Abandoning the Dark Arts: Scientific Approaches to Efficient Deep Learning

Kurt Keutzer (EECS, UC Berkeley)
with PhD students and post-docs
Alon Amid, Zhen Dong, **Amir Gholami**, Qijing Huang, Suresh Krishna**, Bichen Wu**,
Zhewei Yao, Yang You, Xiangyu Yue, Tianjun Zhang, **Sichen Zhao**

and an army of undergraduate and MS-level researchers
Daiyaan Arfeen, Yaohui Cai, Ravi Krishna,, Aniruddha Nrusimha, Sheng Shen, Bernie Wang,
Yifan Yang (Tsinghua), Xuanyu Zhou and many others

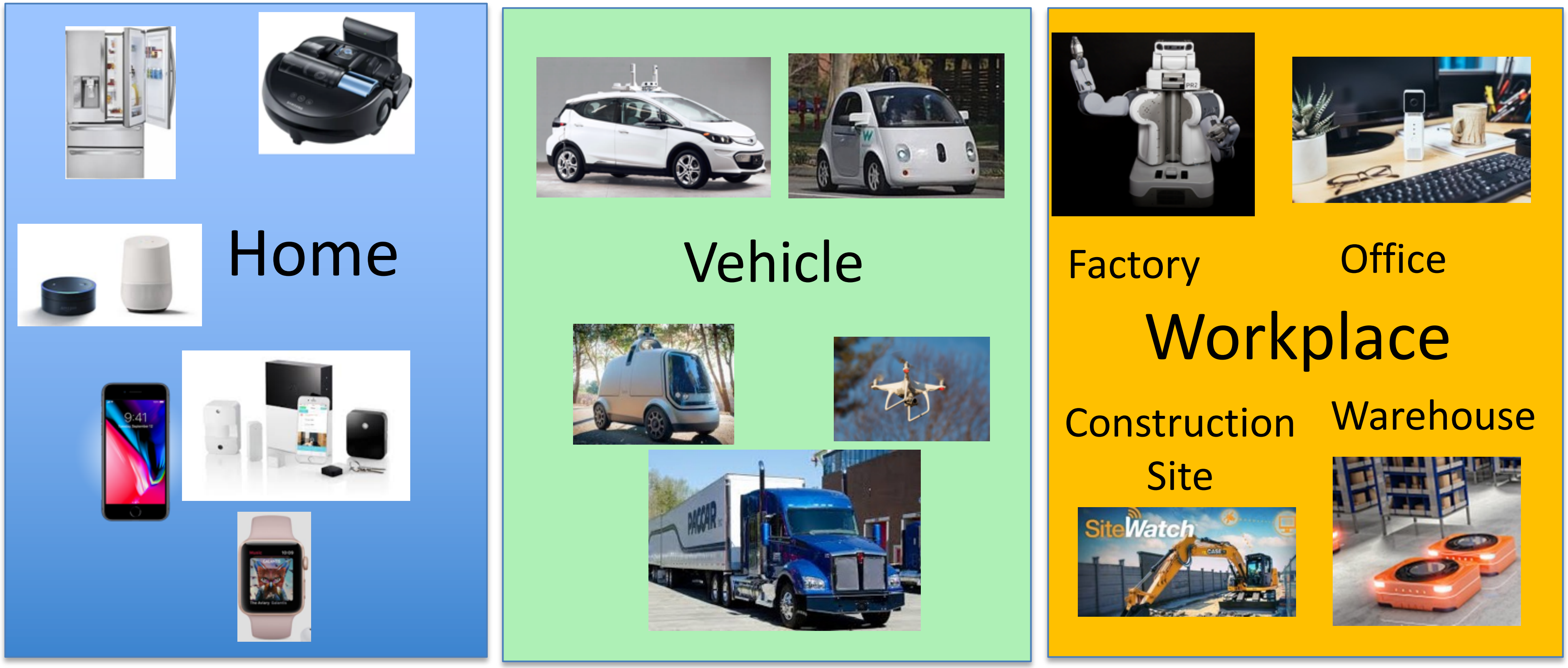
with fellow faculty, Joey Gonzalez, Mike Mahoney, Krste Asanovic, Jim Demmel and Sanjit Seshia, Alberto S-V

as well as Peter Vajda (and others) at Facebook, Kiseok Kwon at Samsung,
Forrest Iandola, Albert Shaw & Others (DeepScale → Tesla),
Michaela Blott and Kees Vissers (and others) at Xilinx, and Liang Ma and Luciano Lavagno at P de Torino

Berkeley DeepDrive

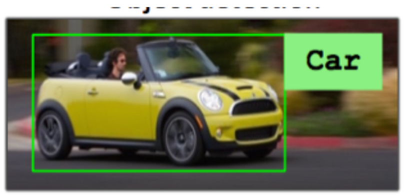
DNNs Will Bring Intelligence to the Edge

BAIR



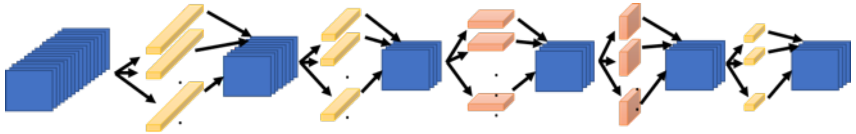
An Integrated Approach to DNN Design Has Four Key Aspects

Rapidly Training the DNN



Aggregating training data

Finding the right Deep Neural Network model



Efficiently implementing the DNN on embedded HW / co-design DNN accelerators

We've Gotten a Perspective

Rapidly Training the DNN

FireCaffe
- CVPR 2016

LARS, LAMB
- ICPP 2018 – Best Paper Award

Hessian Aware Methods
- NeurIPS'18

ICRA 2018
ICRA 2019
ICMR 2019
ISTC 2019
ICCV 2019
NeurIPs 2019

Aggregating training data

Finding the right Deep Neural Network model

SqueezeNet (2016): 1606 citations
SqueezeDet
- ECV/CVPR 2017– Best Paper Award

SqueezeNext
- ECV/CVPR 2018

ShiftNet
- CVRP spotlight 2018

SqueezeSeg v1, v2
- ICRA 2018, 2019

DiracDeltaNet
- FPGA 2019

FBNet
- CVPR Oral 2019

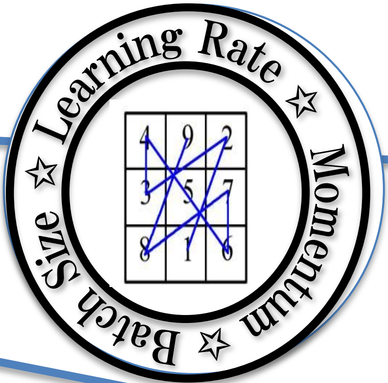
Efficiently implementing the DNN on embedded HW / co-design DNN accelerators

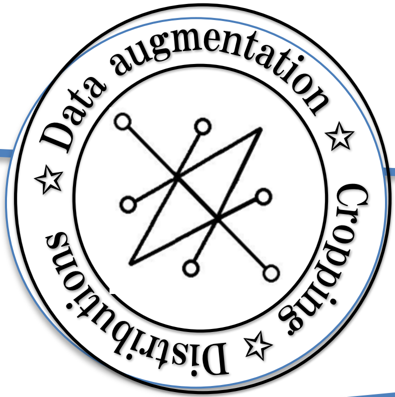
HAWQ: ICCV2019
AAAI 2020
Squeezelerator
- DAC 2018
- IBM J. Res. Dev. 2019

Synetgy
- FPGA 2019

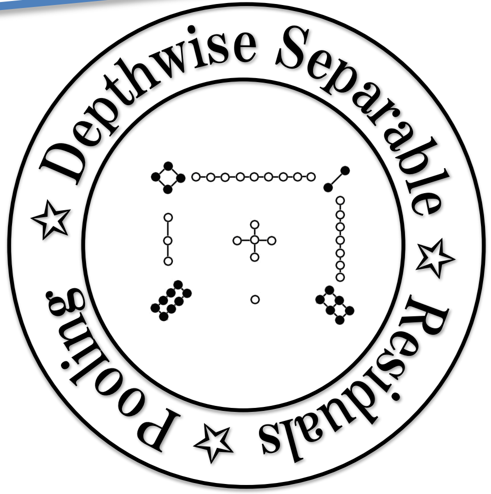
An Integrated Approach to DNN Design Has Four Key Aspects

Training
- Momentum,
- Learning rate
- Batch size

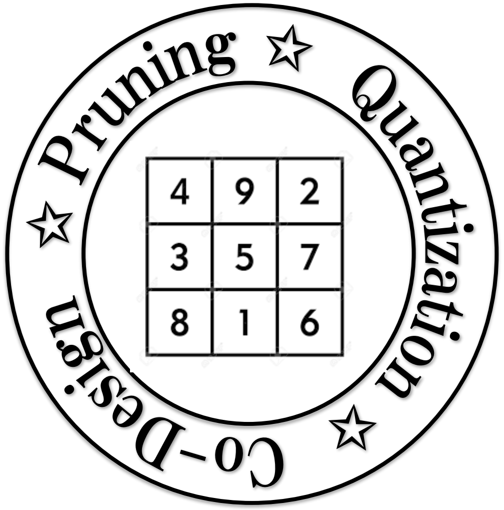




Training Data
- Data Augmentation
- Cropping
- Data distribution



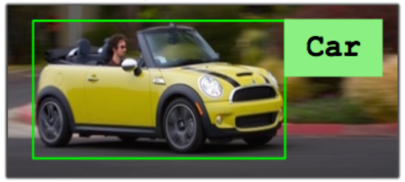
DNN Design
- # Layers and types
- Residuals



Efficient Implementation
- Pruning
- Quantization

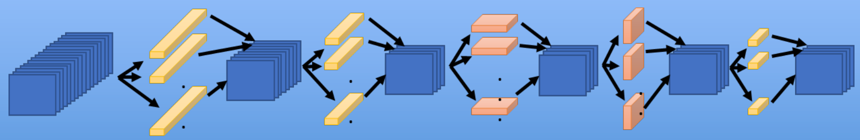
Deep Neural Net Design, Training, and Implementation

Rapidly Training the DNN



Aggregating training data

Finding the right Deep Neural Network model

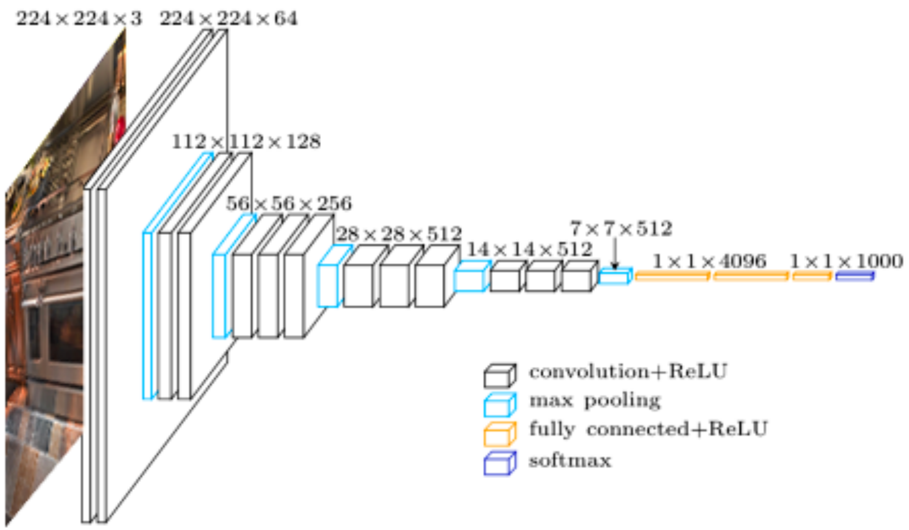


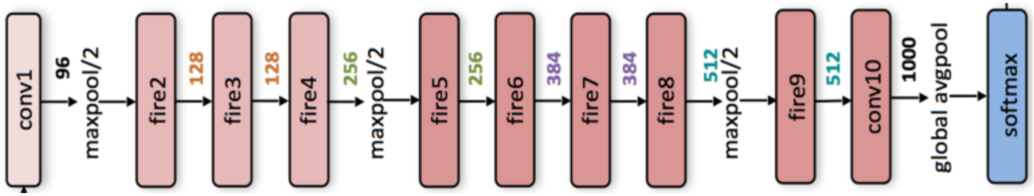
Efficiently implementing the DNN on embedded HW / co-design DNN accelerators

Berkeley DeepDrive

Challenge #0: DNNs are hard to understand

BAIR

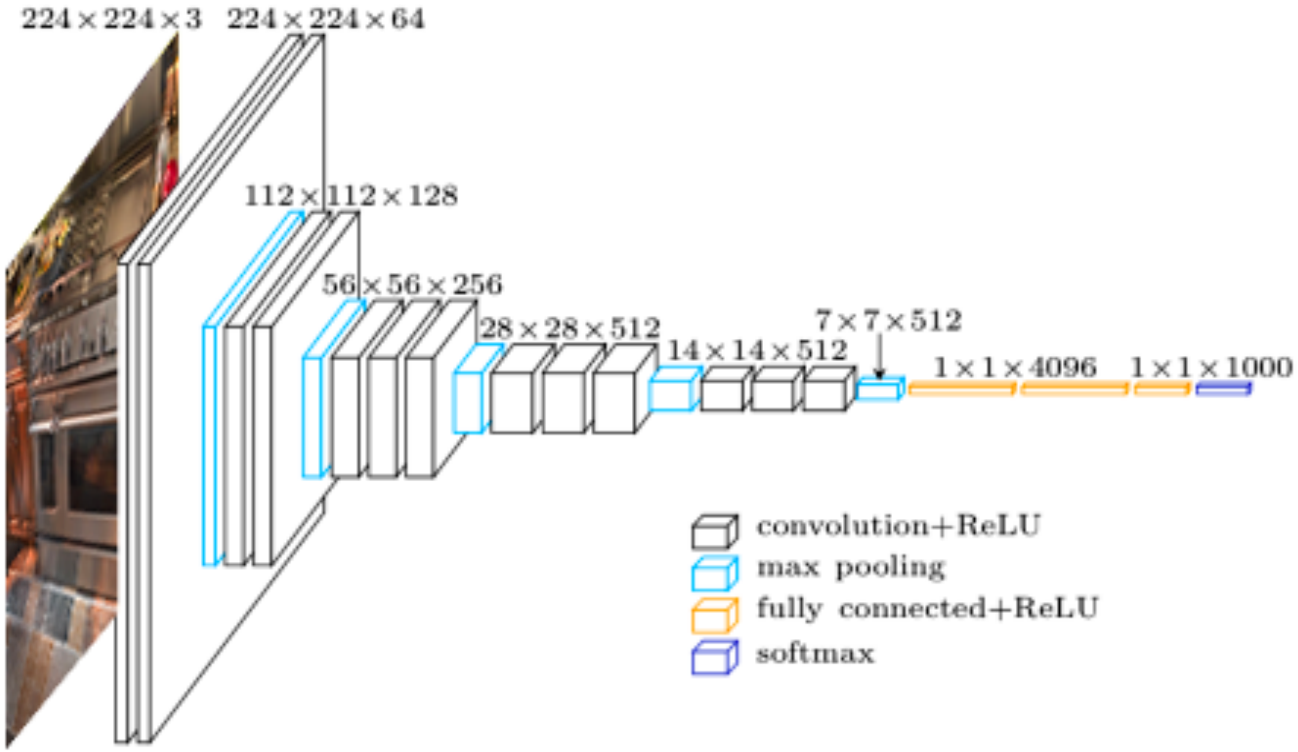




- Deep Neural Nets are a somewhat counter-intuitive medium for expressing algorithmic ideas
- But that’s our fault not their fault

Challenge #1: Large space of alternatives (aka Design Space)

