Tutorial on Optimizing Machine Learning for Hardware

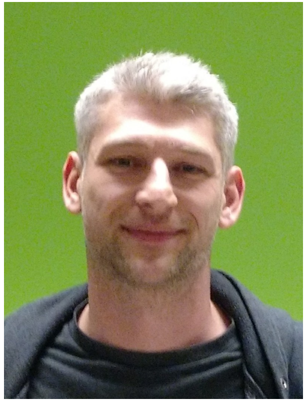
Prof. Warren Gross and Prof. Brett H. Meyer
Electrical and Computer Engineering
McGill University

At EPEPS 2019, October 6, 2019

More Acknowledgments

Adam Cavatassi

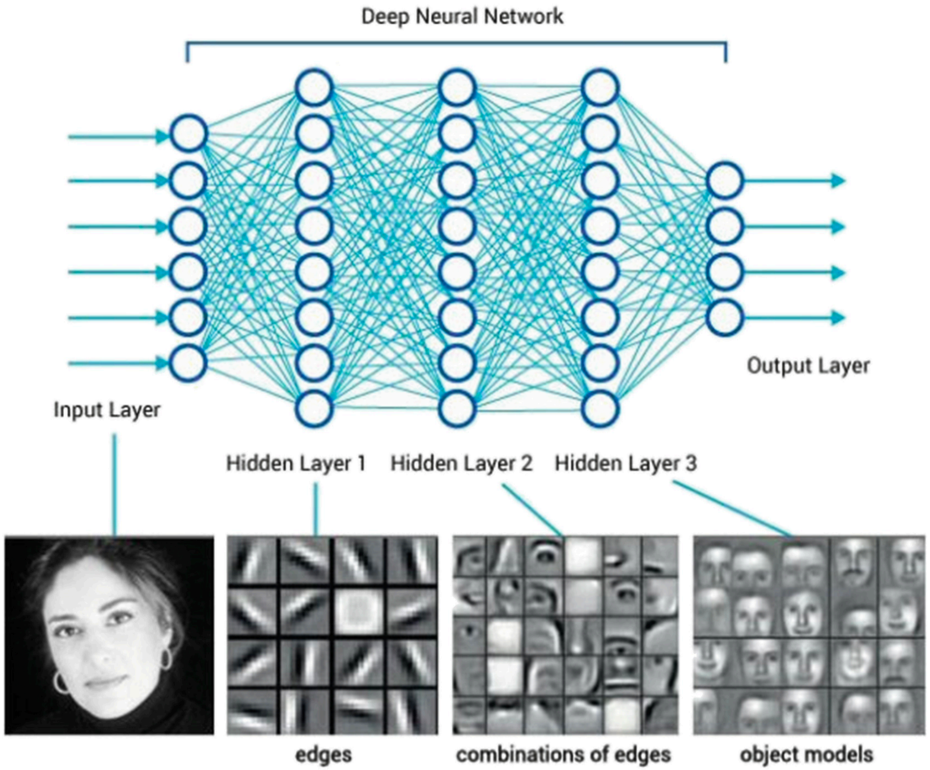
Adithya Lakshminarayanan



Sean Smithson

Recall: Deep Learning is Complex!

- Deep learning automates *feature extraction*
- DNN therefore
  - Have many weights
  - Rely on much data
  - Require lots of training
- What does this imply for deployment?



A Deeper Understanding of Deep Learning

Cloud Deployment

- Computational resources are abundant
  - GPGPUs with specialized, parallel, hardware
- GTX Titan Z
  - 5760 CUDA threads @ 705 MHz w/ 12 GB DDR5 RAM, and 672 GB/s
  - 700 W!!!

Cloud Deployment

- In the Cloud, systems are historically optimized for accuracy alone
  – Throughput is another key metric
- That isn't to say there aren't problems …
  – Model size, training time, training cost, inference delay, can still be issues

Elliot Turner
@eturner303

Holy crap: It costs $245,000 to train the XLNet model (the one that's beating BERT on NLP tasks..512 TPU v3 chips * 2.5 days * $8 a TPU) - arxiv.org/abs/1906.08237

XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang*[1], Zihang Dai*[12], Yiming Yang[1], Jaime Carbonell[1], Ruslan Salakhutdinov[1], Quoc V. Le[2]
[1]Carnegie Mellon University, [2]Google Brain
{zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

Artificial Intelligence / Machine Learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

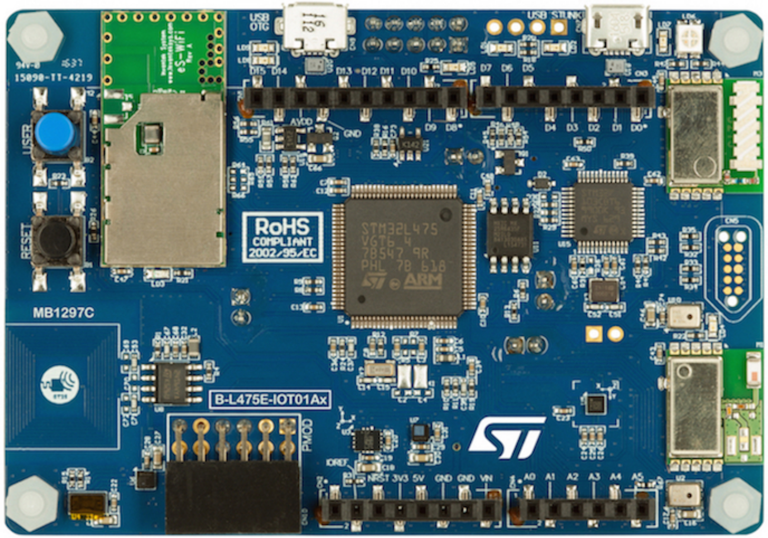
Deep learning has a terrible carbon footprint.

by **Karen Hao** Jun 6, 2019

MIT Technology Review

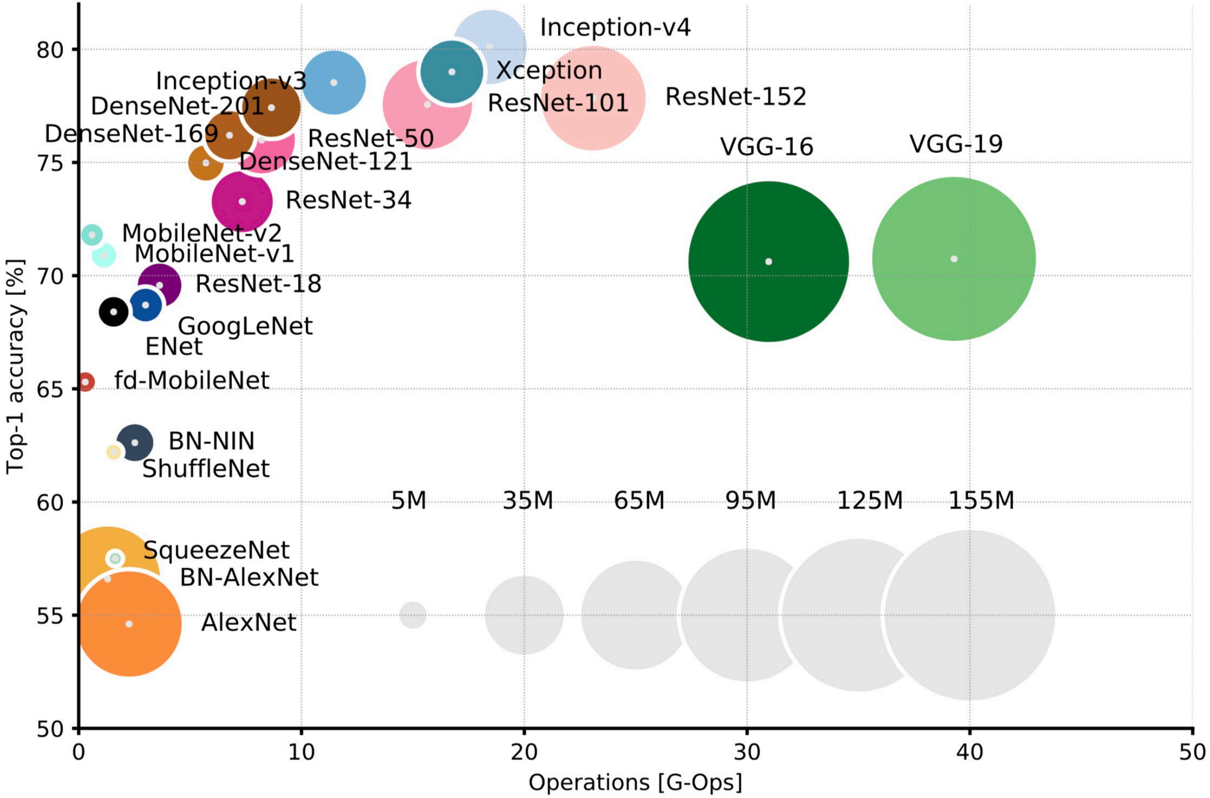
Edge and IoT Deployment

- Computational resources are limited, in comparison
  - IoT devices are often low-power, low-cost microcontrollers
- STM32L4 @ 80MHz w/ 128K SRAM, and FPU
  - 30 mW!
- Systems must be optimized for a variety of metrics
  - Memory footprint
  - Real-time systems: inference latency
  - Mobile and ultra-low-power systems: inference energy



