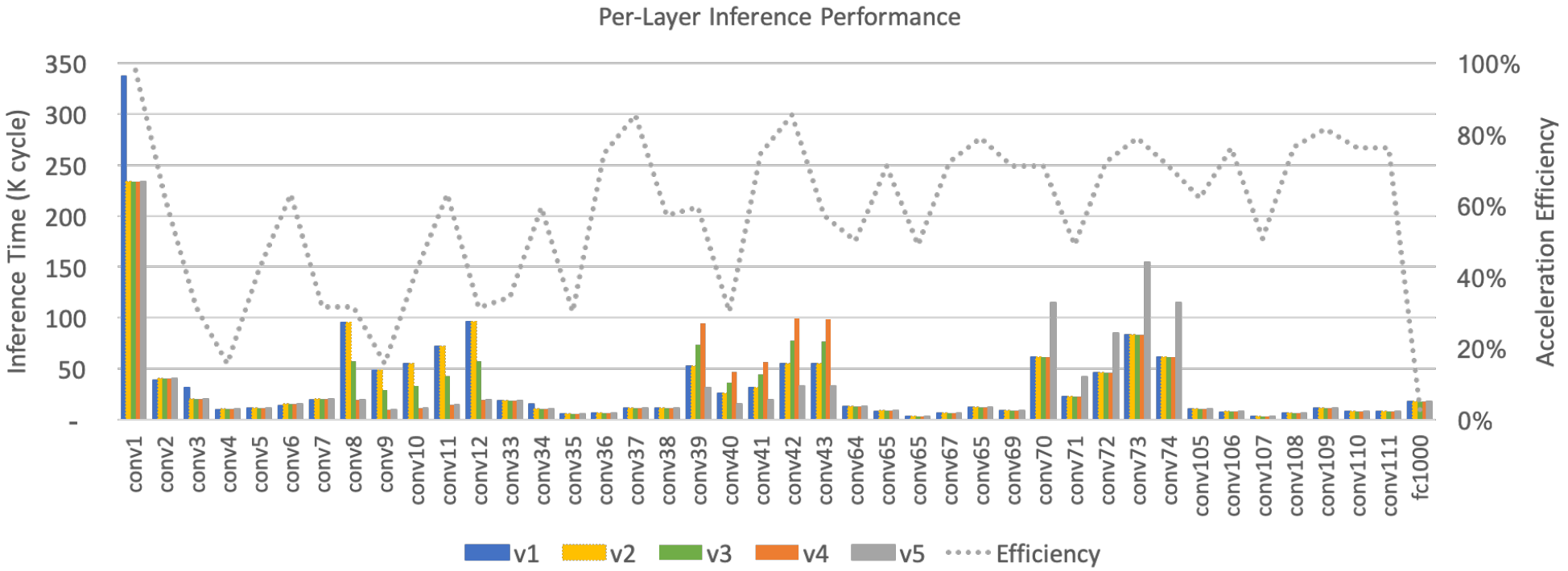
- Instead of using DW Conv, let's use Spatial Separable Convs
- **Manually** adapt the NN architecture based on HW metrics



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.

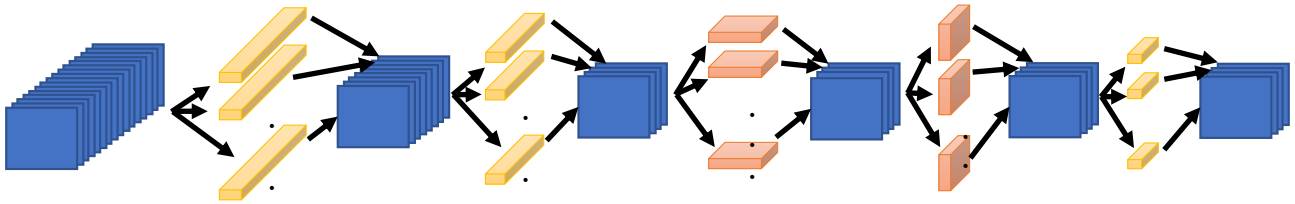
Per layer inference utilization

- Some layers have poor hardware utilization
  - Redesign the macro-architecture based on hardware utilization



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.