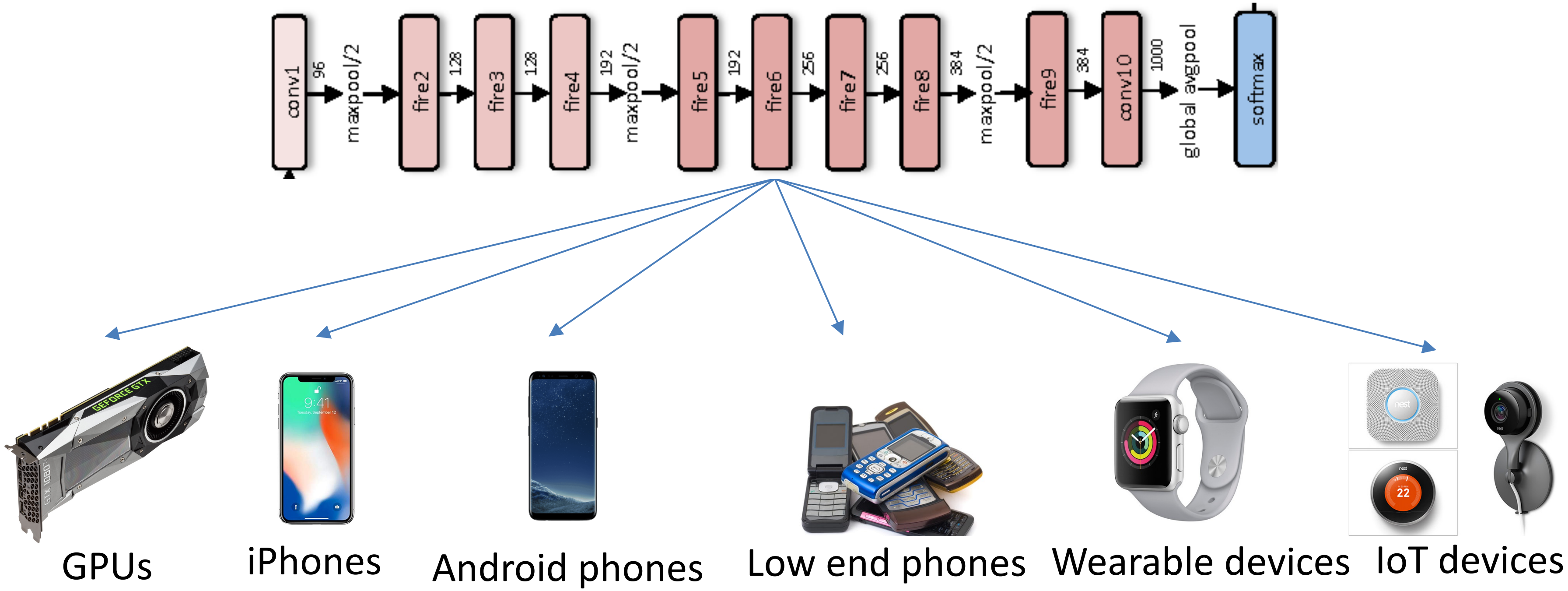
- Design space of Deep Neural Nets is huge!
  - Number of layers
  - Design choices for each layer:
    - Layer type
    - kernel size = {1, 3, 5}
    - channel size = {32, 64, 128, 256, 512}



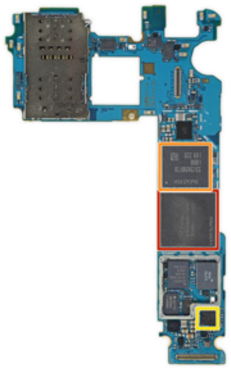
[1] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

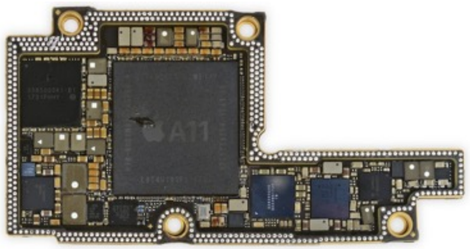
Challenge #2: Diverse Targets for DNN, Each is Different

- Ideally, we should design different Neural Networks to different devices/tasks/computation budgets
- Each one has different computational characteristics
  - Different latency/energy profile and trade-offs



Challenge #3a
Edge Applications Extra Bring Constraints





Power (Watts)

- Convenient and economical **packaging** limits how much power our mobile devices can dissipate
- ***2-5W*** max seems common among mobile handsets, up to 20W in other applications
- IP Blocks will have much stricter constraints

Energy (Joules) = power * time

- **Battery life** limits the total energy that our mobile devices can use
- iPhoneX battery ***10.35 WHours***

Applications may bring further constraints on accuracy and latency

https://www.macrumors.com/2017/11/03/iphone-x-teardown-ifixit/

Challenge #3b: Constraints are Tight at the Edge

Average Power Dissipation

Battery Life

11 W
2.4 Hours

2.5 W
8.4 Hours

1-1000 mW
8-240 Hours

15 inch Macbook Pro |
76Wh = 273.6kJ
Apple: 10h = 7.6 W

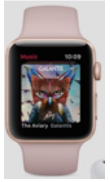
13 inch Macbook Air | 54Wh = 194.4kJ
Apple: 12h = 4.5 W

Eee PC 1000HE | 49Wh = 176kJ
Asus: 9.5h = 5.2 W

iPad Pro | 41Wh = 147kJ
Apple: 10h use = average 4.1W

Kindle Oasis | 0.91Wh = 3.276kJ
Ebook Friendly: “15days @ 30m/day” = 7.5h @ 0.12 W average

2017 iPhone8 | 6.96Wh = 25kJ
Talk time: 14h = 0.5 W
Video: 13h = 0.54 W



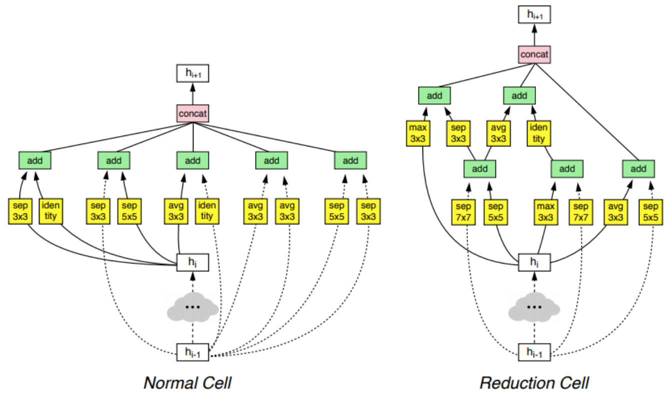
iwatch Series 3 1.07Wh | 3.8kJ
60mW, 18 hours

OZO Digital Pedometer
80µW, 0.72Wh | 1 year

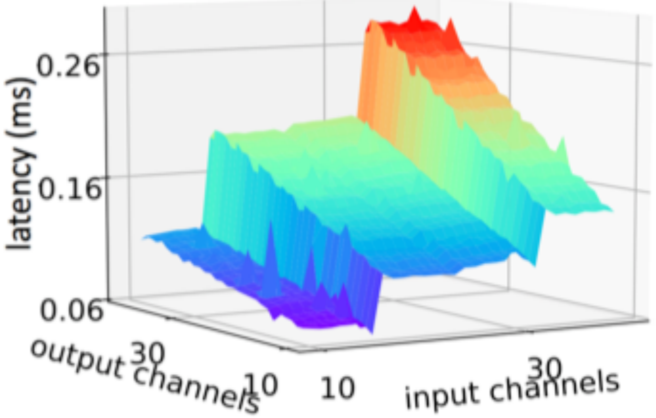
1 Wh = 3.6 kJ

Challenge #4: Hard to Accurately Measure Key Metrics

- Our primary concerns are (accuracy), latency, and energy
- What's easy to measure is flops and model parameters
- However, a lower FLOP count does not necessarily mean lower latency
  - NASNet-A has slightly smaller FLOPs than MobileNetV1, but the latency is 1.6x slower
  - SqueezeNet V1.0 50x smaller than AlexNet but slower on some targets



NASNet [1]

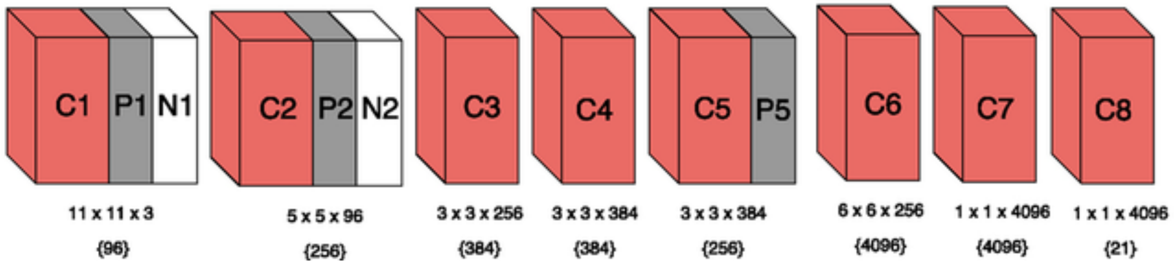


Input/output channels vs latency [2]

[1] Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." *arXiv preprint arXiv:1707.070122*.6 (2017).
[2] Dai, Xiaoliang, et al. "ChamNet: Towards Efficient Network Design through Platform-Aware Model Adaptation." *arXiv preprint arXiv:1812.08934* (2018).

SqueezeNet: A Child of the Dark Arts

AlexNet [1]



50x parameter reduction

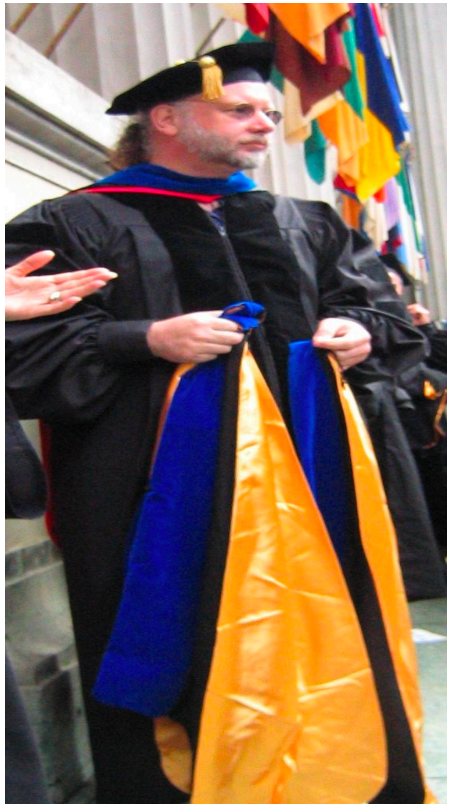
- 1000 neural architectures explored
- Average 32 GPUs x 12 hours per NN explored
- Cost on AWS: 1000 x 32 x 12 x \$0.9 ≈ \$350K

SqueezeNet [2]

CNN	Top-5 Accuracy ImageNet	Model Parameters	Model Size
AlexNet[1]	80.3%	60M	243MB
SqueezeNet[2]	80.3%	1.2M	4.8MB

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." NIPS2012
[2] Iandola, Forrest N., et al. "**SqueezeNet**: AlexNet-level accuracy with 50x fewer parameters and< 1MB model size." arXiv: 1602.07360 (2016). (February 2016)

Dark Art or Art?



Dark Art:

- Blindly invoking forces that you do not understand or truly control
- Unbridled parameter tuning
- Cf. Faust, Goethe, Marlowe,

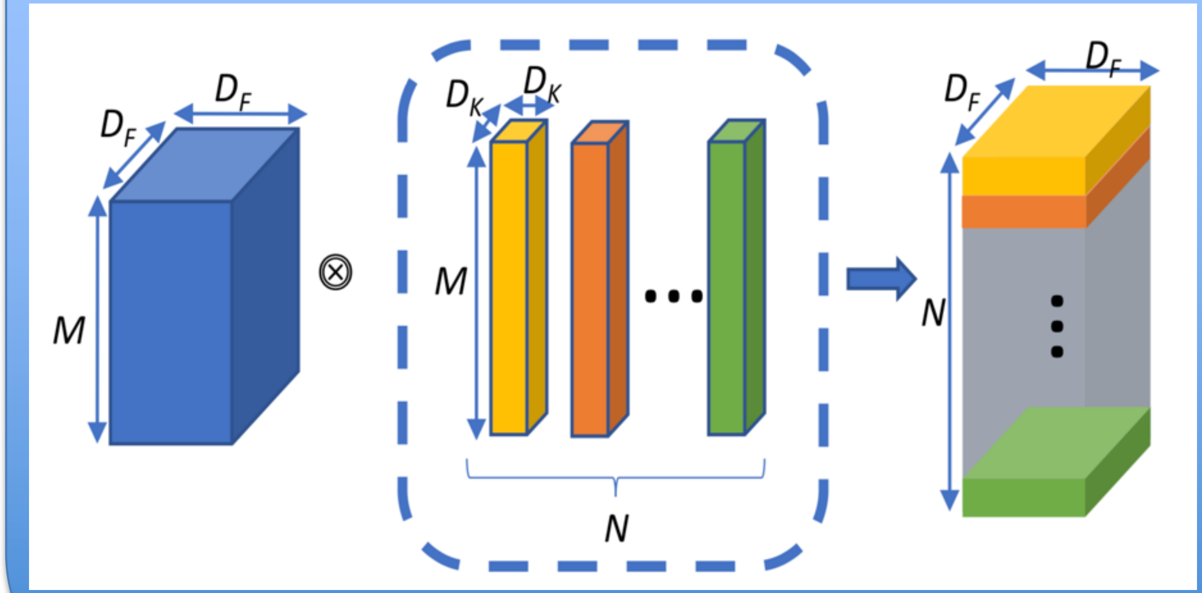
Art:

- Intelligent application of design principles

CNN Layer have different Computational Characteristics - 1

Normalized AI: 1.0

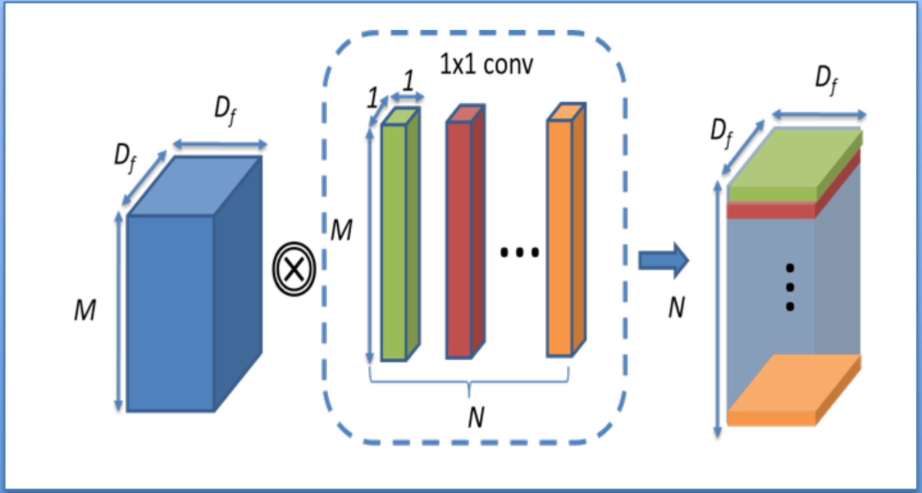
Spatial Convolution



$$\frac{MND_K^2D_F^2}{D_F^2(M+N)+D_K^2MN}$$

Normalized AI: 0.6

Pointwise (1x1) Convolution

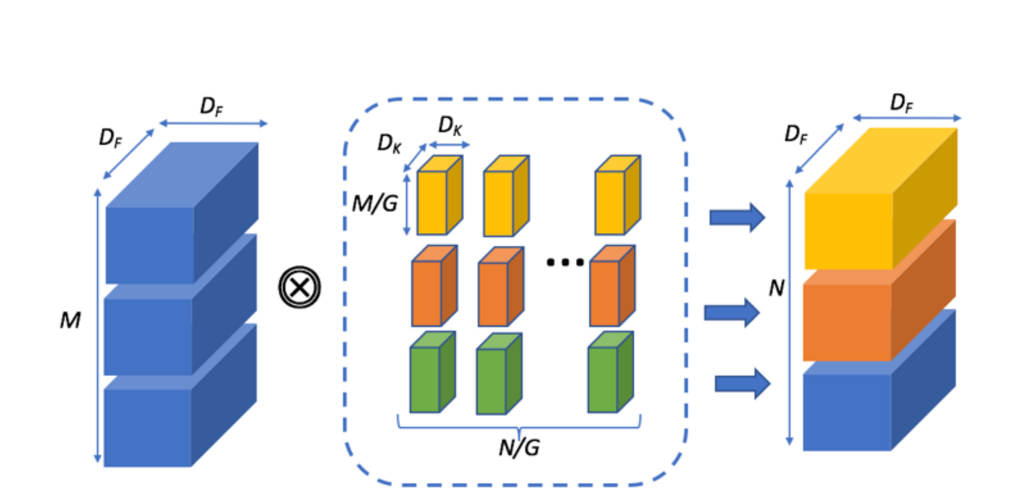


$$\frac{MND_F^2}{D_F^2(M+N)+MN}$$

CNN Layer have different Computational Characteristics - 2

Normalized AI: 0.8

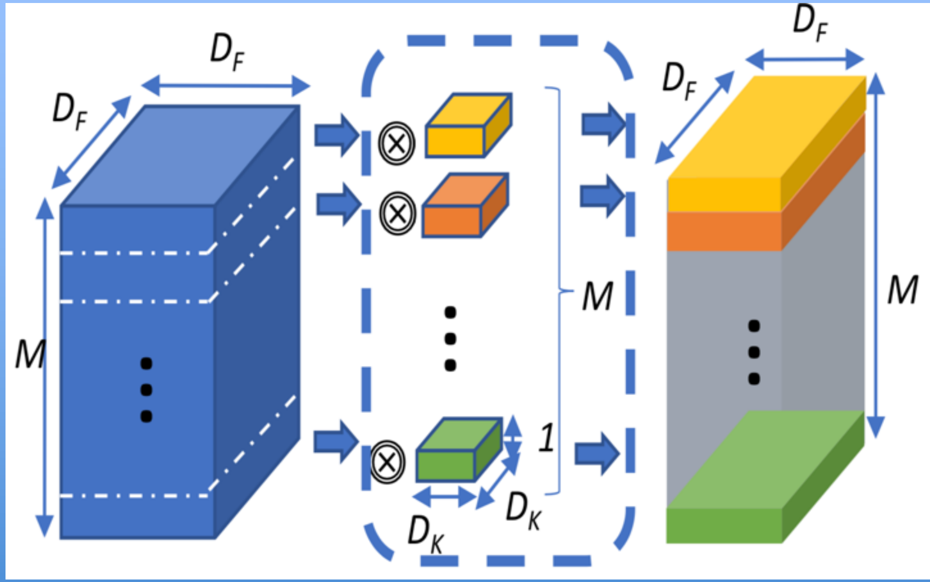
Group Convolution



$$\frac{MND_K^2D_F^2/G}{D_F^2(M+N)+D_K^2MN/G}$$

Normalized AI: 0.02

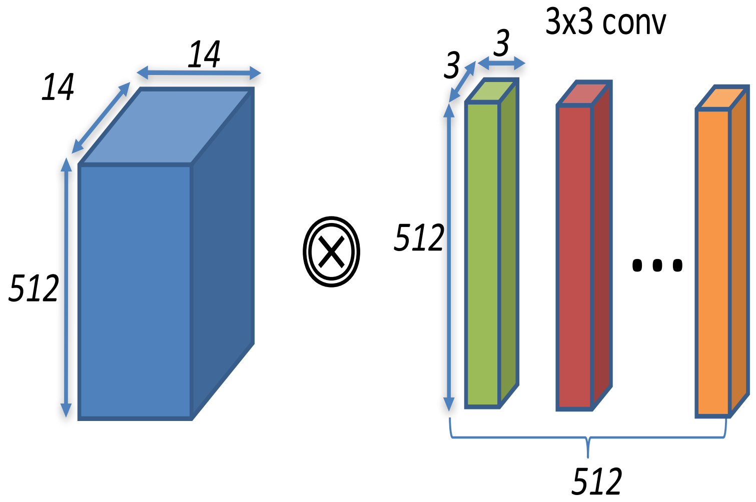
Depthwise Convolution



$$\frac{MD_F^2D_K^2}{2D_F^2M+D_K^2M}$$

Comparision of Arithmetic Intensity of Common CNN Layers

1. AI for layer types is different
2. AI for Depthwise very low
3. But, total FLOPs is much lower
4. Consider Accuracy/(AI/OPS)?