DNN Complexity and Accuracy

Canziani, Paszke, and Culuriello, https://arxiv.org/abs/1605.07678

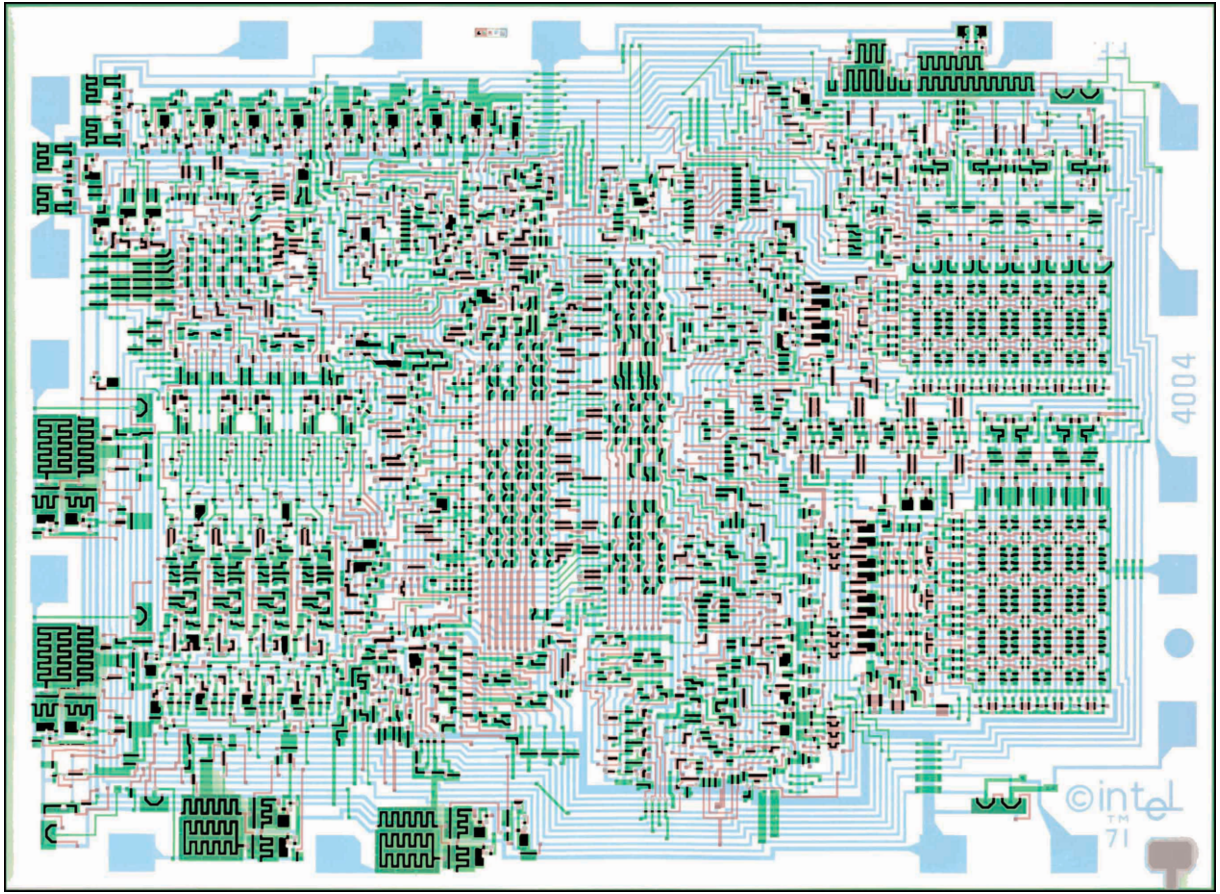


DNN Design? It's Complicated.

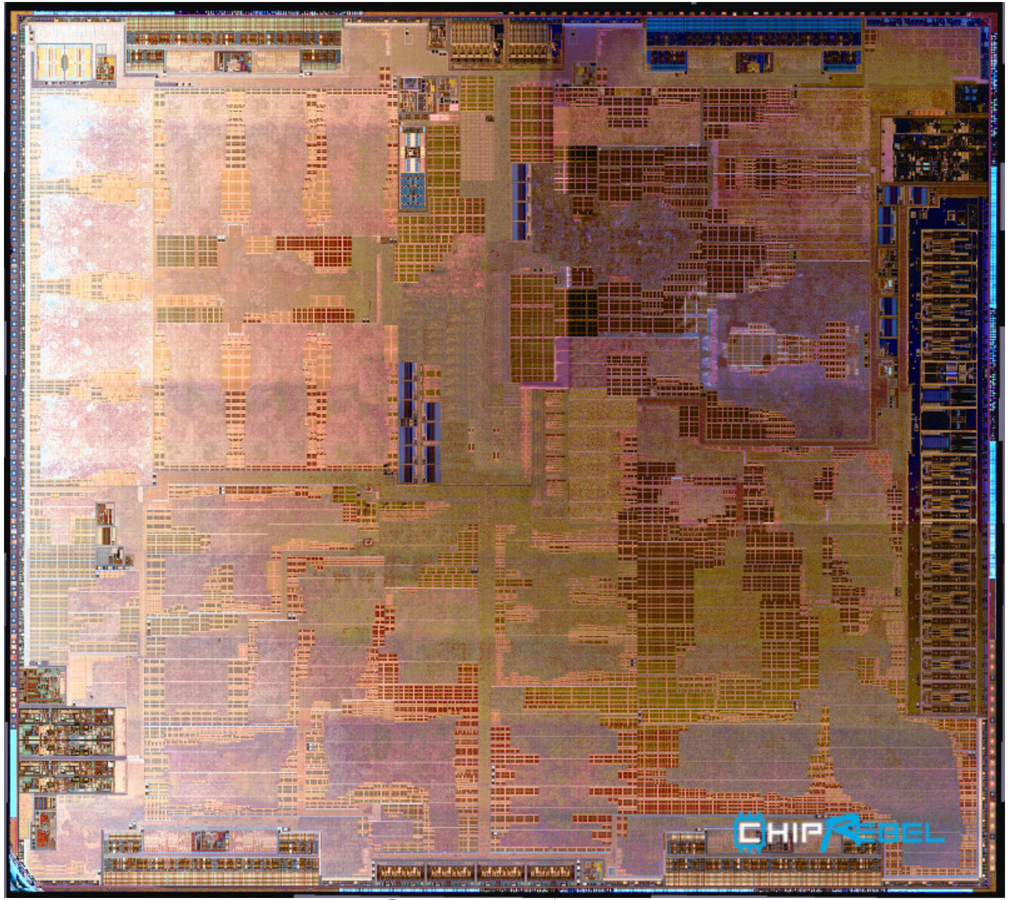
- How is such complexity coped with today?
  - Manual design and optimization!
  - Warehouse-scale computers
  - Adaptation of large networks to small problems
    - Fine-tuning
    - Weight pruning
    - Quantization

Has such complexity been overcome before?

Intel 4004: 2,300 Transistors in ‘71



Huawei Kirin 980: 6.9B transistors in ‘18



From the 4004 to the Kirin 980

- Transistor and circuit models **CUDA**
- Hardware description languages **TensorFlow**
- Performance, power, and cost models **Ops, weights, arithmetic intensity**
- System-level abstractions **Keras**
- Algorithms to automate lower-level design **AutoML**

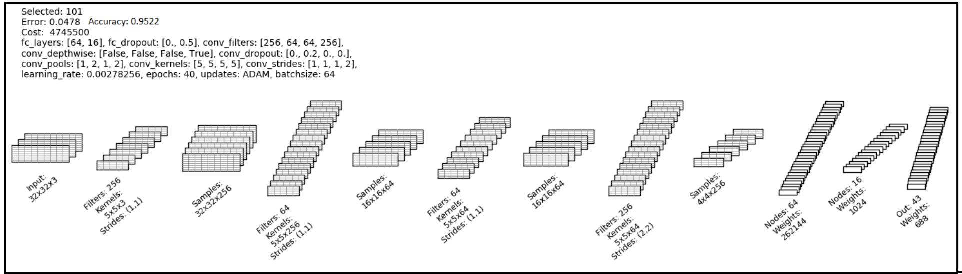
What parallels exist in machine learning?