SqueezeNext Accuracy with Squeezelator Optimization

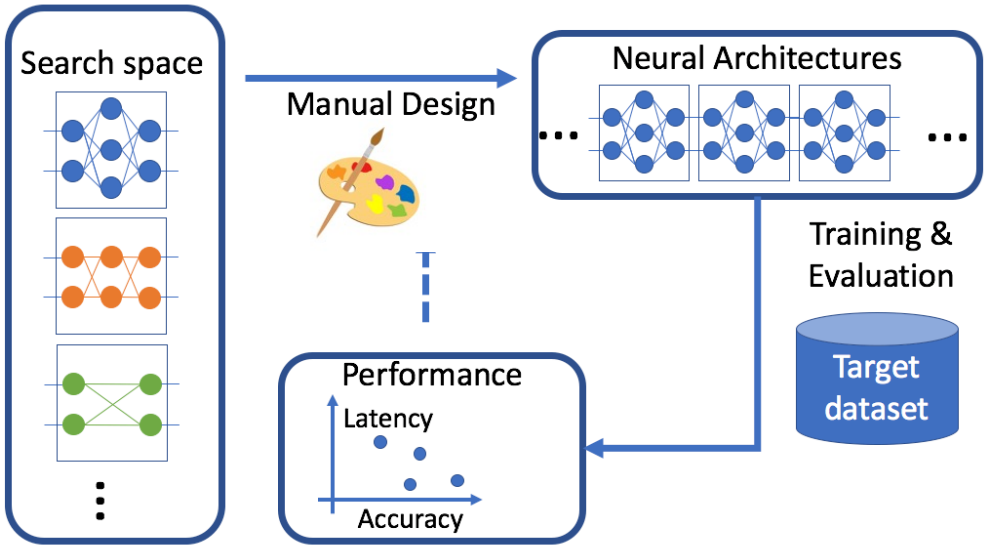


Model	Top-1	Top-5	Params
AlexNet	**57.10**	**80.30**	**60.9M**
SqueezeNet	57.50	80.30	1.2M
1.0-SqNxt-23	59.05	82.60	0.72M
1.0-G-SqNxt-23	**57.16**	**80.82**	**0.54M**
MobileNet	67.50(70.9)	86.59(89.9)	**4.2M**
2.0-Sqxt-23v5	**67.44(69.8)**	**88.20(89.5)**	**3.2M**

- Matches AlexNet with **112x** smaller parameters
- Deeper version achieves VGG accuracy with **36x** smaller model
- Exceeds MobileNet's top-5 by **+1.6%** with **1.3x** fewer parameters

But Manual Design is not Scalable

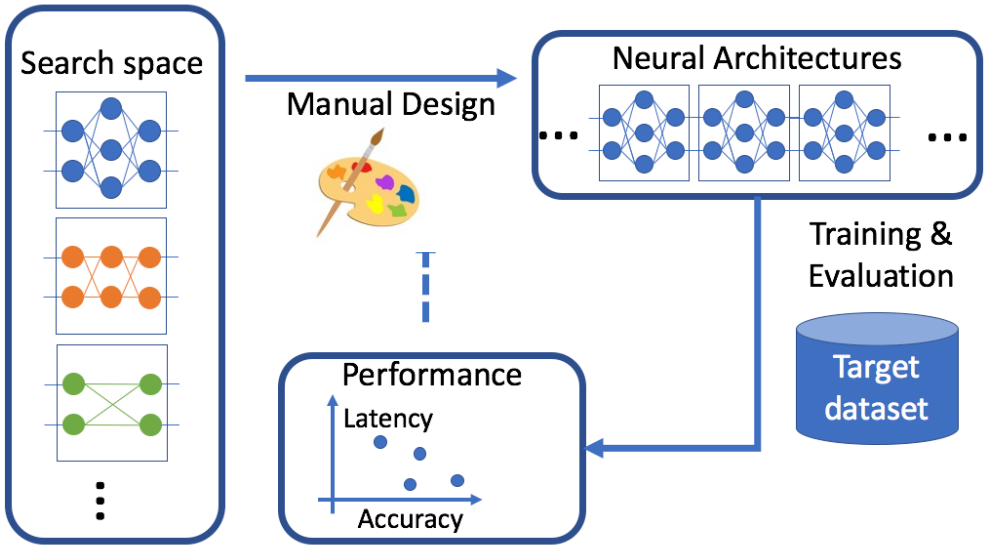
- Manual design:
  - Each iteration to evaluate a point in the design space is very expensive
  - Exploration limited by human imagination