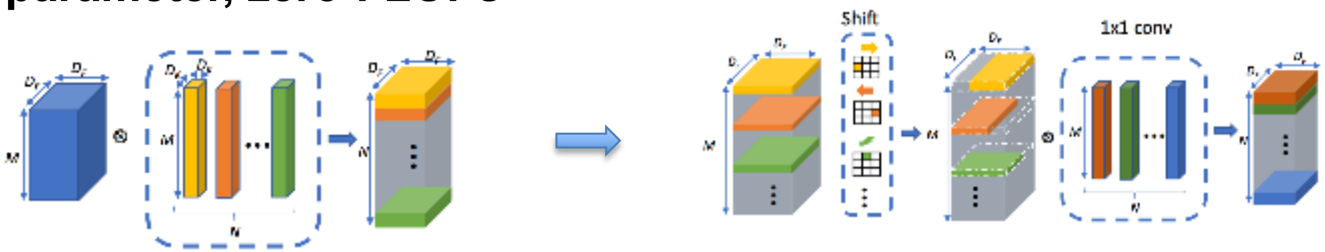
Net Layer	FLOPs (Million)	Memory Ops: param+activation (Million)	Arithmetic Intensity	Normalized Arithmetic Intensity
Spatial convolution	462	2.560	180	1.0
Point-wise convolution	51	.463	110	0.6
Group convolution	116	.790	146	0.8
Depthwise convolution	0.9	.205	4.3	0.02

- D_K: kernel size = pointwise conv 1x1
- D_K: kernel size = spatial conv 3x3
- M: input channels = 512
- N: filters, output channels = 512

- $D_{F:}$ input resolution width, height = 14
- G: Group Size, group convolution = 4
- G: Group Size, depthwise conv = 512

Radical New Architecture: ShiftNet (CVPR spotlight 2018)

- A lesson from SqueezeNet: spatial convolution (3x3, 5x5, etc.) is expensive ...
  - Replace spatial convolutions with the “Shift” operation[1] that requires **zero-parameter, zero-FLOPs**



- Classification:

	Top-1 Acc.	Parameter size	Reduction
AlexNet	57.2	60 million	1X
SqueezeNet	57.5	1.2 million	50X
ShiftNet-C	58.8	0.78 million	77X

- Other tasks:
  - Face verification: 37X parameter reduction
  - Style transfer: 6X parameter reduction

[1] Wu B, Wan A, Yue X, Jin P, Zhao S, Golmant N, Gholaminejad A, Gonzalez J, Keutzer K. Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions. arXiv preprint arXiv:1711.08141. 2017 Nov 22. *CVPR 2018*

Various Research Directions with Shift

Improving the Shift operator

[1] Constructing Fast Network through Deconstruction of Convolution, NeurIPS18 – Learnable shift

[2] AddressNet: Shift-Based Primitives for Efficient Convolutional Neural Networks – GPU implementation of Shift

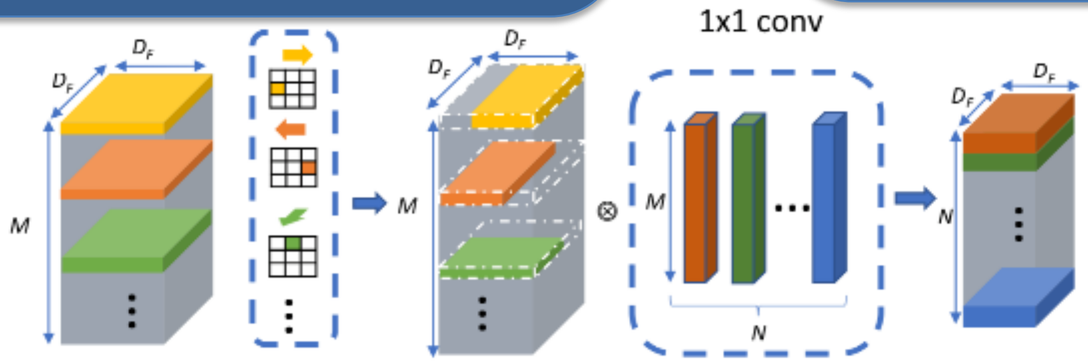
[3] All You Need is a Few Shifts: Designing Efficient Convolutional Neural Networks for Image Classification

Hardware-software Co-design:

[4] Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas, FPGA19 – Shift on FPGA

[5] Mapping Systolic Arrays onto 3D Circuit Structures: Accelerating Convolutional Neural Network Inference

[6] Full-stack Optimization for Accelerating CNNs with FPGA Validation



Applications:

[7] Spatial Shortcut Network for Human Pose Estimation

[8] Temporal Shift Module for Efficient Video Understanding

[9] Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices

[10] Motion feature network: Fixed motion filter for action recognition

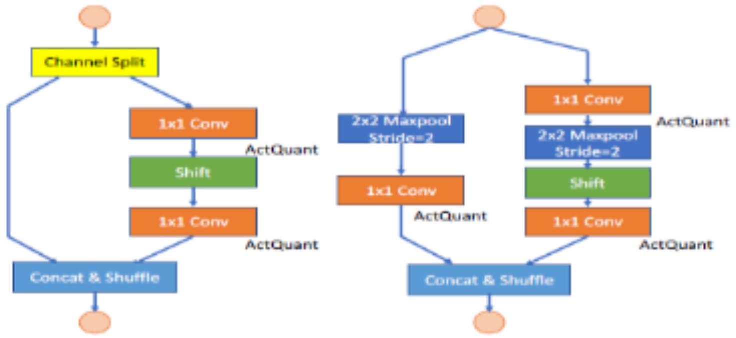
[11] MobiVSR: A Visual Speech Recognition Solution for Mobile Devices

DiracDeltaNet
CNN with No Spatial Convolutions

- Even nets for embedded/edge applications rely heavily on linear algebra
- ShuffleNetV2
  - 1x1 conv
  - 3x3 conv stride=2
  - 3x3 depth-wise conv stride=1
  - 3x3 depth-wise conv stride=2
  - 3x3 max-pooling
  - Shuffle and concatenation
- But … FPGAs are relatively weaker at linear algebra but stronger at supporting Boolean operations and low precision

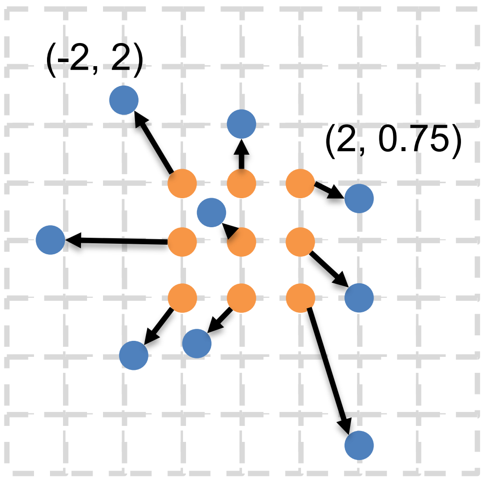
- To avoid FPGA's weaknesses and exploit FPGA strengths we created:
- DiracDeltaNet
  - No spatial (e.g. 3 x 3) convolutions
  - 1x1 conv
  - 2x2 max-pooling
  - Shift
  - Shuffle and concatenation

  - 4 bit weights
  - 4 bit activations



Deformable convolutions –see poster – Jenny Huang

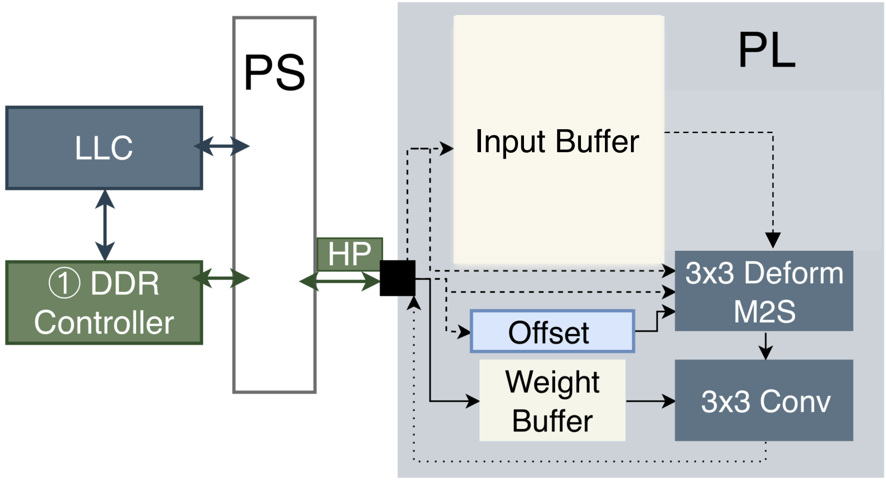
Algorithm Modification:



0. Original Deformable

Accuracy [1](mIoU ↑): **79.9**

Hardware Optimization:

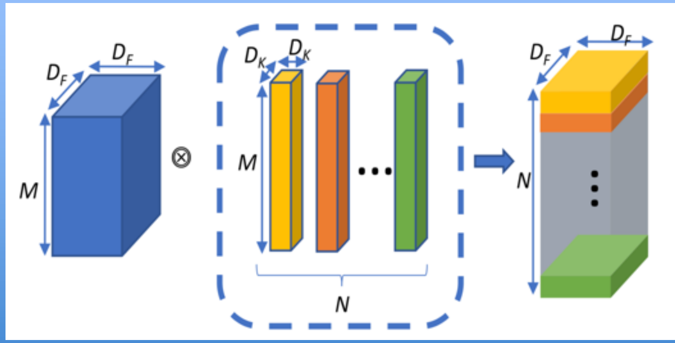


- **Preloads weights to on-chip buffer**
- **Loads input and offsets directly from DRAM**

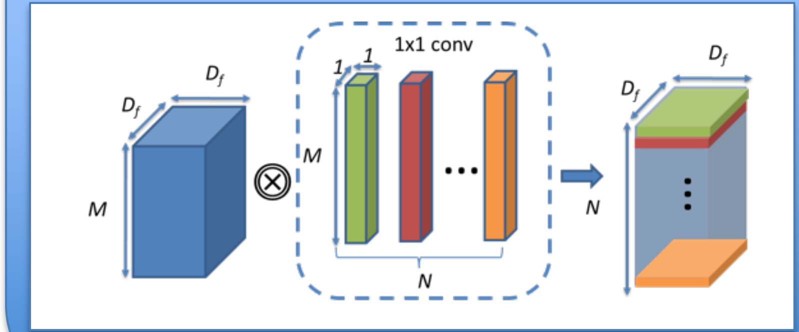
[1] Accuracy for Semantic Segmentation on CityScapes