Hyperparameter Optimization

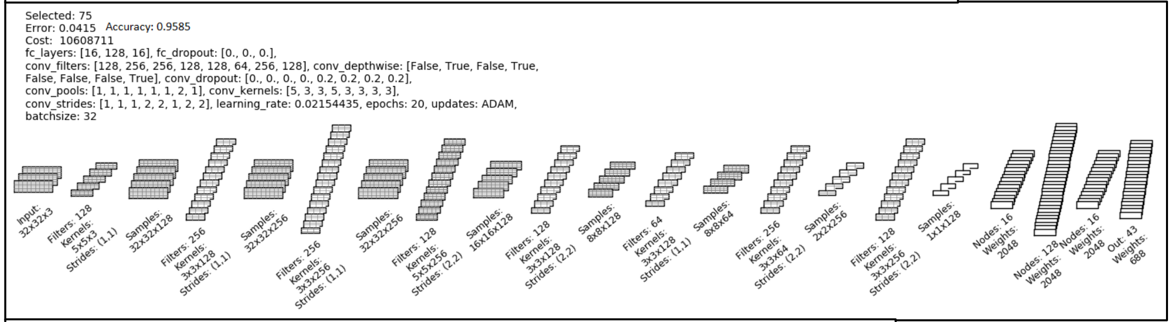
- Introduction to Architecture Search
  - Convolutional neural networks
  - Quantization
- Optimization for IoT devices
  - Quantization
  - Memory footprint optimization

Architecture Search is Difficult

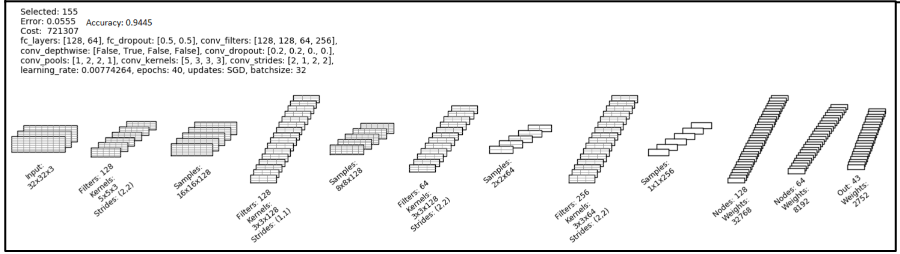
Inference Energy



1

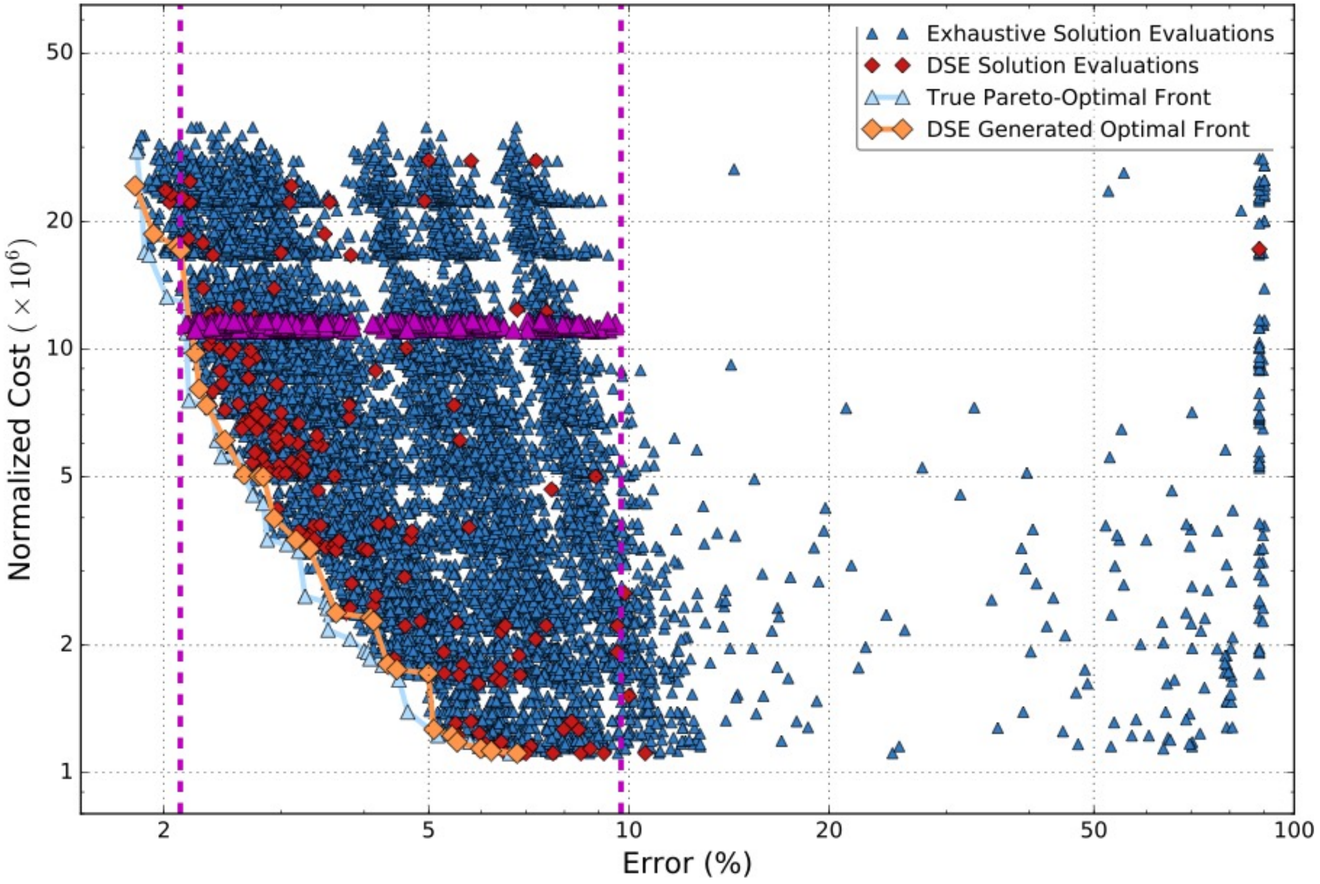


2.2



0.15

Architecture Search is VERY Difficult



So Many Hyper-parameters, So Little Time

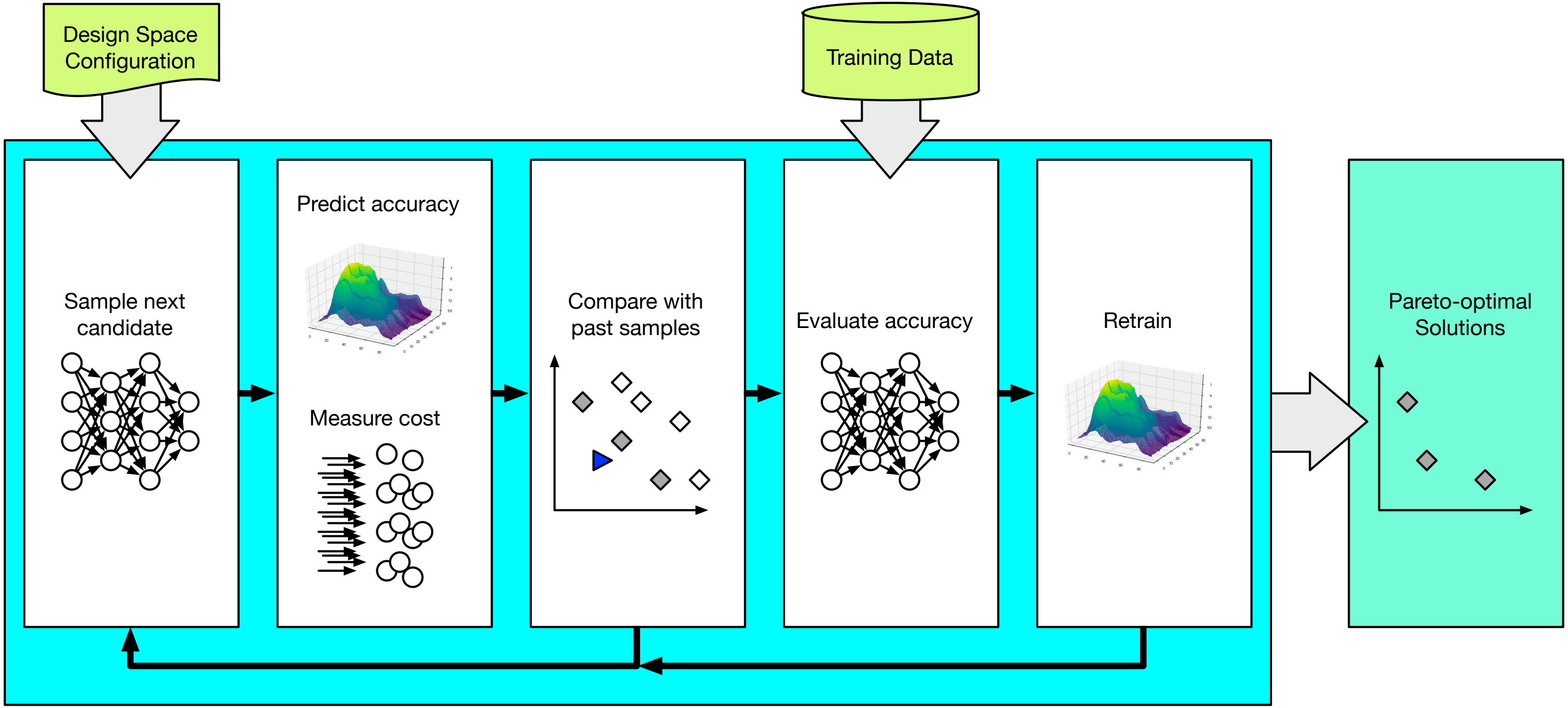
- Artificial neural networks are appearing everywhere, supporting diverse applications
  - Embedded and mobile devices
  - In the cloud, and at the edge of the IoT
  - *Different domains have different constraints*
- Hyper-parameter selection affects performance (*accuracy*) and cost (e.g., *energy* or *delay*)
  - E.g., number of layers, types of neurons, etc.
- But, no intuitive patterns in large design spaces