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

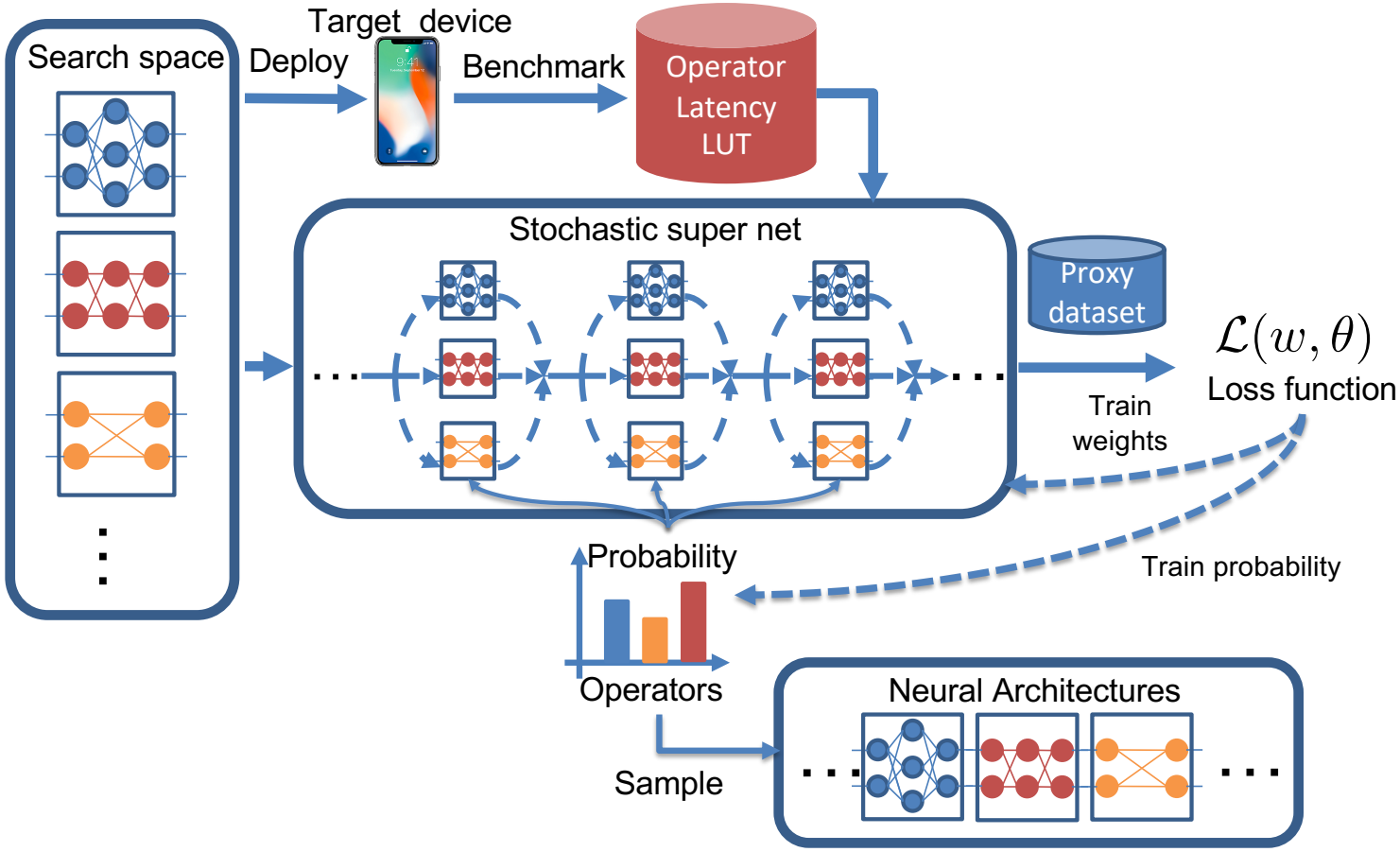
Can we automate this?