The DNN Architect's Palette

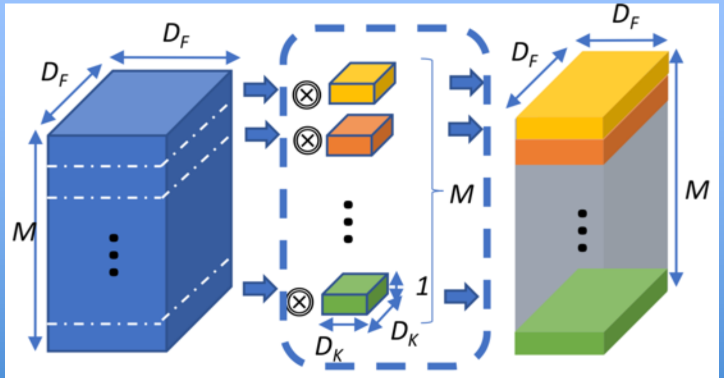
Spatial Convolution e.g. 3x3



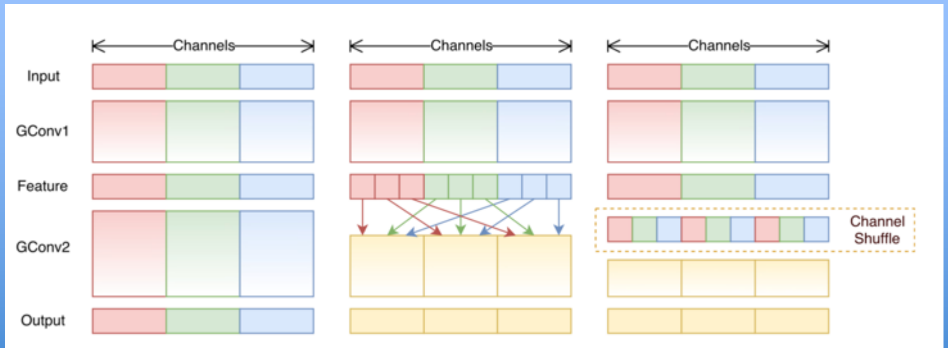
Pointwise Convolution 1x1



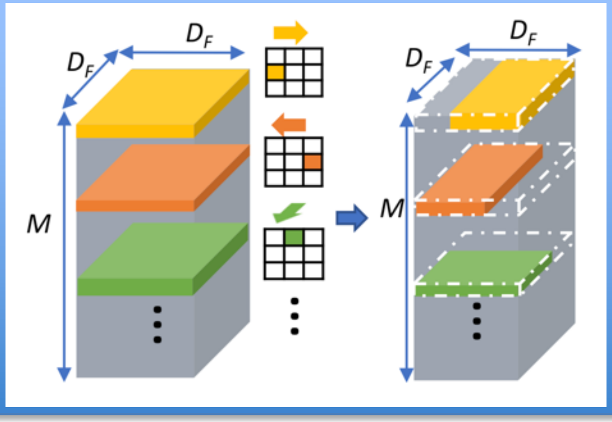
Depthwise Convolution



Channel Shuffle



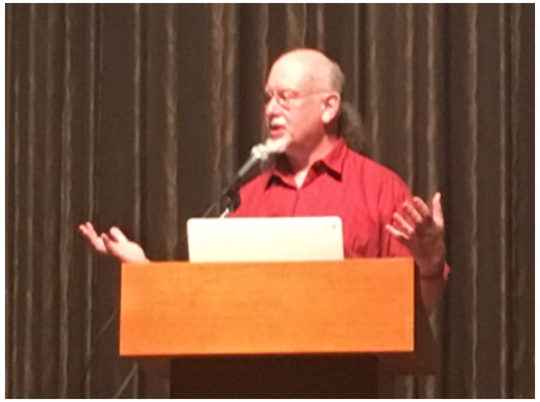
Shift



Small Neural Nets Are Beautiful
Keynote– ESWeek – Art of DNN Design

The Art

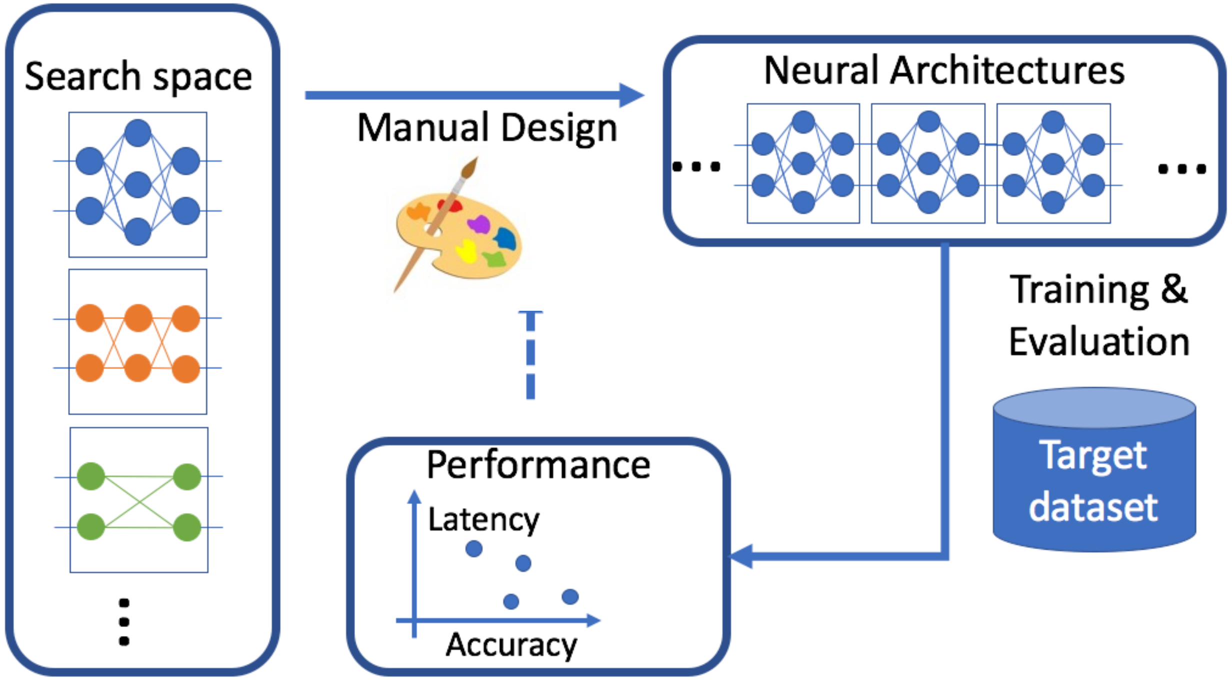
- Overall architecture: economize on layers while retaining accuracy
- Layer types
  - Kernel reduction: 5 x 5 → 3 x 3 → 1 x 1
  - Channel reduction: e.g. FireLayer
  - Experiment with novel layer types that consume no FLOPS
    - Shuffle
    - Shift
- Residual connections



Iandola, Forrest, and Kurt Keutzer. "Small neural nets are beautiful: enabling embedded systems with small deep-neural-network architectures." In *Proceedings of the Twelfth International Conference on Hardware/Software Codesign and System Synthesis Companion*, p. 1. ACM, 2017. (ESWeek 2017). Also, (arXiv:1710.02759)

From Dark Art to Art: Designing a Deep Neural Net

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



Berkeley DeepDrive

Moving DNN Design from an Art to a Science

BAIR

DNN Model Architecture space: $\mathcal{A}$

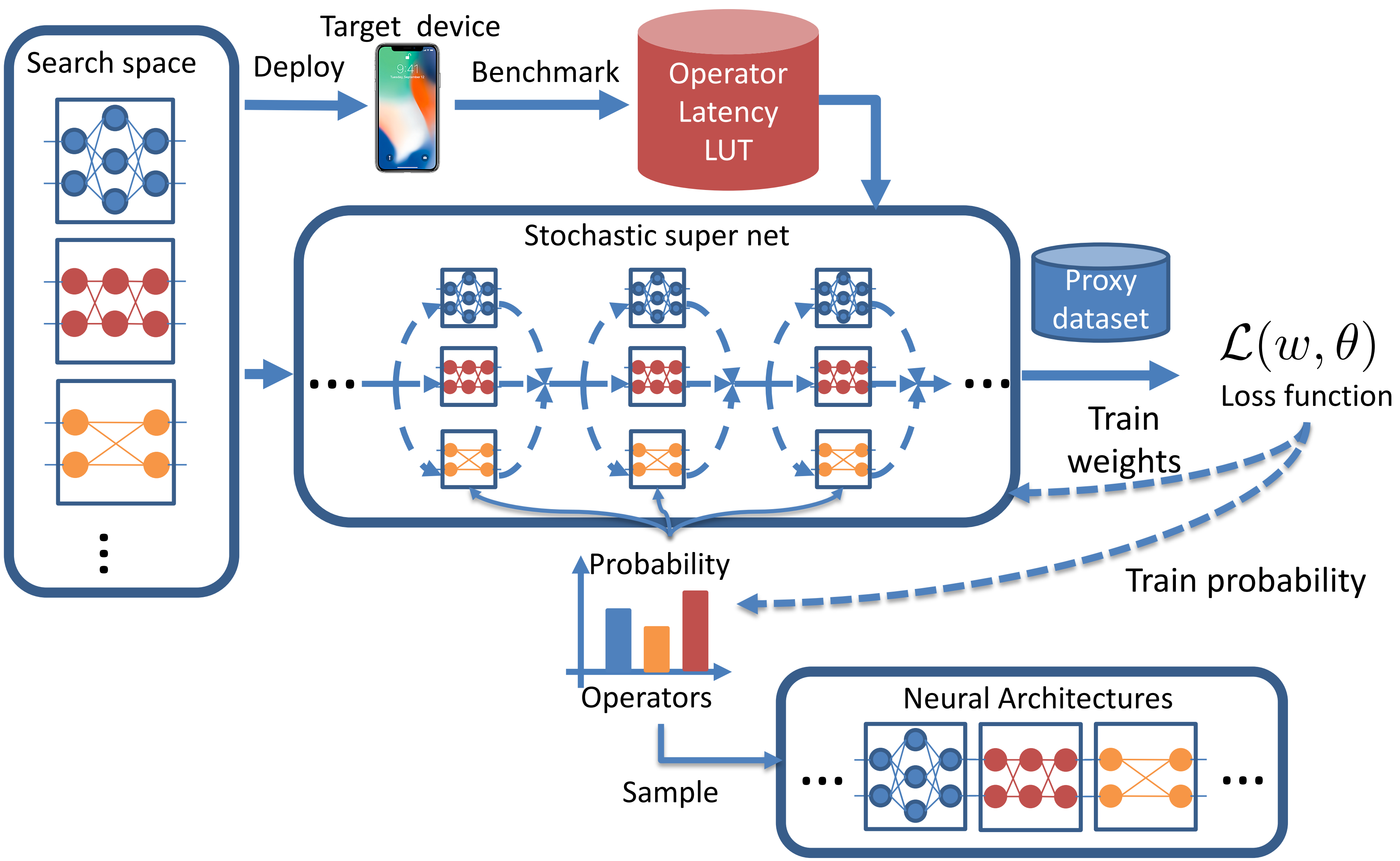
candidate: a

weights: w_a

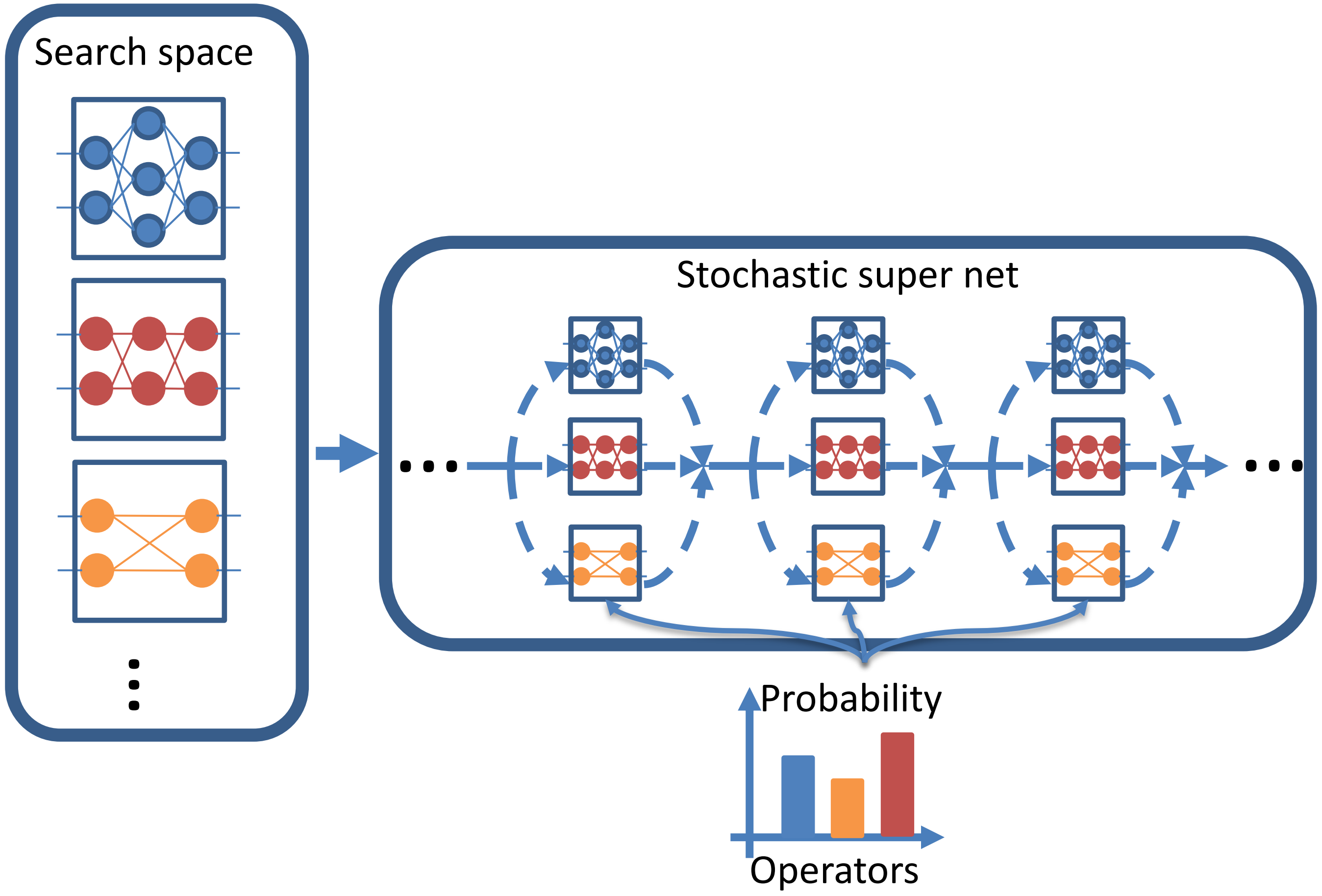
Problem formulation:

$$\underbrace{\min_{a \in \mathcal{A}} \underbrace{\min_{\boldsymbol{w}_a} \mathcal{L}(a, \boldsymbol{w}_a)}_{\text{Inner problem: training a neural network}}}_{\text{Outer problem: enumerating candidates architectures in } \mathcal{A}}$$

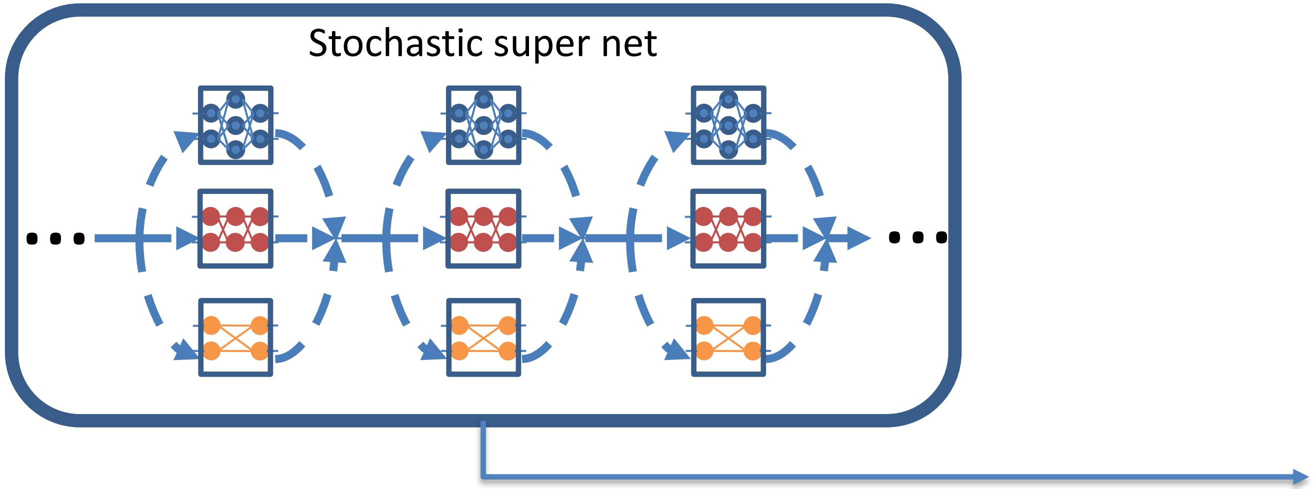
Our approach: Differentiable Neural Architecture Search

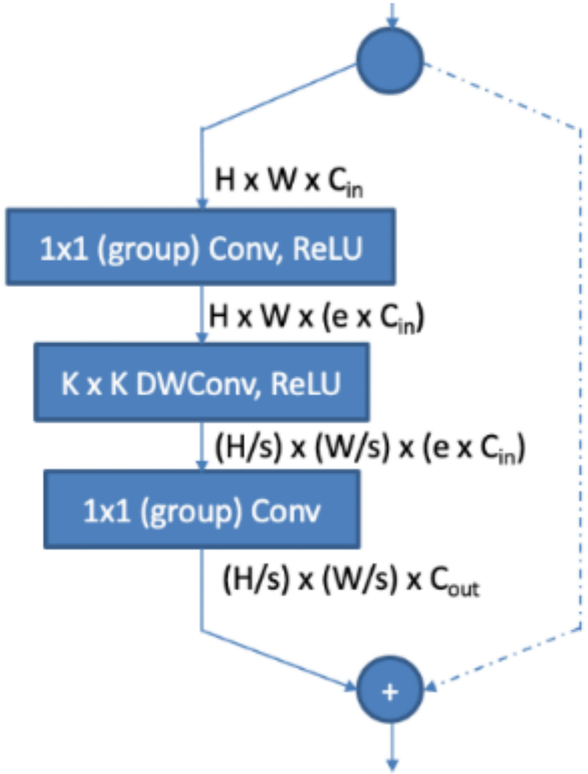


Define the Search Space



An instantiation of a Stochastic Super Net

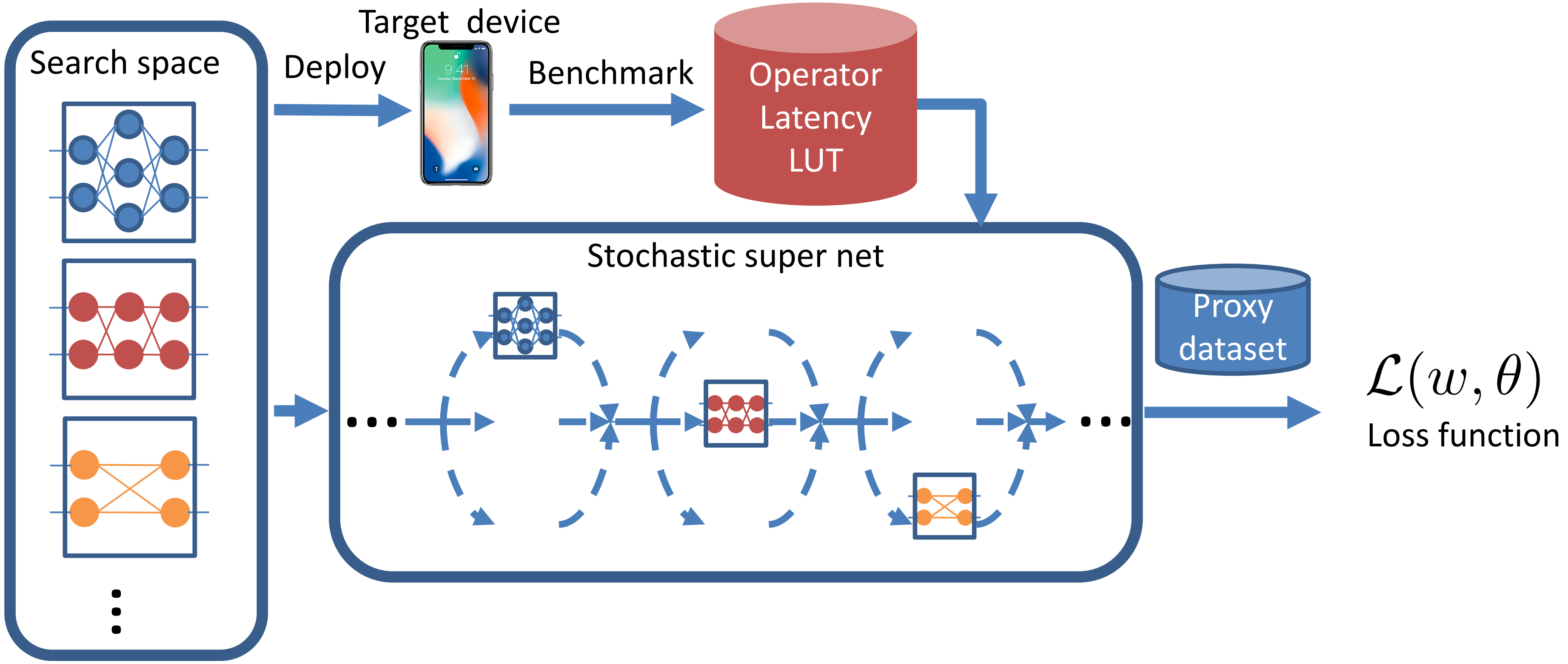




- A 22 layer network divided into 7 groups
- Base channel sizes of each group is pre-determined
- Down-sampling at the first convolution at group {1, 3, 4, 5, 6}
- Each layer can choose a variation of the template module