One solution: apply design automation techniques to deep learning

Ordinary People Accelerating Learning

- OPAL models the DNN design space with a many-dimensional *response surface* (hyperplane)
- A meta DNN (*mDNN*) learns which areas of the design space strike interesting trade-offs
  - Iteratively evaluates target DNNs (*tDNN*)
  - Builds a model to predict which *tDNN*
- Returns a near-Pareto-optimal set
  - E.g., from *high accuracy*, **high cost**, to **low accuracy**, *low cost*, and everything in between

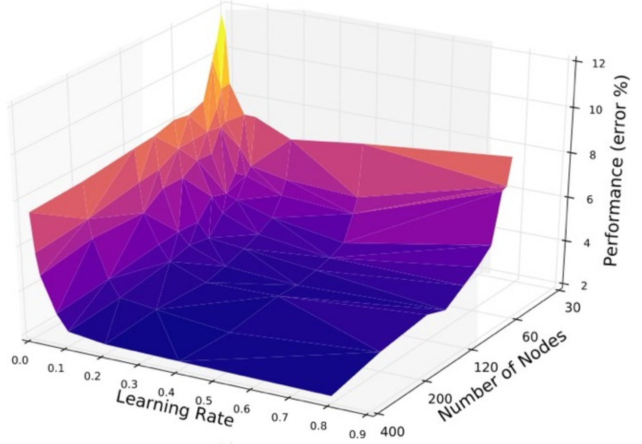
Ordinary People Accelerating Learning

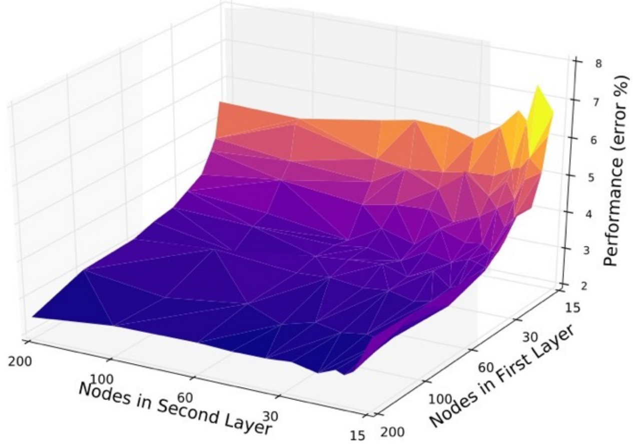


Smithson, Yang, Gross, and Meyer, ICCAD 2016

Response Surface Modeling

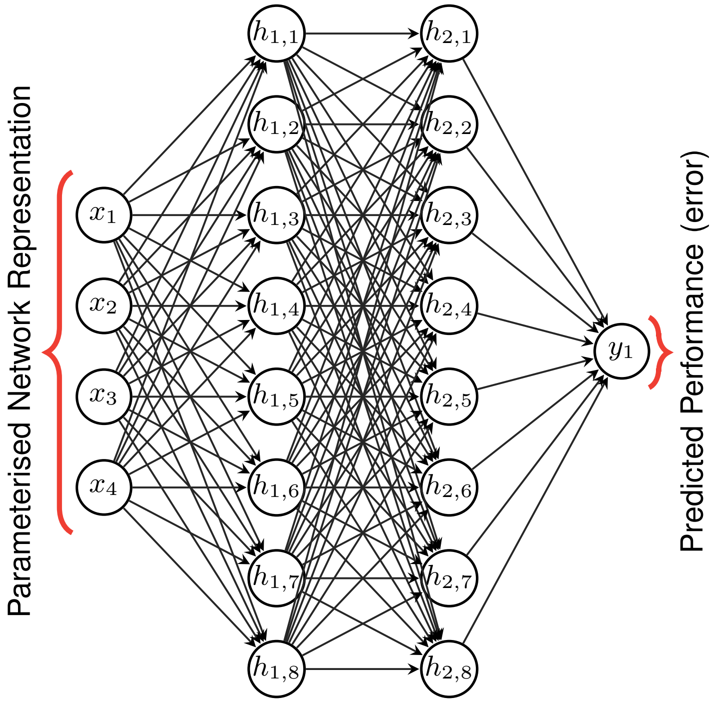
- *mDNN* models *tDNN* performance as a function of hyper-parameters
- Response surface is fit to evaluation data
- *tDNN* evaluation is **slow**, *mDNN* estimation is *fast*





Performance Modeling: *mDNN*

- Surface modelled with two hidden layers
- Retrained after each new solution is evaluated
- Little training data needed for prediction of *tDNN* error



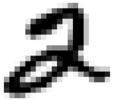
Actual mDNN is larger; smaller layers shown for visualization only

Cost Modeling

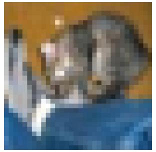
- There are several bad options for cost metrics
  - MACs, or weights, or parameters
  - These are not predictive of performance
- There are many good options for cost metrics
  - Inference delay, or inference energy
  - Arithmetic intensity
  - Memory footprint
- For now, we use *inference energy*
  - A weighted sum of MACs and memory accesses (about 100:1)

Experimental Setup

- How well does automatic search perform?
- Evaluated with image recognition benchmarks:
  - MNIST: grayscale images of handwritten digits

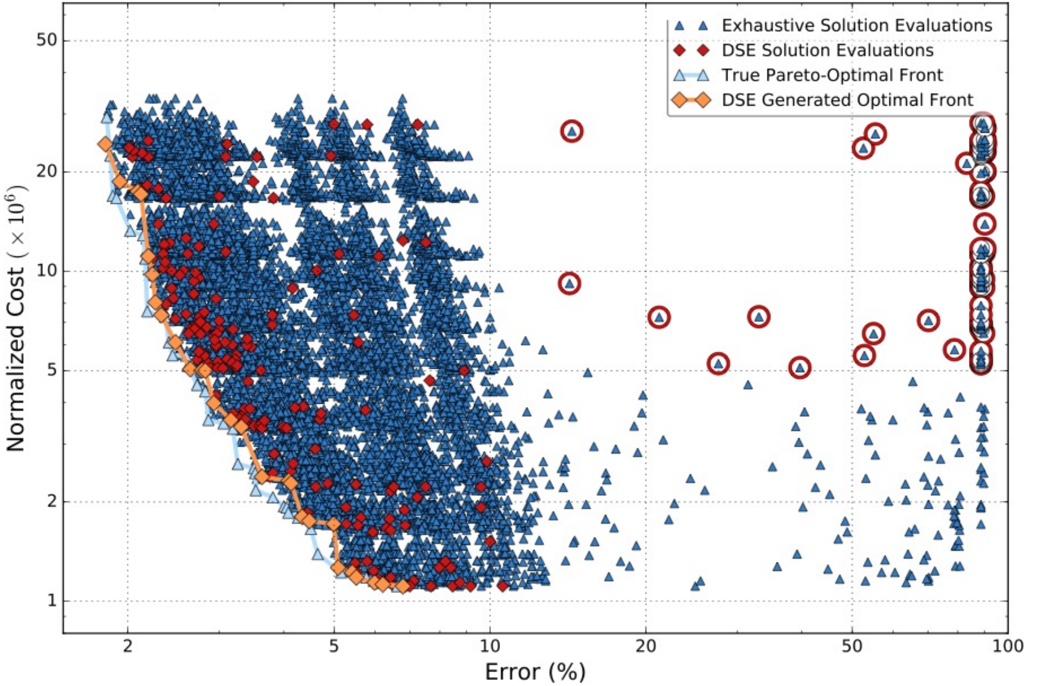




  - CIFAR-10: RGB color images, different classes



- Evaluated designing:
  - Fully-connected (FC) multi-layer perceptrons (MLPs)
  - Convolutional neural networks (CNNs)