- Manual design:
  - Each iteration to evaluate a point in the design space is very expensive
  - Exploration limited by human imagination



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

Differentiable Neural Architecture Search



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

Design Study 1:
A11 vs Snapdragon 835

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

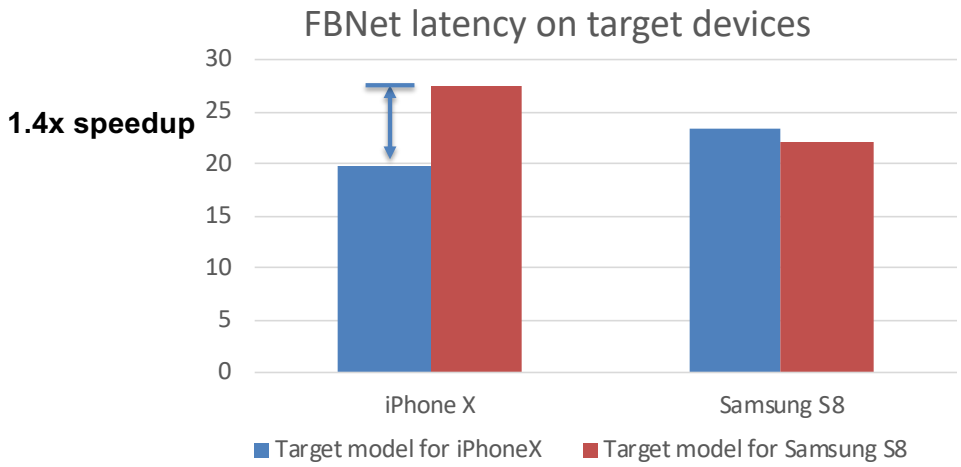
Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. CVPR 2019.
Slide: Courtesy of Bichen Wu, and Kurt Keutzer

Result: FBNet for different target devices

- 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

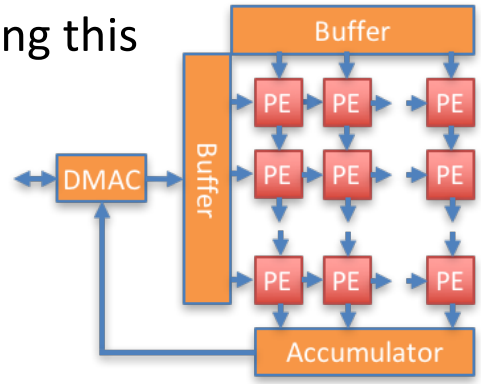
- Under similar accuracy constraint (73.27% vs 73.20%), FBNet optimized for iPhone-X achieves 1.4x speedup over the Samsung optimized model

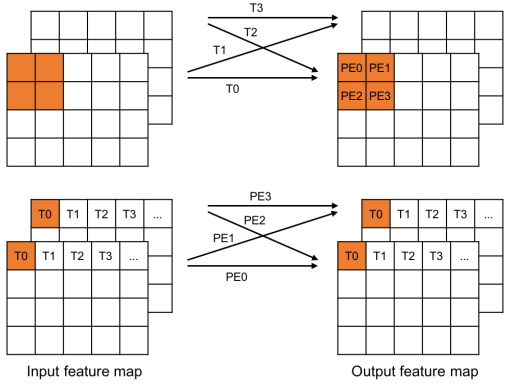


Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. CVPR 2019.
Slide: Courtesy of Bichen Wu, and Kurt Keutzer

Can We Co-Design the Hardware as Well?