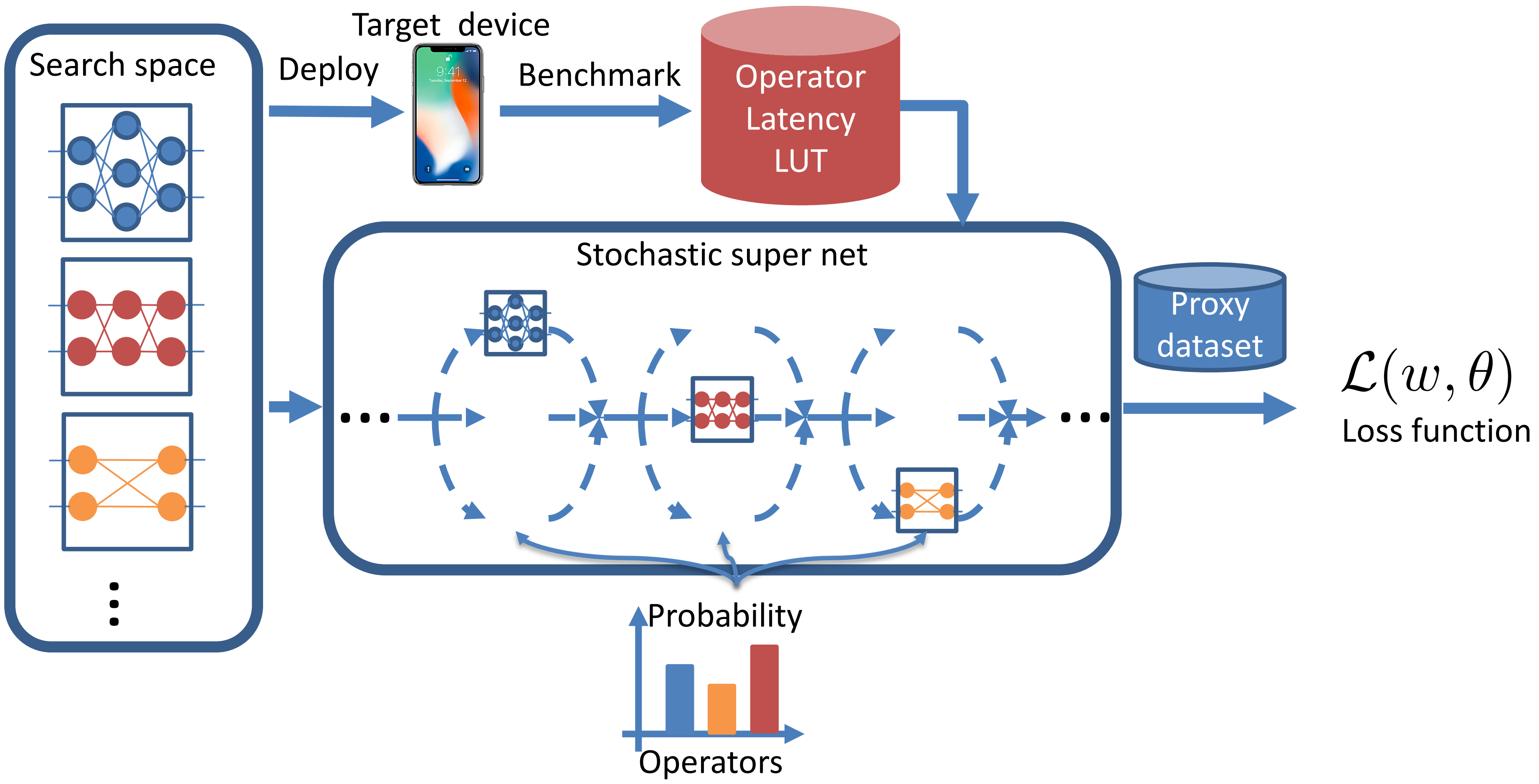
Layer template with different configurations of
- Kernel size of the depthwise convolution
- Expansion ratio (Channel size)

Measuring Latency of a Point (DNN) in the Design Space



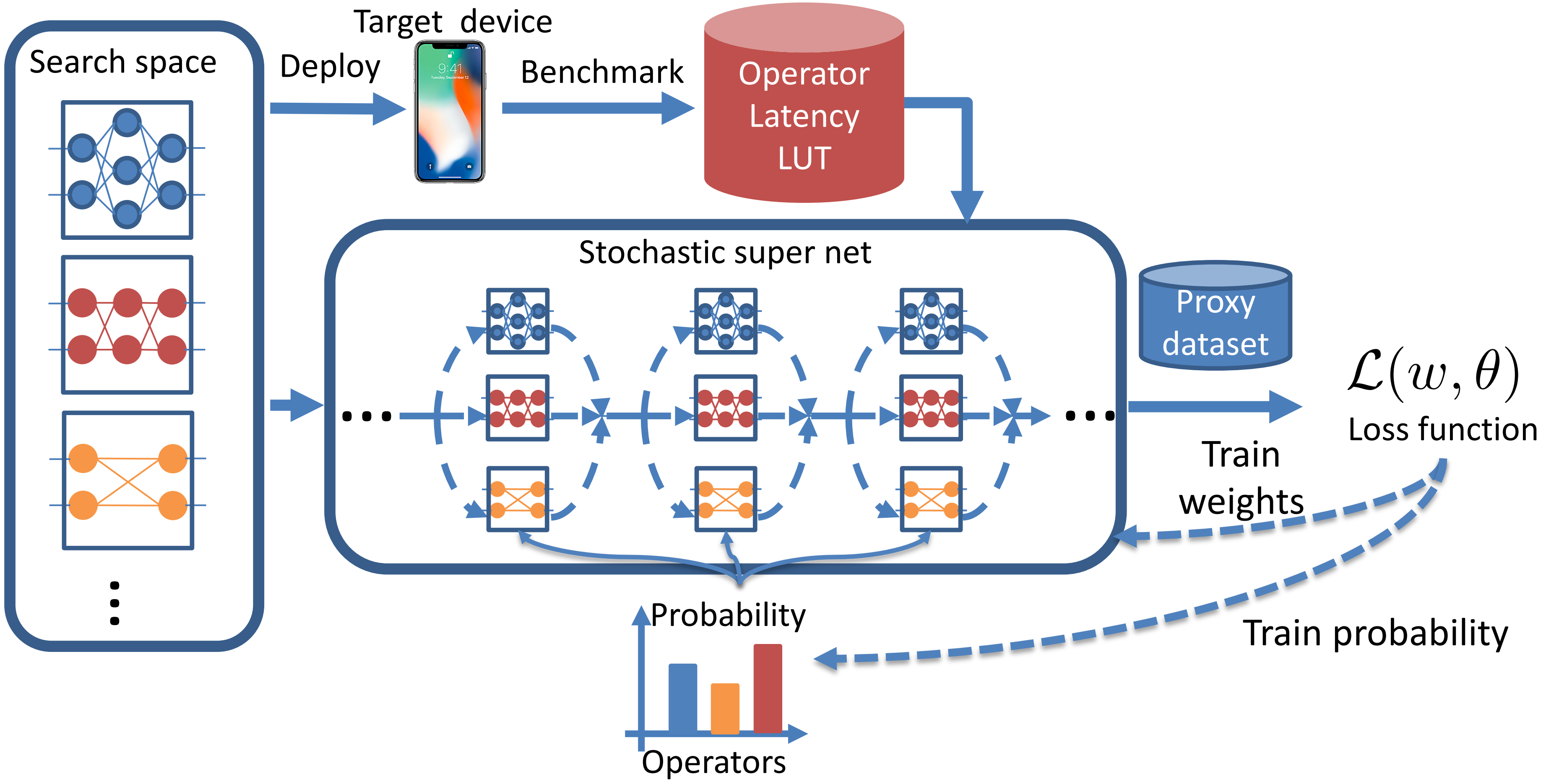
- We want to move from measuring:
  - MACs, model parameters
- To latency on the real target

Measuring Latency of a Point (DNN) in the Design Space

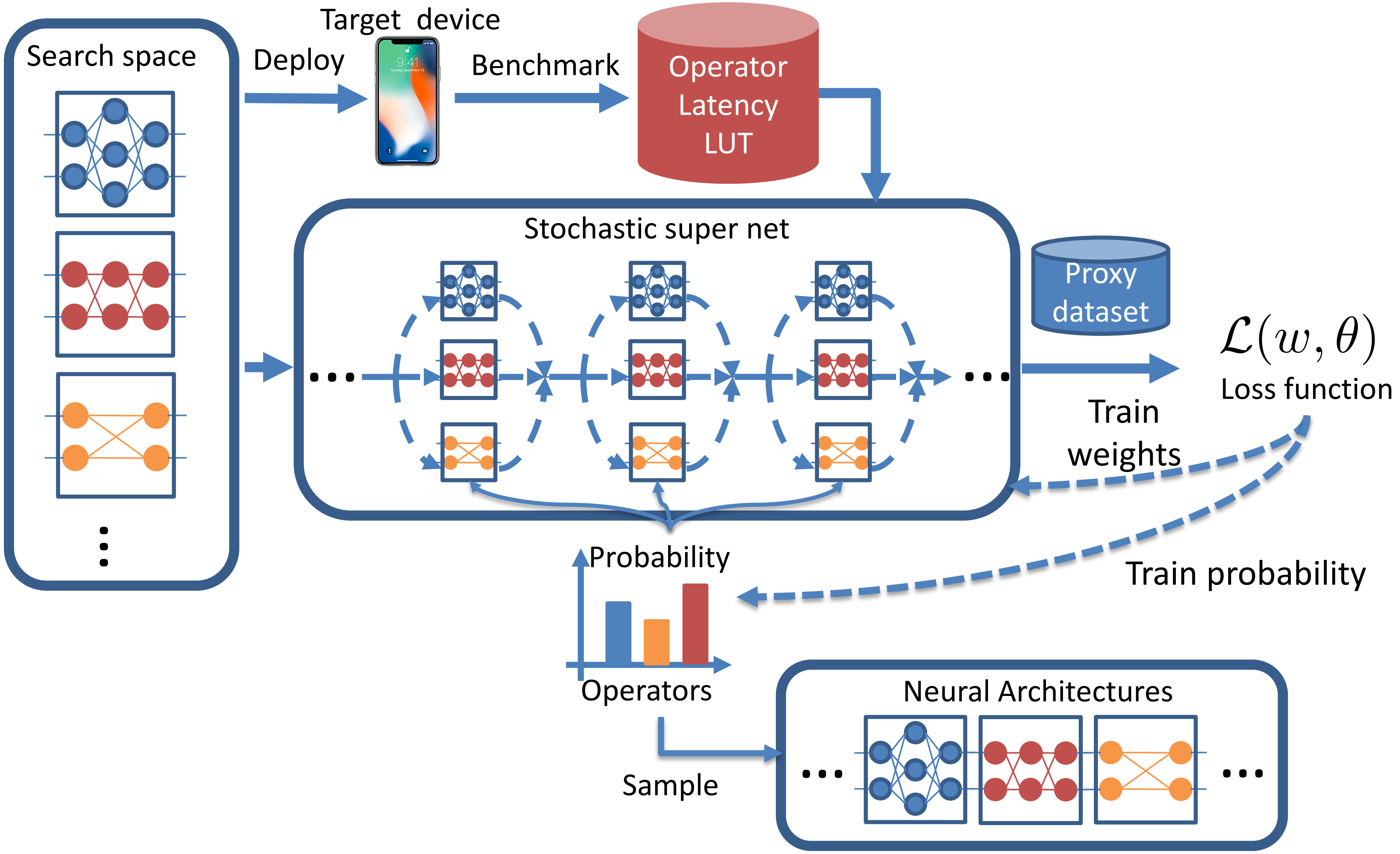


- Approximate overall latency as sum of the latency of the layers of the DNN

Train with to Minimize Loss Function



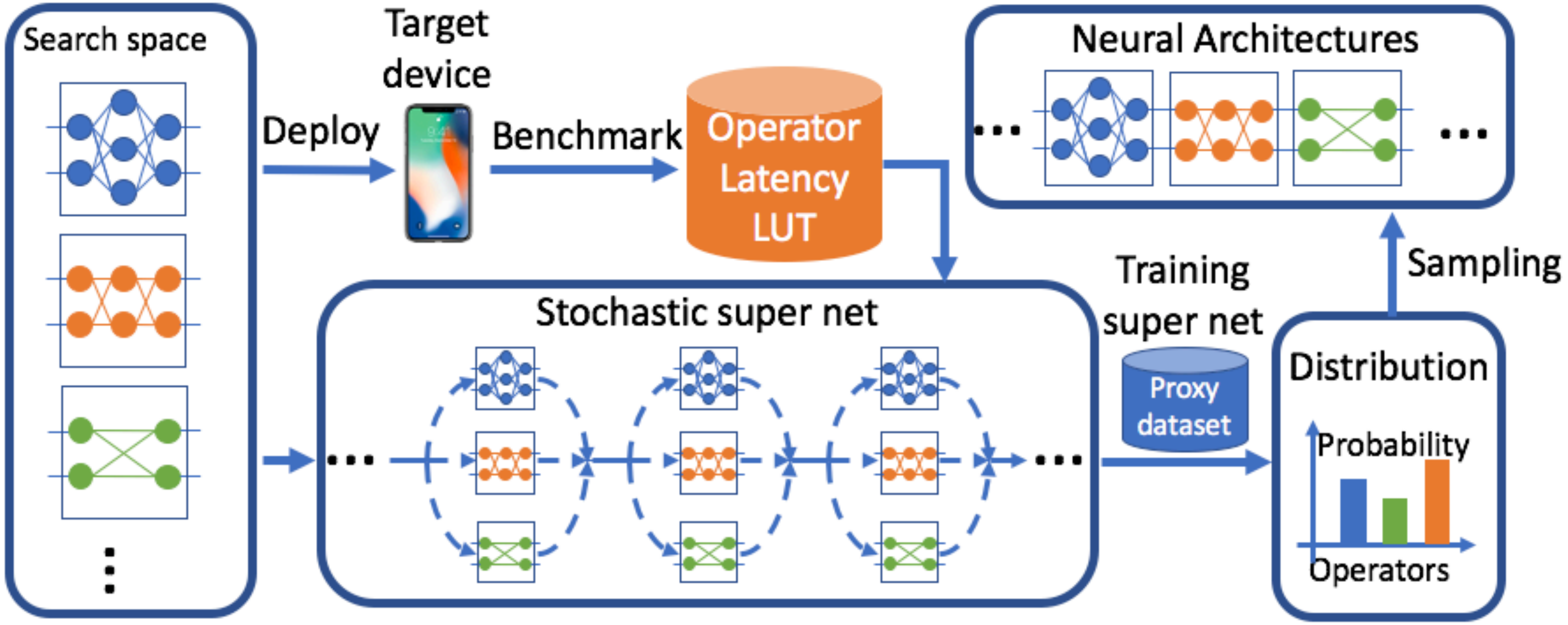
Our approach: Differentiable Neural Architecture Search