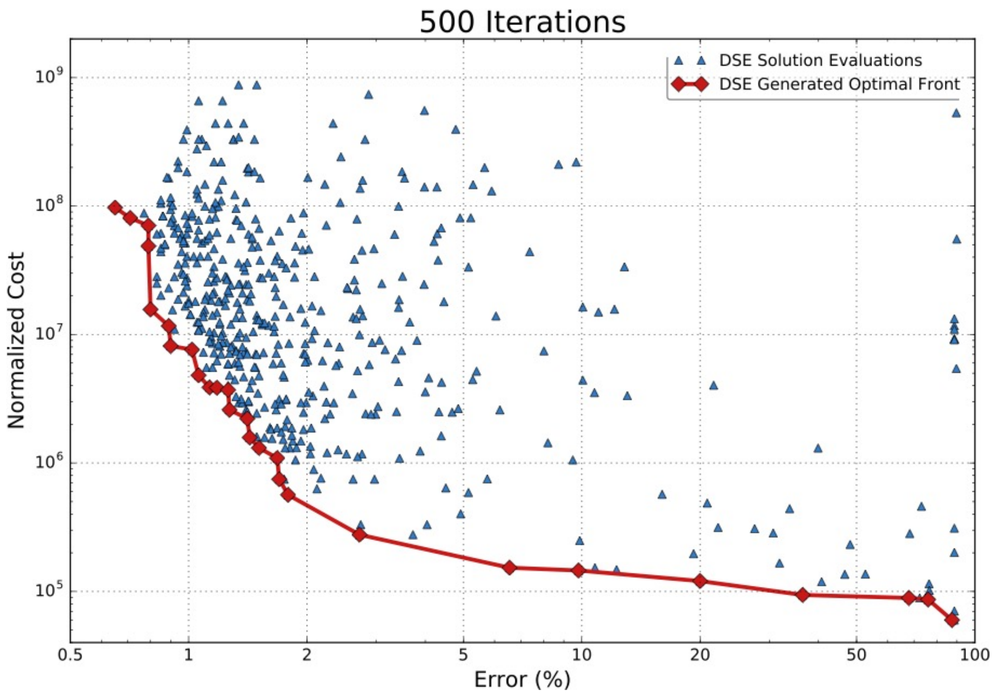
Exhaustive Search vs DSE Results



- *Majority of explored points* are near the Pareto-optimal front
- Many fewer *objectively bad* solution are evaluated

DSE: CNN on MNIST

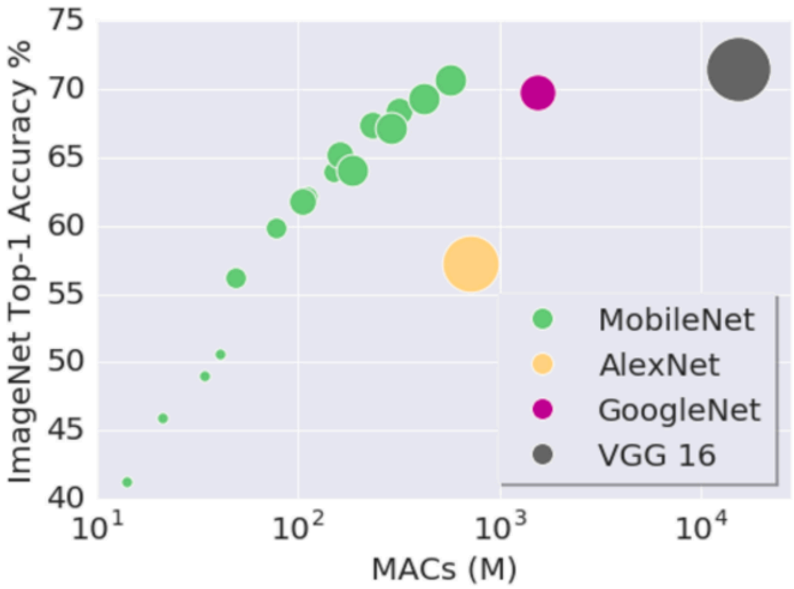
- Design space has over 10^7 configurations



- 1-2 CNN layers
- 8-128 filters per CNN
- Kernel: 1x1-5x5
- Max-pool: 2x2-4x4
- 1-2 FC layers
- 10-250 nodes per FC
- LR: 0.01-0.8

Experimental Setup

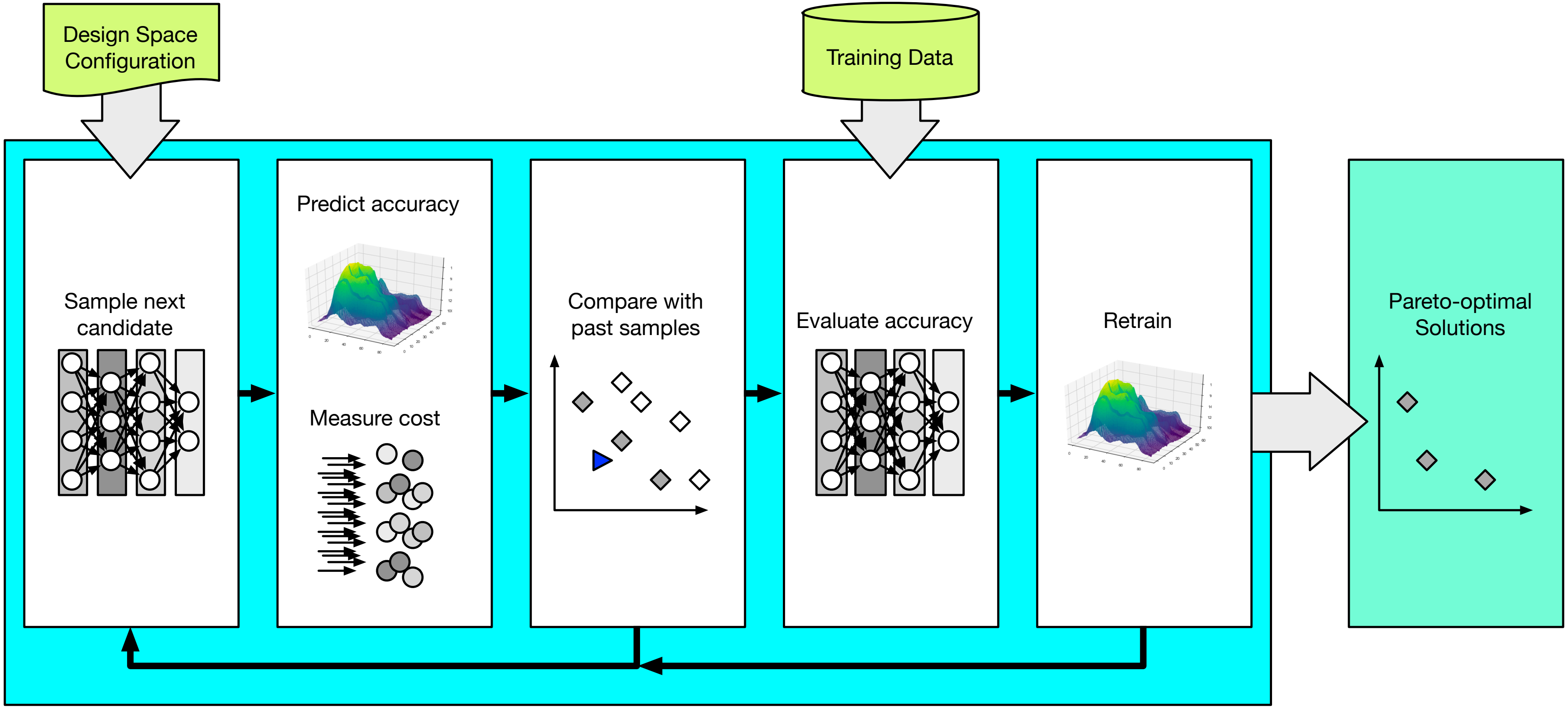
- Can automatic search also effectively consider quantization?
- Evaluated with CIFAR-10
- Evaluated designing CNN
  - Per-layer fixed point, and binary quantization
  - Cost function: inference energy weighted by bit width
- Compared with Google MobileNets