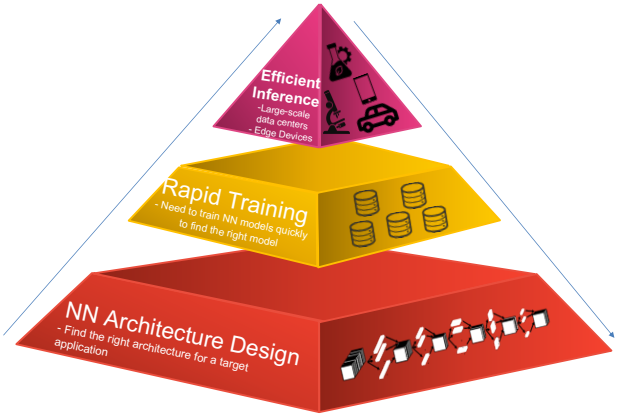
- An interesting next step would be to couple Neural Architecture Search with hardware design
- It may be possible to then perform a large scale search to find an architecture that has good performance for a range of tasks
- This requires a fast, and accurate hardware simulator
  - TimeLoop is a recent work enabling this

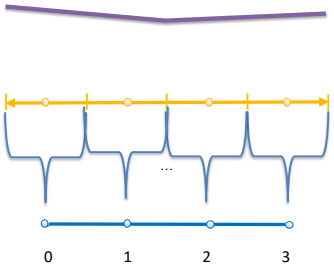




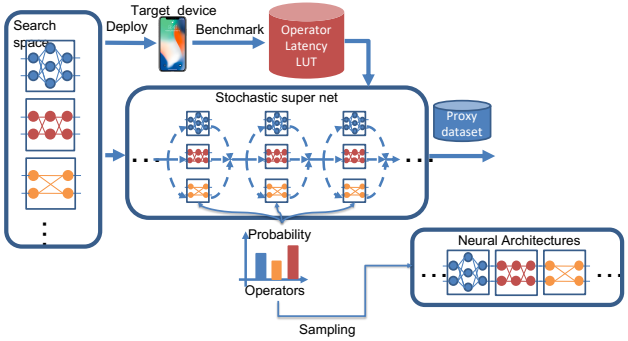
Parashar A, Raina P, Shao YS, Chen YH, Ying VA, Mukkara A, Venkatesan R, Khailany B, Keckler SW, Emer J. Timeloop: A systematic approach to DNN accelerator evaluation. In2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) 2019 Mar 24 (pp. 304-315).

Summary: Three Elements of Efficiency at the Edge



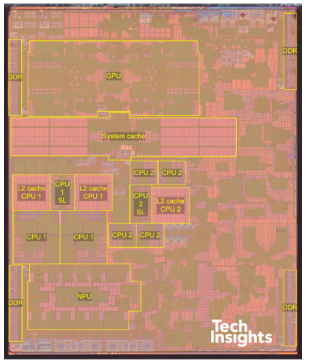


New DNN Models



Optimizations:
Pruning
Quantization
Distillation

New Processors And DNN Accelerators



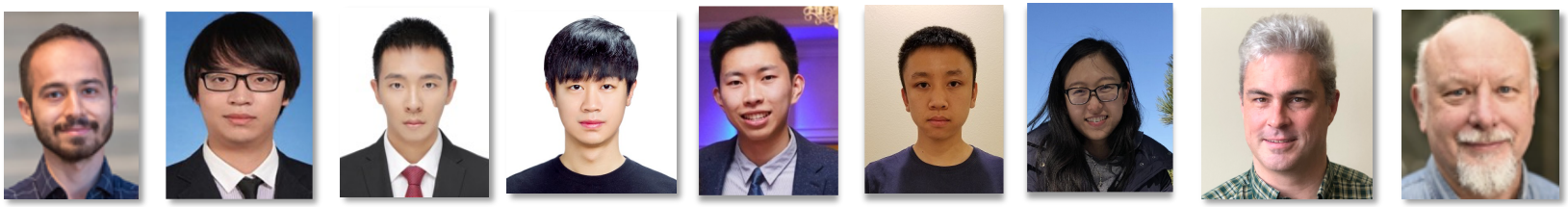
Summary

- Not all layers have the same hardware utilization
  - Pointwise convolution typically achieves high utilization
  - DepthWise covolution typically achieves very low utilization
- 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 of NN and hardware
  - SqueezeNext was an early work followed by automated DNAS but much more work is left to do

Thanks for Listening

For any feedback/questions please contact

amirgh@berkeley.edu



Berkeley
UNIVERSITY OF CALIFORNIA