DNAS: Differentiable Neural Architecture Search
CVPR 2019 Oral

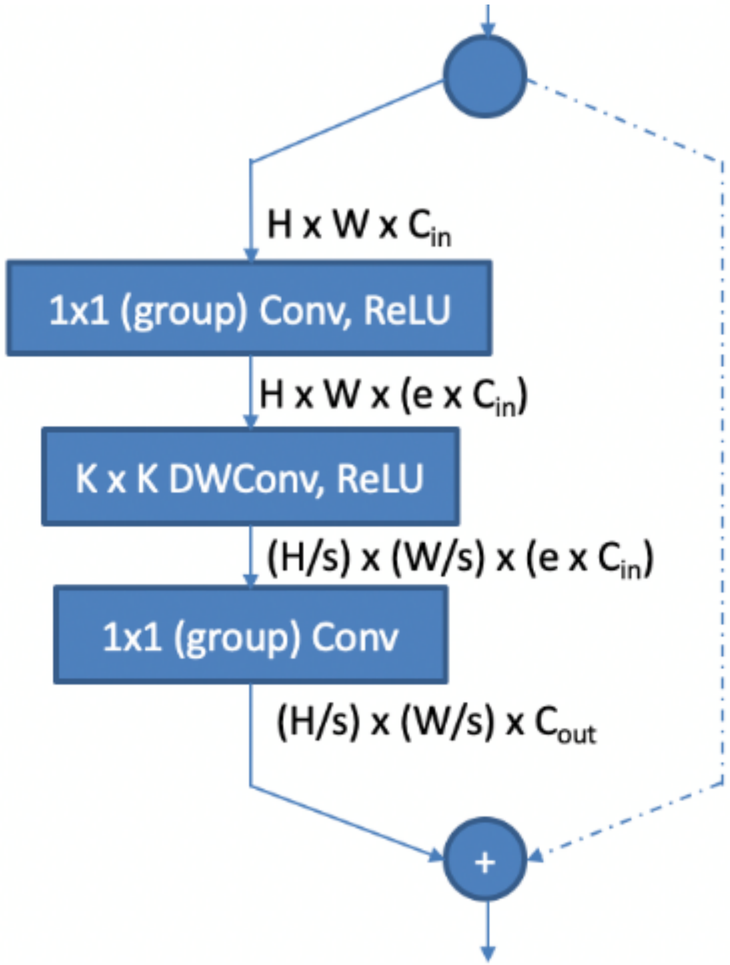
Differentiable Neural Architecture Search:

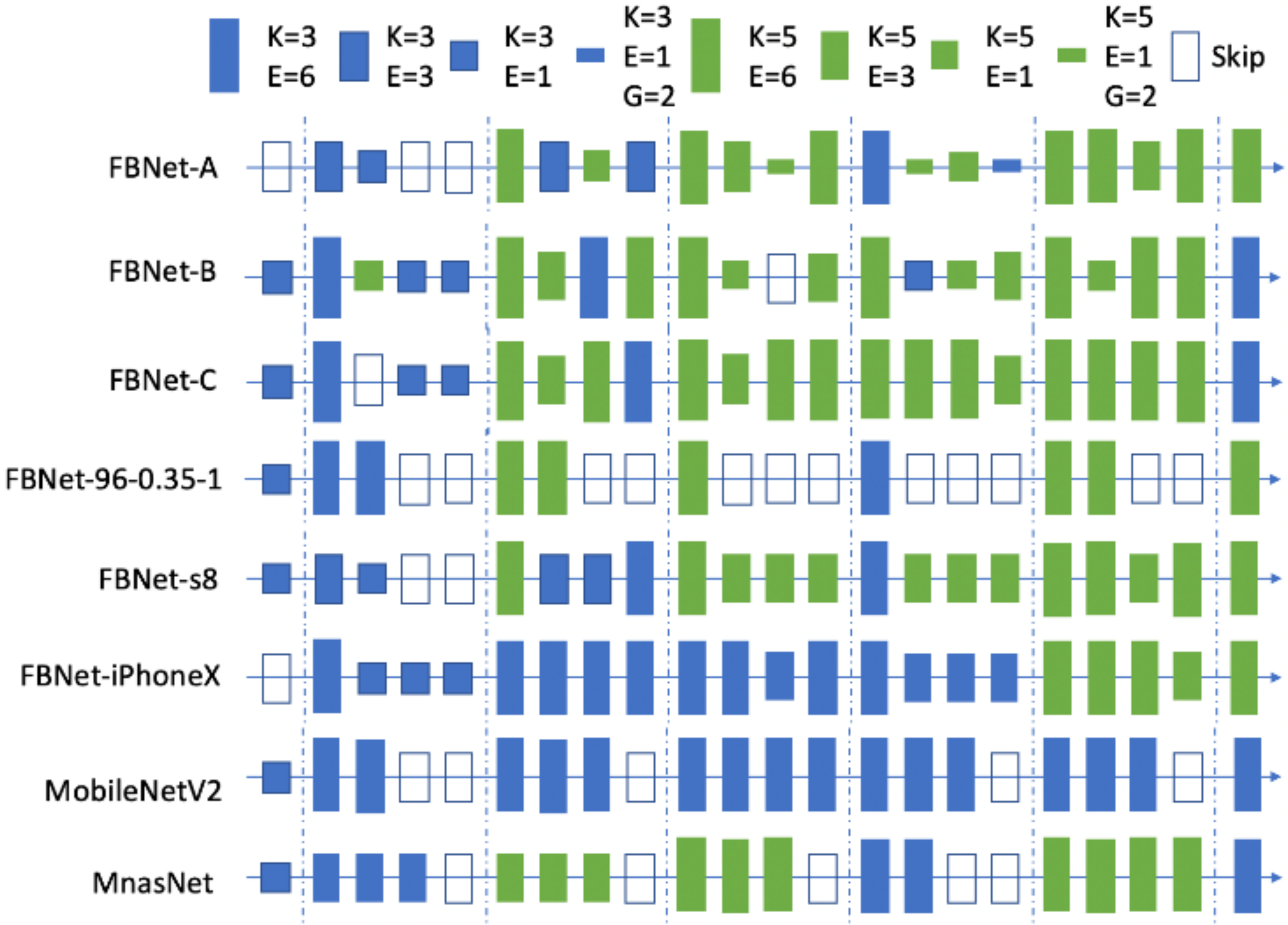
- Extremely fast: 8 GPUs, 24 hours
- Optimize for actual latency
- General: can be applied to different problems



In collaboration with FB
Peizhao Zhang,
Yanghan Wang,
Fei Sun,
Yiming Wu,
Yuandong Tian,
Peter Vajda,
Yangqing Jia

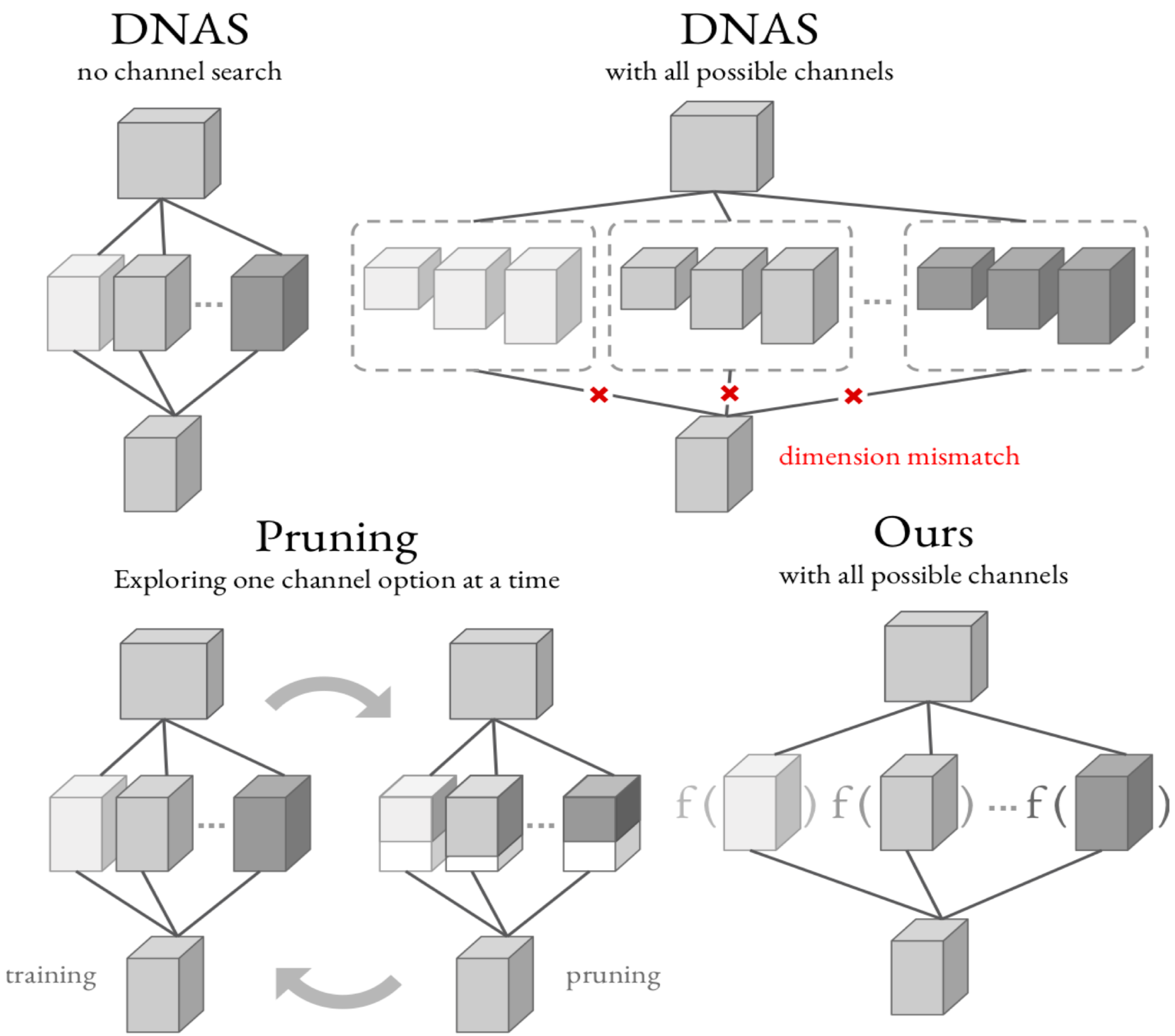
Visualization of Mobile DNNs





Work Continues: FBNet with Channel Search
Alvin Wan and FB

- (top-left) **Original DNAS** does not include channel search
- (top-right) **DNAS** with naive modifications cannot support channel search.
- (bottom-left) **Pruning** can only train one potential architecture at a time.
- (bottom-right) **Our DNAS** can jointly search over multiple channel options.

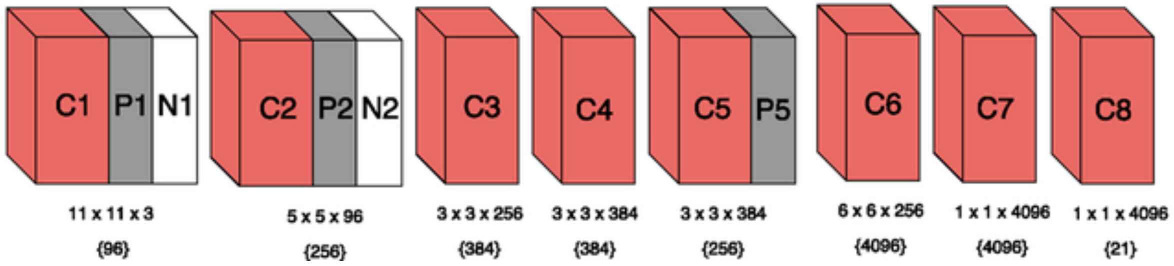


How Does this Address the Challenges of DNN Design

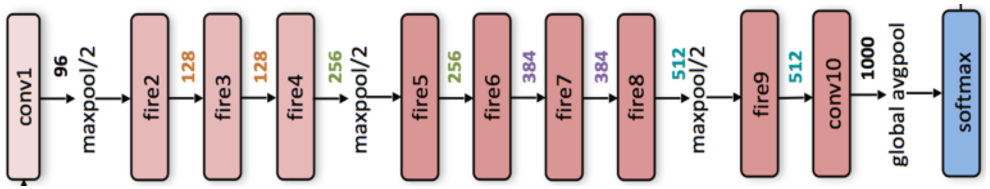
- Challenge 0: Comprehensibility
  - Our considerations are moved to a higher level:
    - Layer definitions, number of layers, activation function
  - We don't have to understand the DNN, the optimization system does
- Challenge 1: Large Design Space
  - Use Stochastic Gradient Descent to efficiently search space
- Challenge 2 and 4: Diverse Targets, Hard to measure
  - Precharacterize layers on targets and gather real latency and energy data
- Challenge 3: Constraints are tight
  - We integrate constraints into the optimization

Further Memory Reductions Through Pruning and Quantization

AlexNet [1]



SqueezeNet [2]



Model Design

50X

CNN	Top-5 Accuracy ImageNet	Model Parameters	Model Size
AlexNet[1]	80.3%	60M	243MB
SqueezeNet[2]	80.3%	1.2M	4.8MB

10X

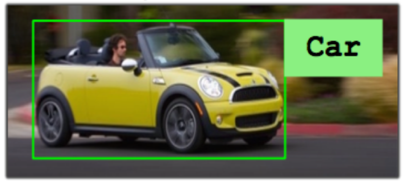
compresses to 500KB

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." NIPS2012

[2] Iandola, Forrest N., et al. "**SqueezeNet**: AlexNet-level accuracy with 50x fewer parameters and< 1MB model size." arXiv: 1602.07360 (2016). (February 2016) 1606 Citations

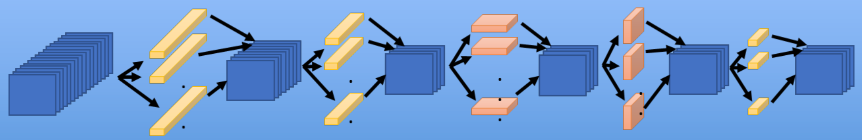
Deep Neural Net Design, Training, and Implementation

Rapidly Training the DNN

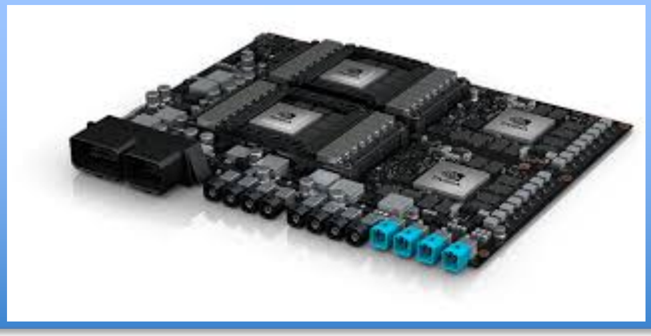


Aggregating training data

Finding the right Deep Neural Network model



Efficiently implementing the DNN on embedded HW / co-design DNN accelerators



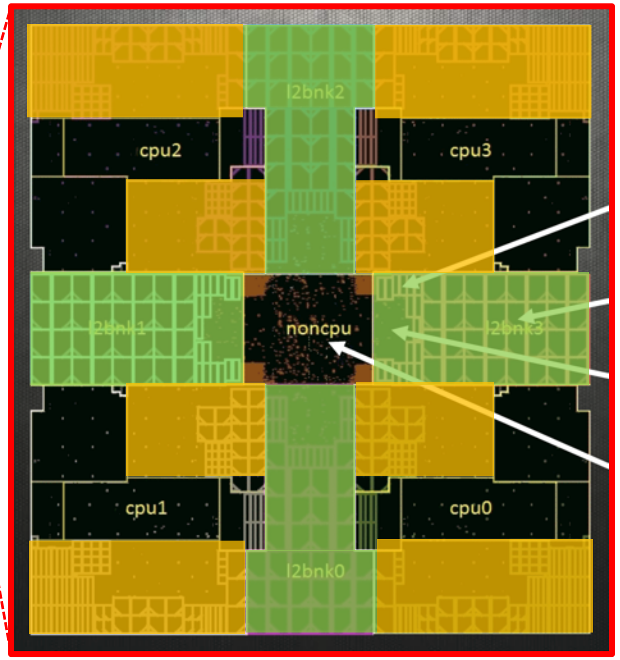
Quantization of weights, activations: Our Workhorse Optimization

Quantization is a very powerful approach:

- Diminishes our biggest cost: memory traffic on and off-chip

Speed and Energy More Impacted by Memory Access than Computation

Galaxy S7



L1 Cache/TLB

L2 Cache

In 45nm process
16 bit operands

- Add: 1
- Mult: 3.4
- Offchip DRAM access: 3556x

M. Horowitz, “Computing’s Energy Problem (and what we can do about it)”, 2014 ISSCC

Samsung Exynos M1 Access Times

	L1 D-Cache (per core)	**L2 Cache (shared)**	**Off-chip DRAM**
Size	32 KB	2 MB	4 GB
Read Latency	4 cycles	22 cycles	**~200 cycles**
Read Bandwidth	20.8 GB/s	166.4 GB/s	28.7 GB/s