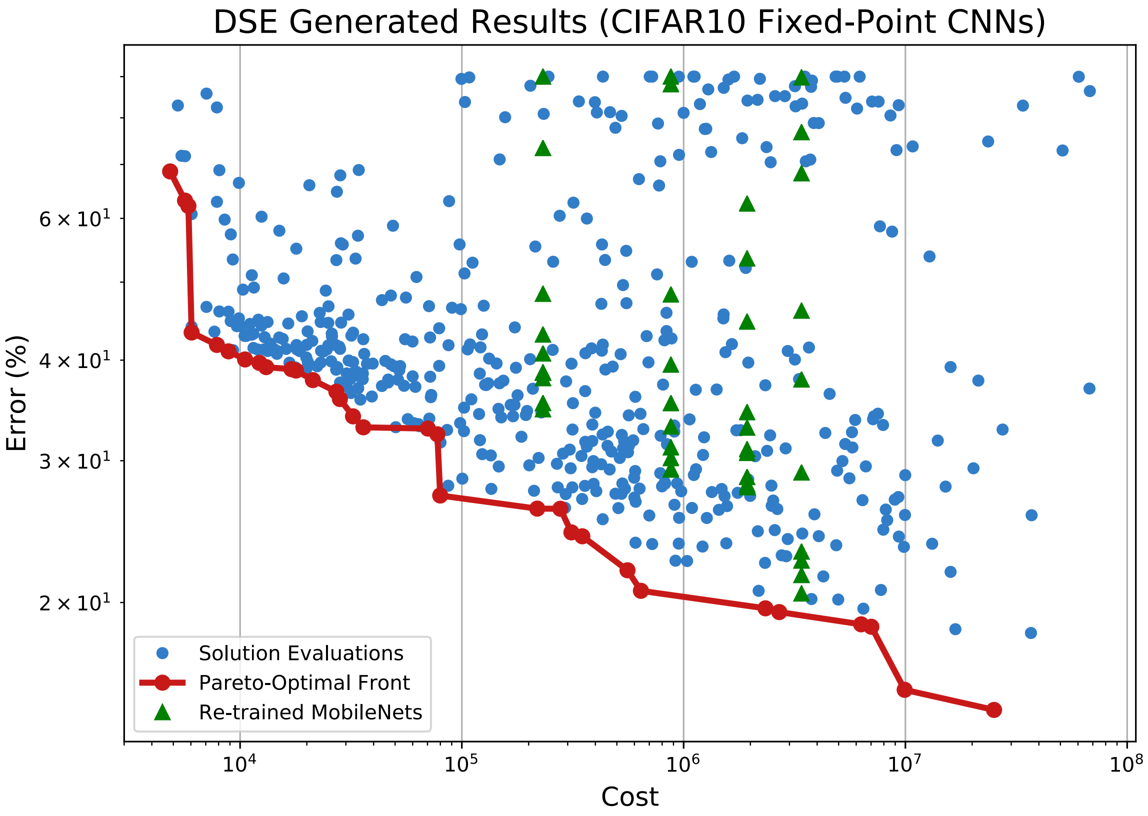
Quantization

- Recall: quantization means not using 32-bit floating point numbers
  - For weights, for activations, etc
- Fixed point quantization is often described in $Q_{m,n}$ notation
  - m bits of integer, n bits of fraction, with $m+n \le N$-1
  - The fewer the bits needed, the lower the complexity (*in theory*)
- Alternatively, weights can be binarized, ternarized, etc

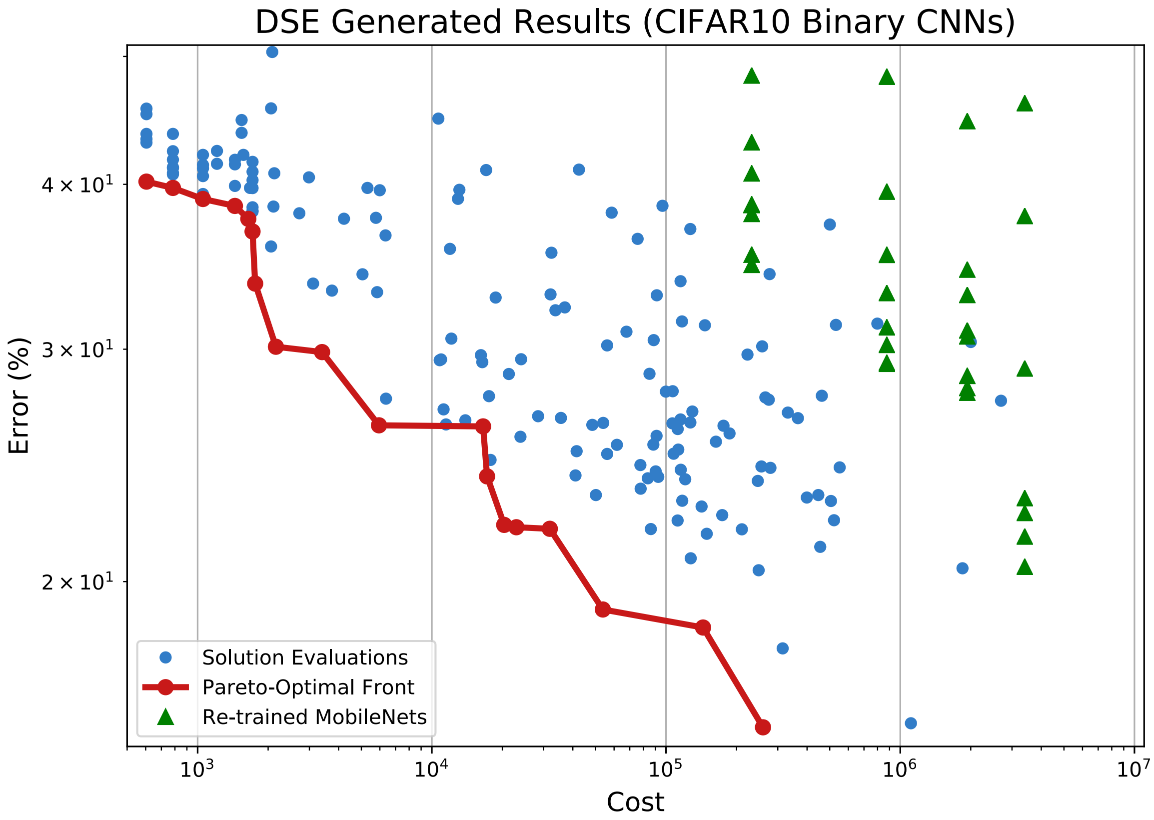
Exploring Quantization



DSE: Fixed-Point CNN on CIFAR-10



DSE: Binary CNN on CIFAR-10



What Makes IoT Deployment Hard?

- Cloud deployment:
  - Keras to TensorFlow to CUDA, and everything works the way you'd expect
  - New, experimental layer? Implement it in Keras, it'll be fine
- IoT deployment:
  - Keras to *depends*
  - Uneven support for *everything*
  - Hardware constraints *limit your options*
  - Multiple, incompatible libraries *for the same processor*

Batch Normalization

- Training in batches can improving training convergence
- Batch normalization manages covariate shift in inputs across the batch of samples
  - Normalizes input features to be in (0, 1]
  - Allows models to better learn and generalize
- A special layer is placed before activations

$$\hat{x}_i = \gamma \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

- This is a standard technique!

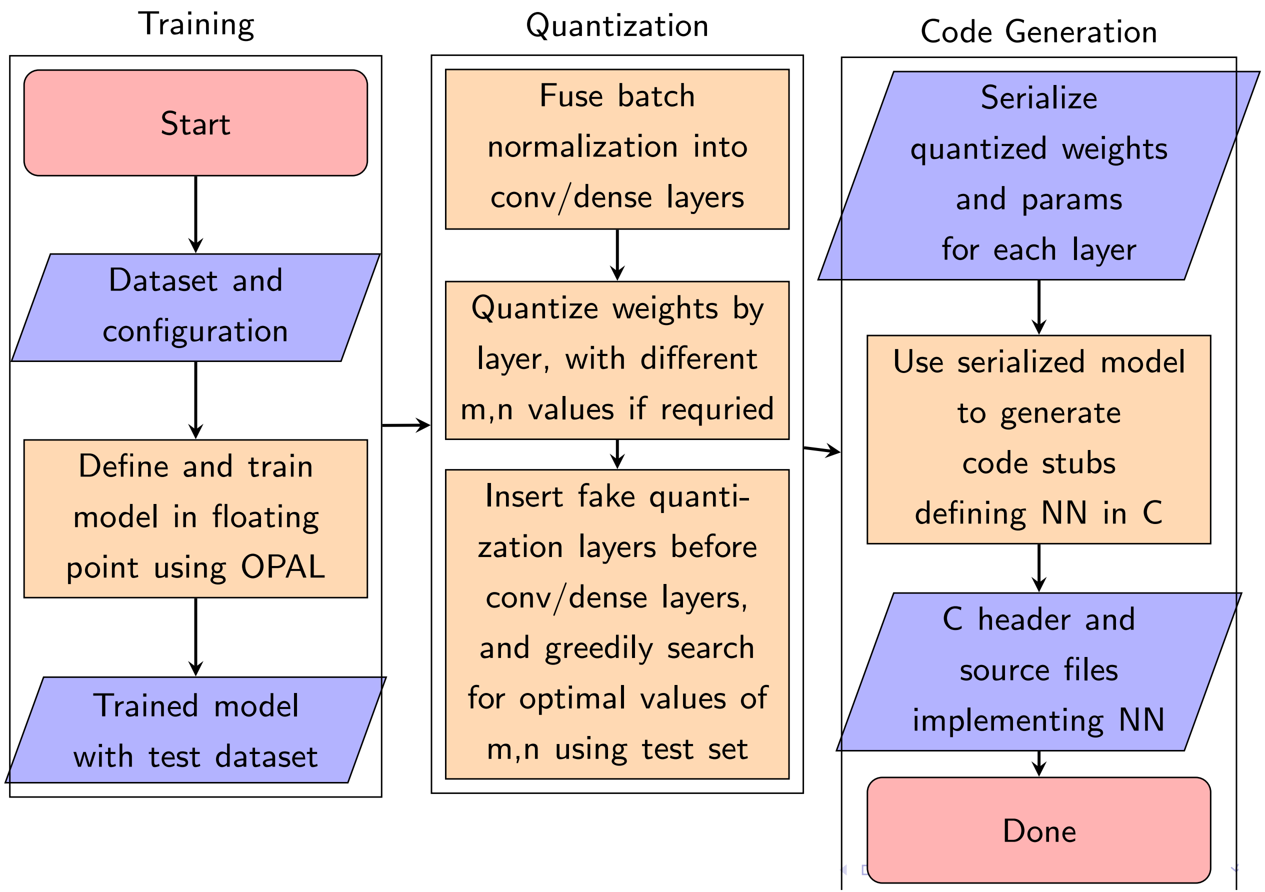
Batch Normalization

- ARM's CMSIS-NN does not support batch normalization
- Instead, batch norm layers must be manually fused with convolutional layers
- Batch normalization is formulated as:

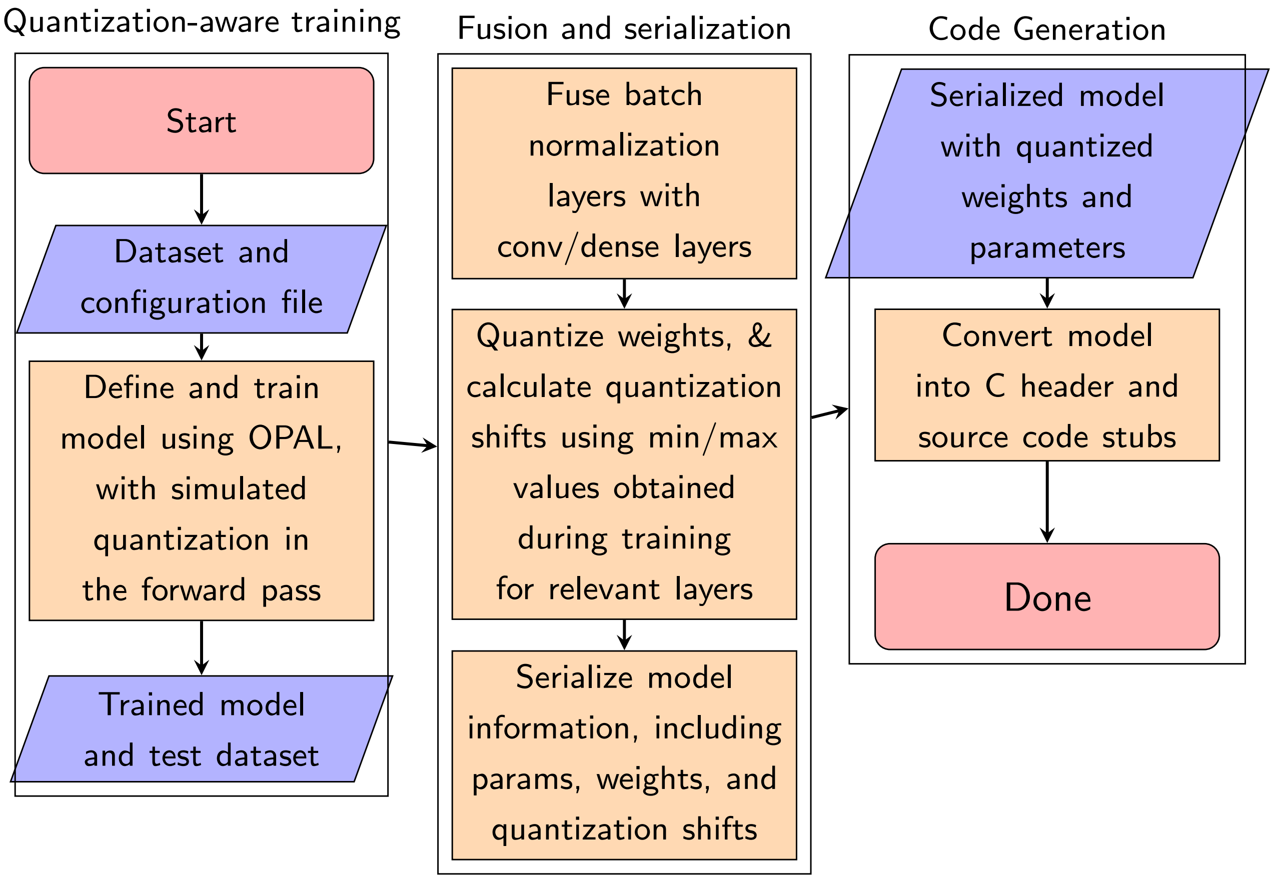
$$\hat{F}_{i,j} = W_{BN} \cdot (W_{conv} \cdot f_{i,j} + b_{conv}) + b_{BN}$$

- This can be combined with a convlutionatl layer if
  – Filter weights are equal to: $W_{BN}\ W_{conv}$
  – And bias weights equal to: $W_{BN}\ b_{conv} + b_{BN}$

Post-training Quantization

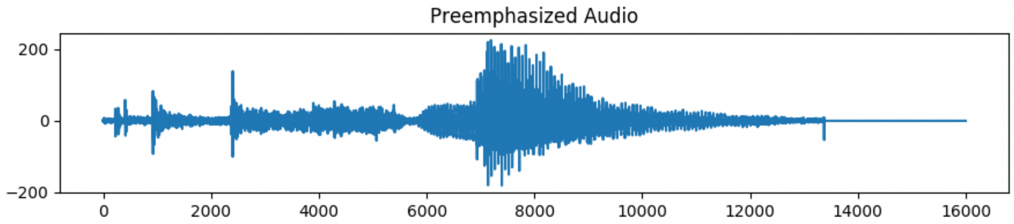


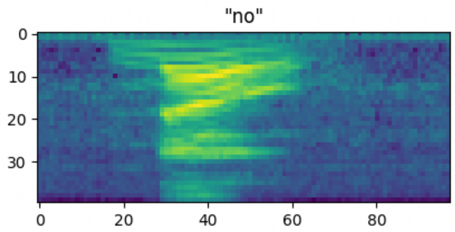
Quantization-aware Training



Experimental Setup

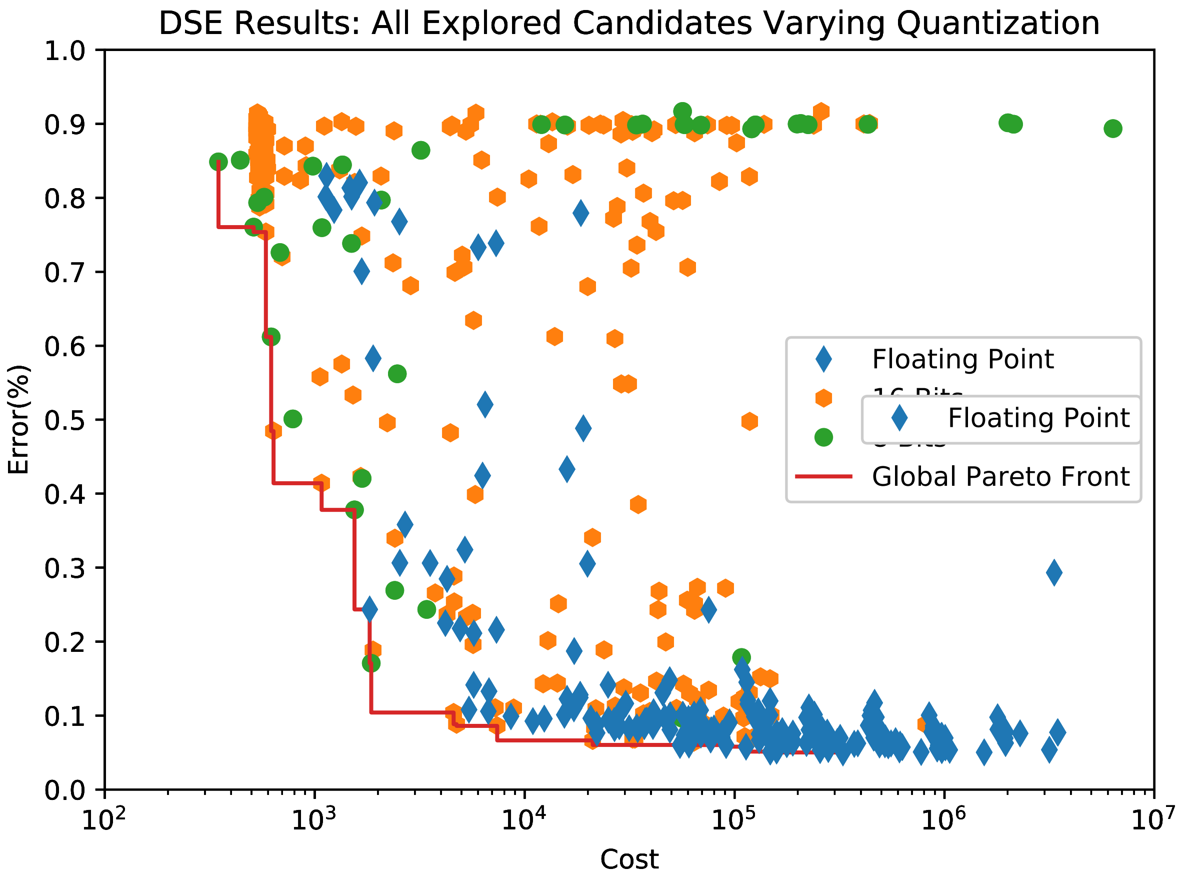
- How do quantized networks compete with FP networks?
- Evaluated with the Google commands dataset:





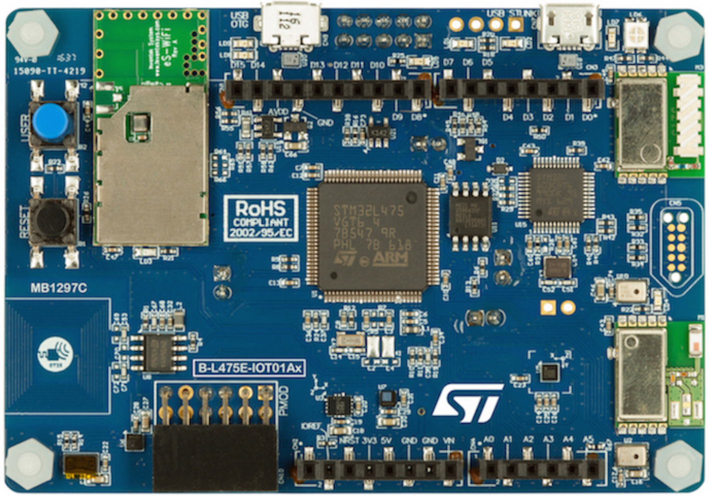
- Evaluated designing CNN, Keras to CMSIS-NN
  – Floating point weights
  – 8- and 16-bit weights, per layer $Q_{m,n}$ formatting
  – Cost function: MACs, weighted by bit width

Quantized vs. Floating-point Weights



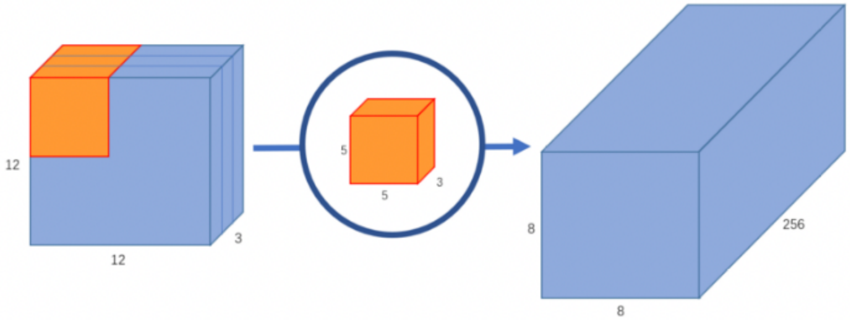
Experimental Setup

- Can we find designs that fit on the STM32L4?
  - Using STM32 Cube.AI to generate optimized C
- Evaluated with the Google commands dataset
- Evaluated designing CNN, Keras to STM32 Cube.AI
  - Floating-point weights
  - Convolution, and depth-wise separable convolution
  - Cost function: memory footprint



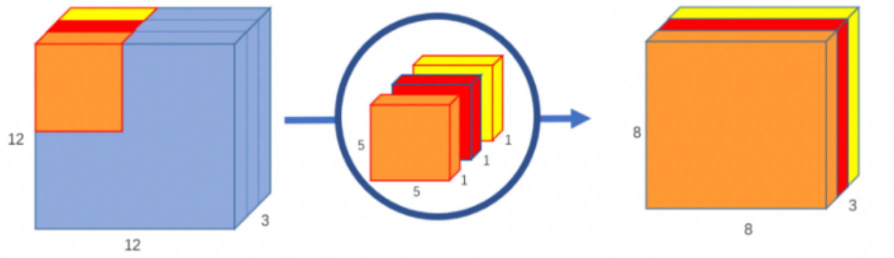
Recall: Convolution is Complex

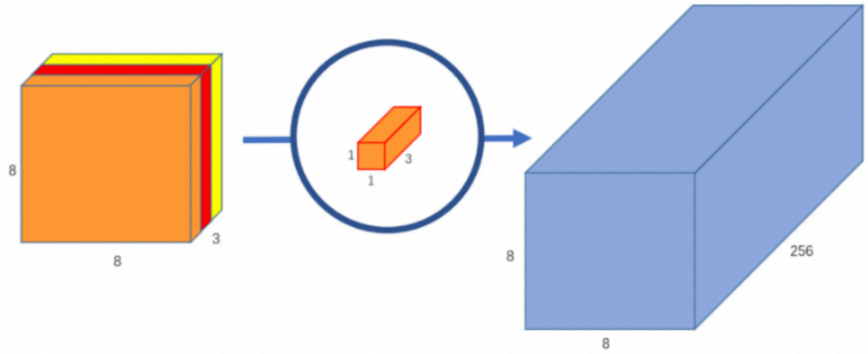
- N input channels
- M output channels, or feature maps
- M sets of N $k \times k$ filters, or kernels, and M bias terms
- This sums to $N\,M\,k^2$ weights



Depth-wise-Separable Convolution

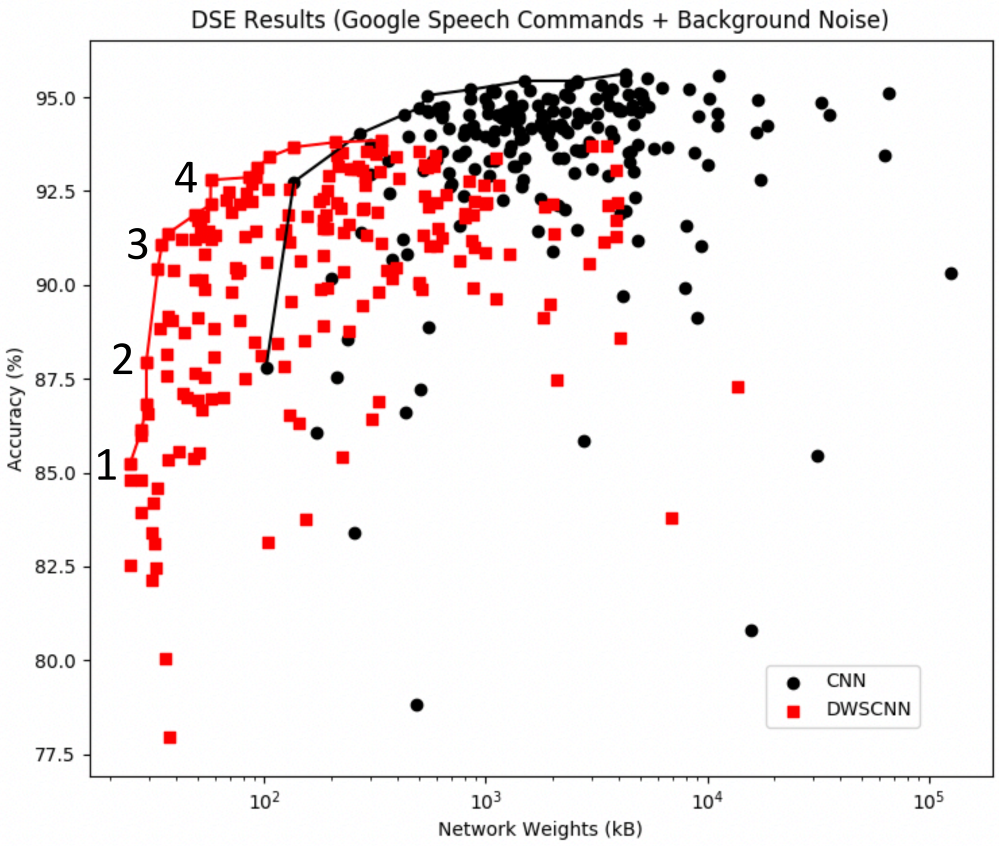
- Transformations can reduce the complexity of convolution
- DWS convolution operation separates convolution into:
  - A depth-wise step, and
  - A point-wise step
- This sums to $N\,(M + k^2)$ weights
- This is employed by MobileNets to reduce model complexity





Memory Footprint Results

#	Acc (%)	Weights	MACs	Weight Mem. (kB)	Activation Mem. (kB)	Latency (ms)
1	84.8	6336	445k	25.54	60.13	107
2	88.8	8672	781k	30.28	60.13	153
3	91.2	10784	1.59M	35.61	245.13	DNF
4	92.8	16791	2.37M	58.92	120.25	DNF



Cavatassi, Gross, and Meyer, tinyML 2019

Conclusions

- Abundant data and compute power is ushering in the era of ubiquitous machine learning
- Efficient deep learning requires
  - Careful hardware design
  - Careful software optimization
- Custom hardware orchestrates data movement, and facilitates model compression
- Architecture search tunes model structure
- *Applications, architectures, and automation must cooperate to unlock the promise of deep learning*