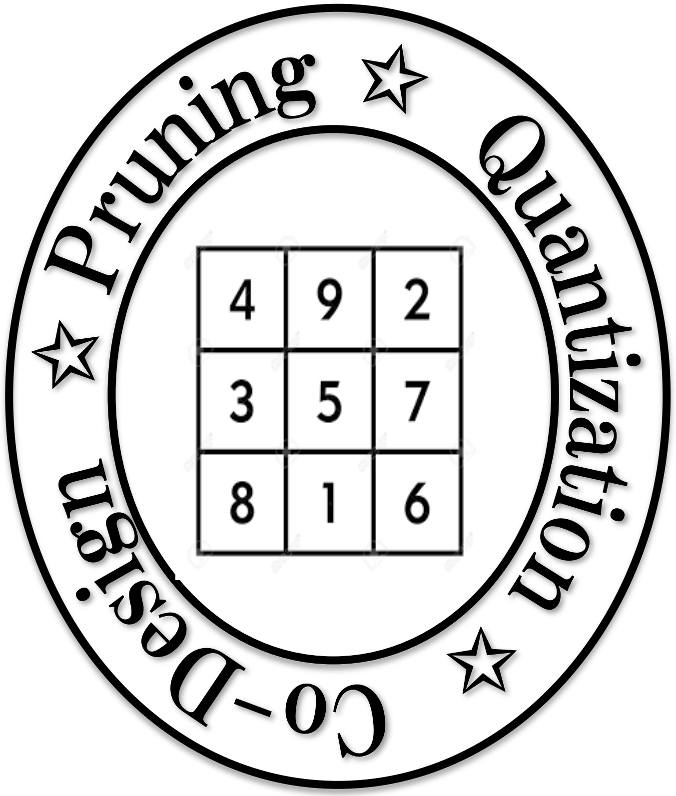
Quantization of weights, activations: Our Workhorse Optimization

Quantization is a very powerful approach:

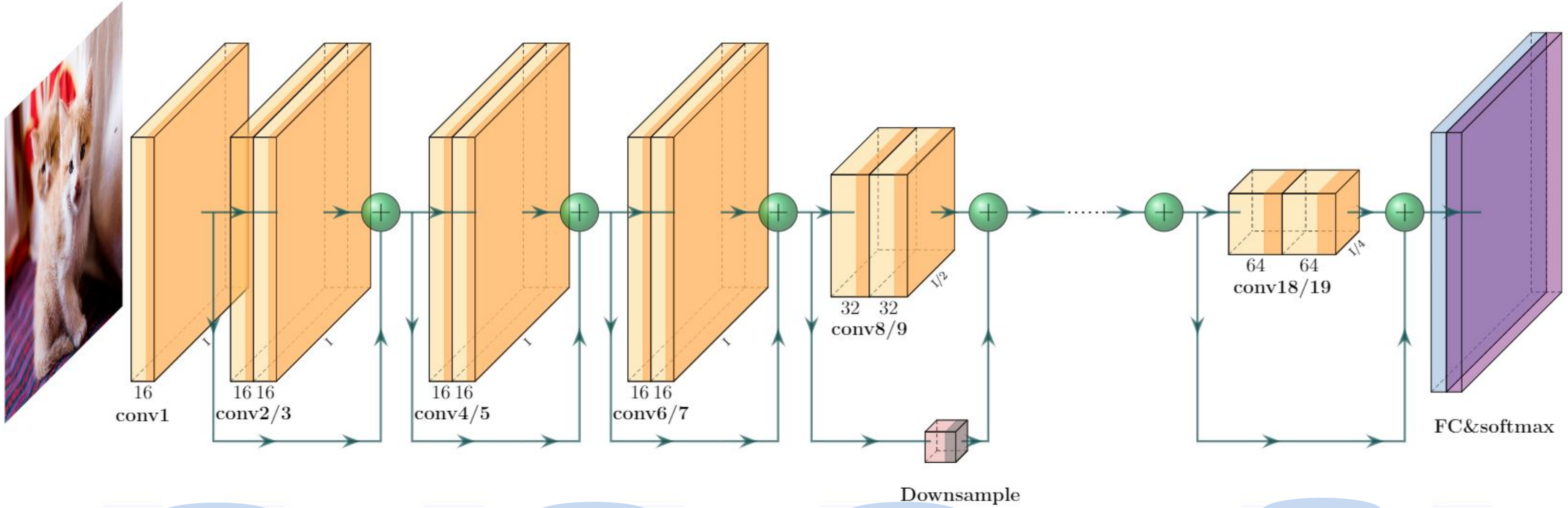
- Diminishes our biggest cost: memory traffic on and off-chip

Good for reducing power and latency

- Lots of "tricks" and expensive hyper-parameter tuning
- **Ad-hoc rules that do not generalize**



Mixed-Precision: Exponential Search Space



2-bit	2-bit	2-bit	2-bit
4-bit	4-bit	4-bit	4-bit
8-bit	8-bit	8-bit	8-bit
16-bit	16-bit	16-bit	16-bit

The bit configuration **grows exponentially** with the number of **layers**

4 Billion configurations for a 16 layer network

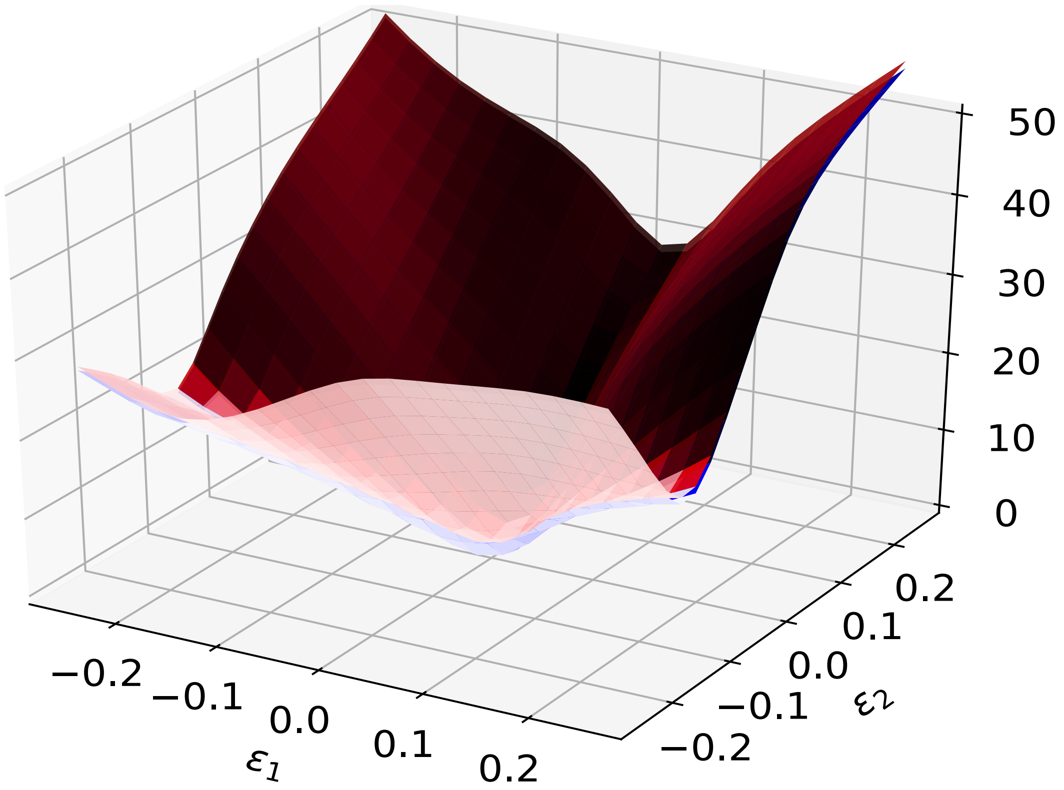
Various Approaches to Quantization

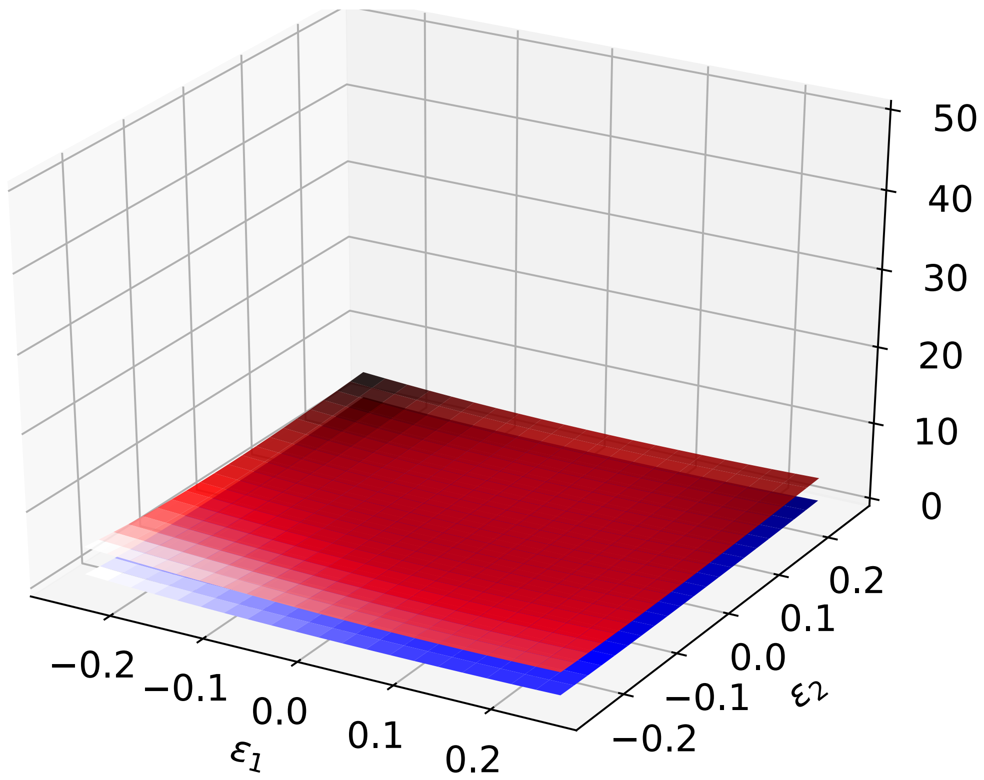
- One approach: reduce the search space by limiting quantization precision options to a uniform precision
  - Example: Deep Compression [1]
- This leads to significant accuracy drop. To address this many modifications were proposed for uniform precision quantization, such as:
  - PACT: Limit activation range through clipping after ReLU [2]
  - LQ-Net: Instead of using min/max values learn the clipping range [3]
  - RVQuant: Use a mixture of uniform and non-uniform quantization [4]

  However, uniform quantization to low bits leads to **significant accuracy degradation** despite using all of the above modifications/improvements
- AutoML based methods that search an exponentially large search space and test several different bit-precision configurations; however, they are VERY expensive

[1]Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv:1510.00149.
[2]Choi J, Wang Z, Venkataramani S, Chuang PI, Srinivasan V, Gopalakrishnan K. PACT: Parameterized clipping activation for quantized neural networks. (arXiv:1805.06085) 2018.
[3]Zhang D, Yang J, Ye D, Hua G. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. ECCV'18.
[4]Park E, Yoo S, Vajda P. Value-aware quantization for training and inference of neural networks. ECCV'18.

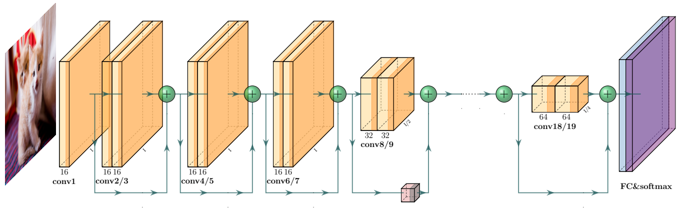
Key Observation of Our Work



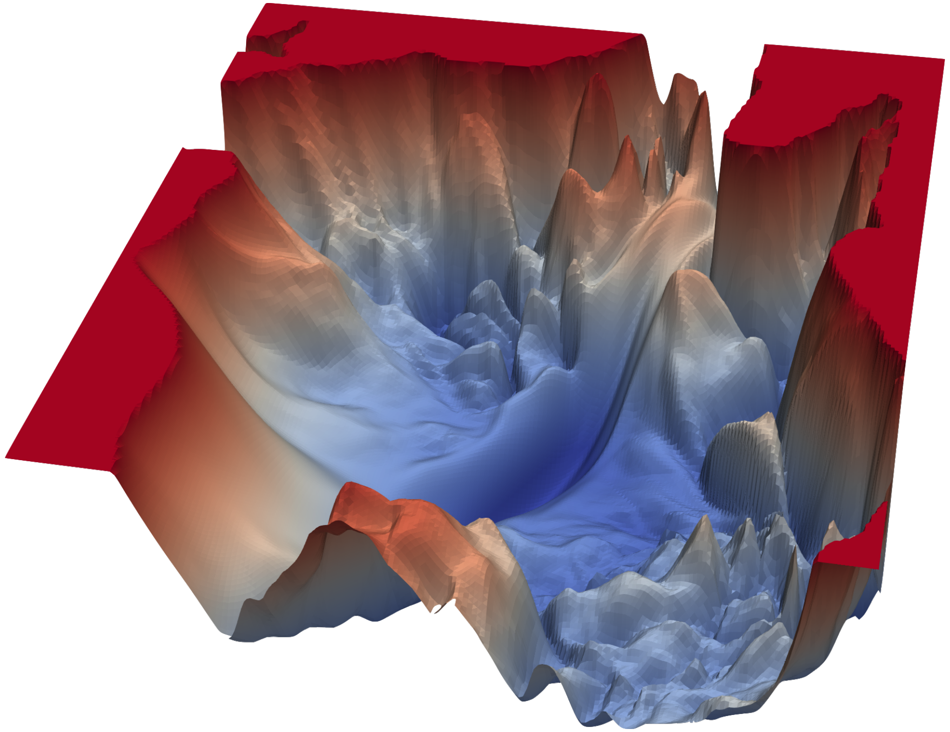


- All you DSP old-timers know, when you move from float to fixed to int we move from a single point to a discrete region in the domain: how do we know whether this will impact the accuracy?

What if we better understood the loss landscape?



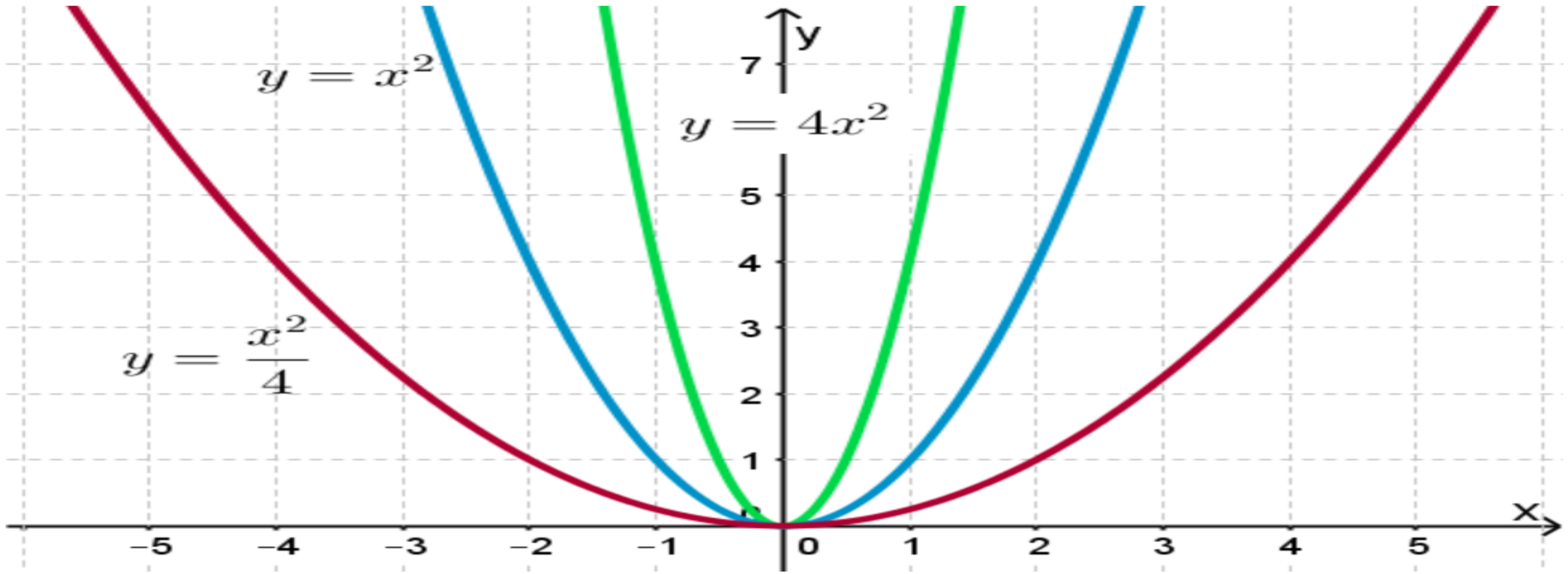
$$\min_{w} \mathcal{J}(w) = \frac{1}{N} \sum_{i=1}^{N} cost(w, x_i)$$



VGG 56

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the loss landscape of neural nets." In *Advances in Neural Information Processing Systems*, pp. 6389-6399. 2018.

The Second Derivative Tells us More About the Shape of a Function (e.g. Loss Function)



- At the origin, the first derivative of $y = x^2$, $y = \frac{1}{4} x^2$, $y = 4 x^2$ is all the same: 0
- But, the second derivatives give more information: 2, ½, and 8 respectively

Use the Hessian Matrix of the DNNs Model Parameters

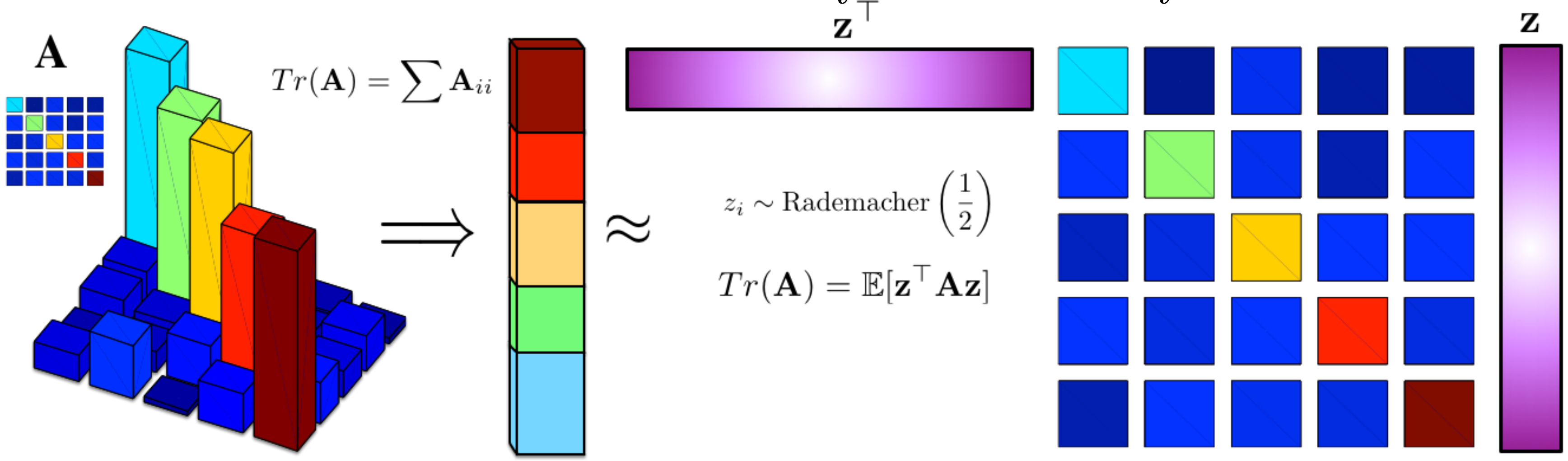
$$\mathbf{H}_{\mathrm{e}}(\mathbf{x}) = \begin{bmatrix} \partial^2_{x_1 x_1}\mathrm{e} & \cdots & \partial^2_{x_1 x_i}\mathrm{e} & \cdots & \partial^2_{x_1 x_N}\mathrm{e} \\ \vdots & \ddots & \vdots & ⋰ & \vdots \\ \partial^2_{x_i x_1}\mathrm{e} & \cdots & \partial^2_{x_i x_i}\mathrm{e} & \cdots & \partial^2_{x_i x_N}\mathrm{e} \\ \vdots & ⋰ & \vdots & \ddots & \vdots \\ \partial^2_{x_N x_1}\mathrm{e} & \cdots & \partial^2_{x_N x_i}\mathrm{e} & \cdots & \partial^2_{x_N x_N}\mathrm{e} \end{bmatrix}$$

- The Hessian Matrix can give us comprehensive information about the Loss Landscape curvature
- Each Hessian matrix entry computes how fast gradient values are changing in different direction -> gradient of gradient

Fortunately, We Don't have to Explicitly Form Hessian

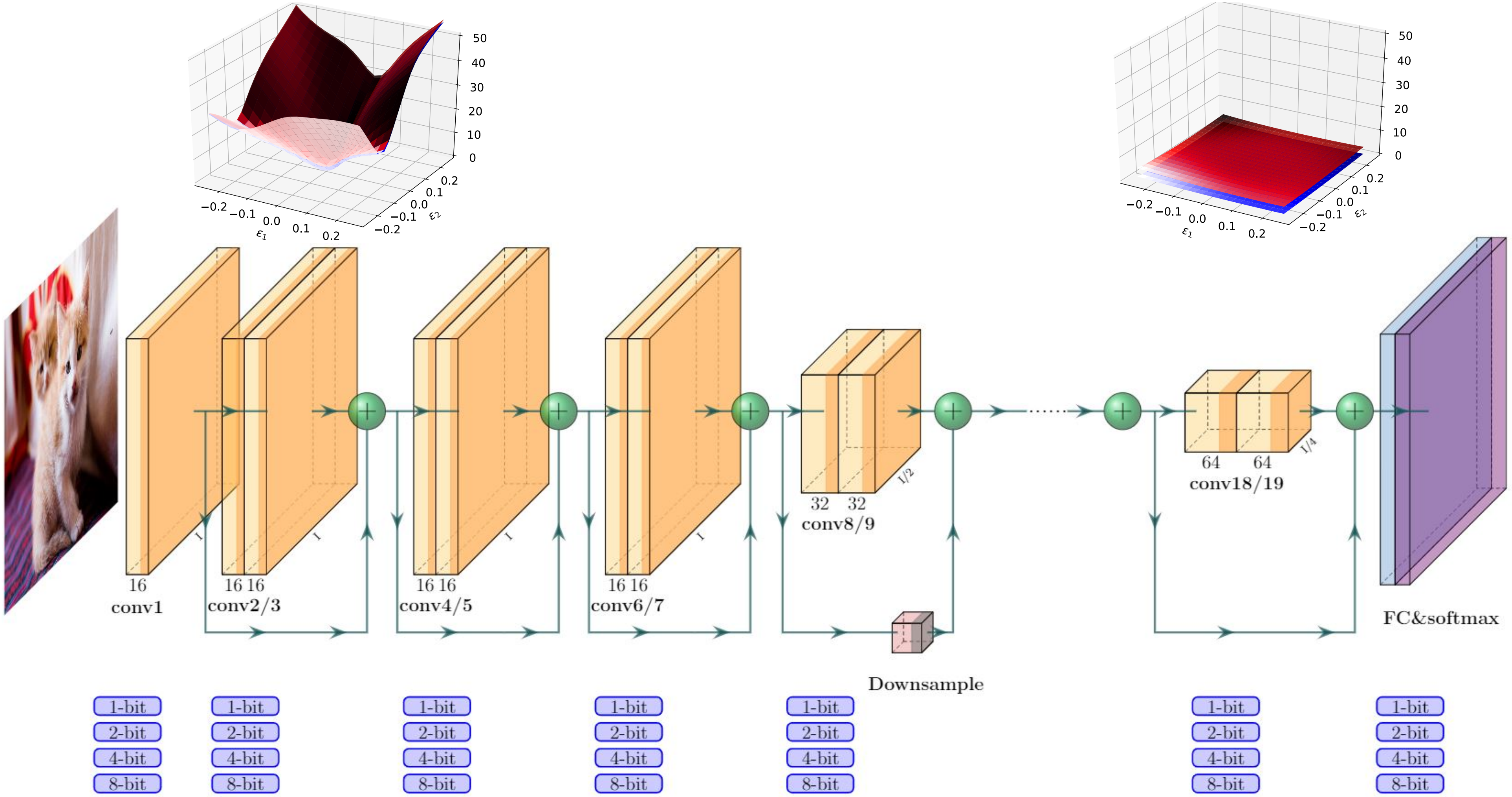
- We only need Hessian eigenvalues and not the matrix itself
- Eigenvalue computation only needs multiplying H to random vectors (so called power iteration)
  - The matrix-vector multiplication can be done by a second gradient backpropogation
  - Therefore, **no need to form the full Hessian matrix.**
  - We can compute Hessian spectrum using **randomized methods**

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



Surpass Ad Hoc Approaches by Using Hessian
ICCV 2019- Zheng Dong

Only quantize to **ultra-low precision** those layers that have **small Hessian spectrum (relatively flat)**



Hessian AWare Quantization

Contributions of HAWQ:

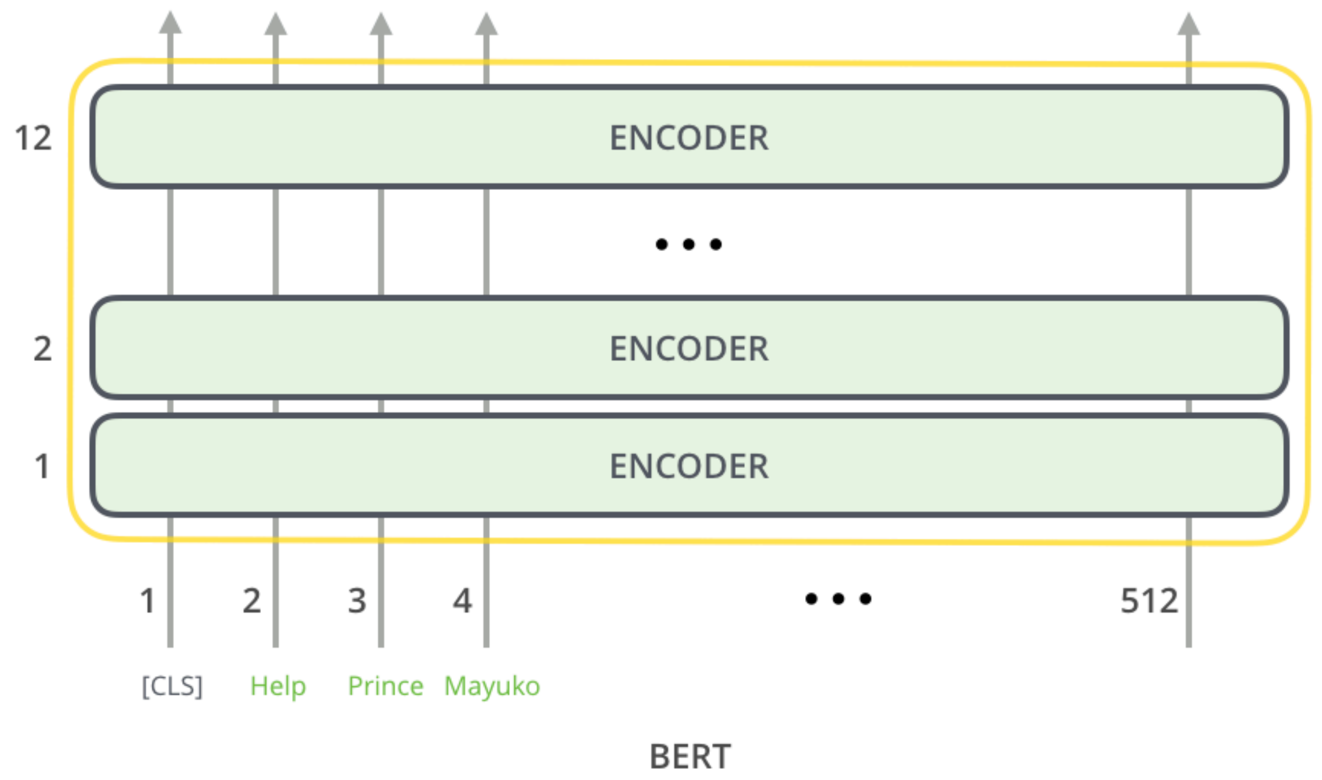
- A systematic, **second-order** algorithm for inference quantization
- Fine-tuning schedule based on second-order statistics
- Novel compression results exceeding **all existing state-of-the-art** methods
- **No more ad-hoc tricks**

- Dong, Zhen, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. "HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision." *arXiv preprint arXiv:1905.03696* (2019). ICCV 2019
- Dong, Zhen, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. "HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks." arXiv preprint arXiv:1911.03852 (2019).

Cloudfree Mobile NLP

- Another important application of quantization is to enable In-Car NLP with real-time inference and without the need for cloud connectivity

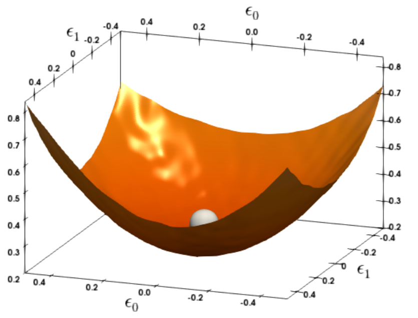
Natural Language Processing with BERT
Big Model → Big Challenge for Quantization



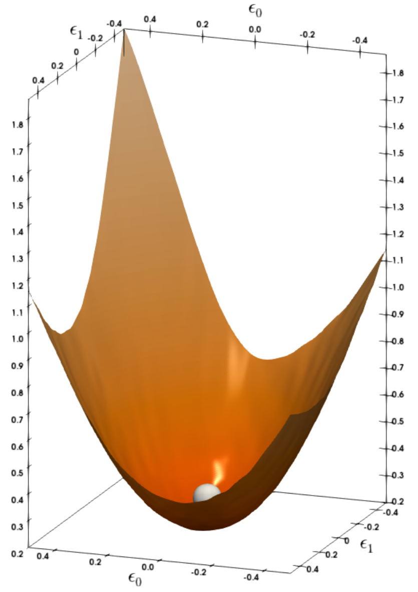
- BERT based models have become de-facto architecture for NLP tasks
  - BERT-base: 12 Layer, 12 Heads, 768 Hidden Dim (110M param, 415MB)
  - BERT-large: 24 Layer, 16 Heads, 1024 Hidden Dim (340M param, 1297MB)

Devlin, Jacob, et al. "BERT: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

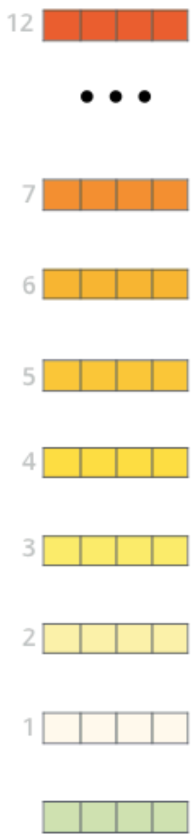
Q-BERT: : Hessian Based Ultra Low Precision Quantization of BERT-- Sheng Shen (AAAI 2020)



10th Layer



4th Layer



(b) MNLI

Method	w-bits	e-bits	Acc m	Acc mm	Size	Size w/o-e
Baseline	32	32	84.00	84.40	415.4	324.5
Q-BERT	8	8	83.91	83.83	103.9	81.2
DirectQ	4	8	76.69	77.00	63.4	40.6
Q-BERT	4	8	**83.89**	**84.17**	63.4	40.6
DirectQ	3	8	70.27	70.89	53.2	30.5
Q-BERT	3	8	**83.41**	**83.83**	53.2	30.5
Q-BERT$_{MP}$	2/4 MP	8	**83.51**	**83.55**	53.2	30.5
DirectQ	2	8	53.29	53.32	43.1	20.4
Q-BERT	2	8	**76.56**	**77.02**	43.1	20.4
Q-BERT$_{MP}$	2/3 MP	8	**81.75**	**82.29**	**46.1**	**23.4**

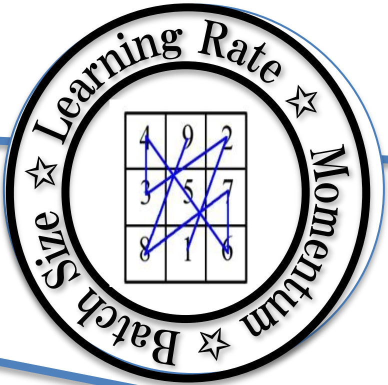
S. Sheng, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT. To Appear AAAI 2020

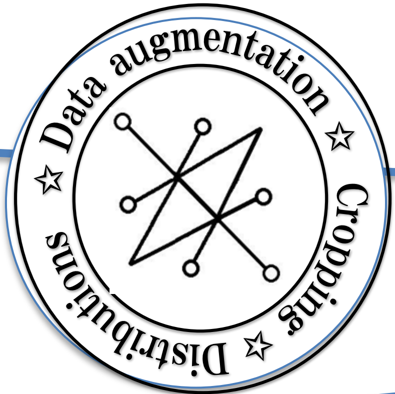
Berkeley DeepDrive

An Integrated Approach to DNN Design Has Four Key Aspects

BAIR

Training
- Momentum,
- Learning rate
- Batch size





Training Data
- Data Augmentation
- Cropping
- Data distribution

DNN Design
- # Layers and types
- Residuals

$$\min_{a \in \mathcal{A}} \min_{\boldsymbol{w}_a} \mathcal{L}(a, \boldsymbol{w}_a)$$

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

Efficient Implementation
- Pruning
- Quantization

Extras