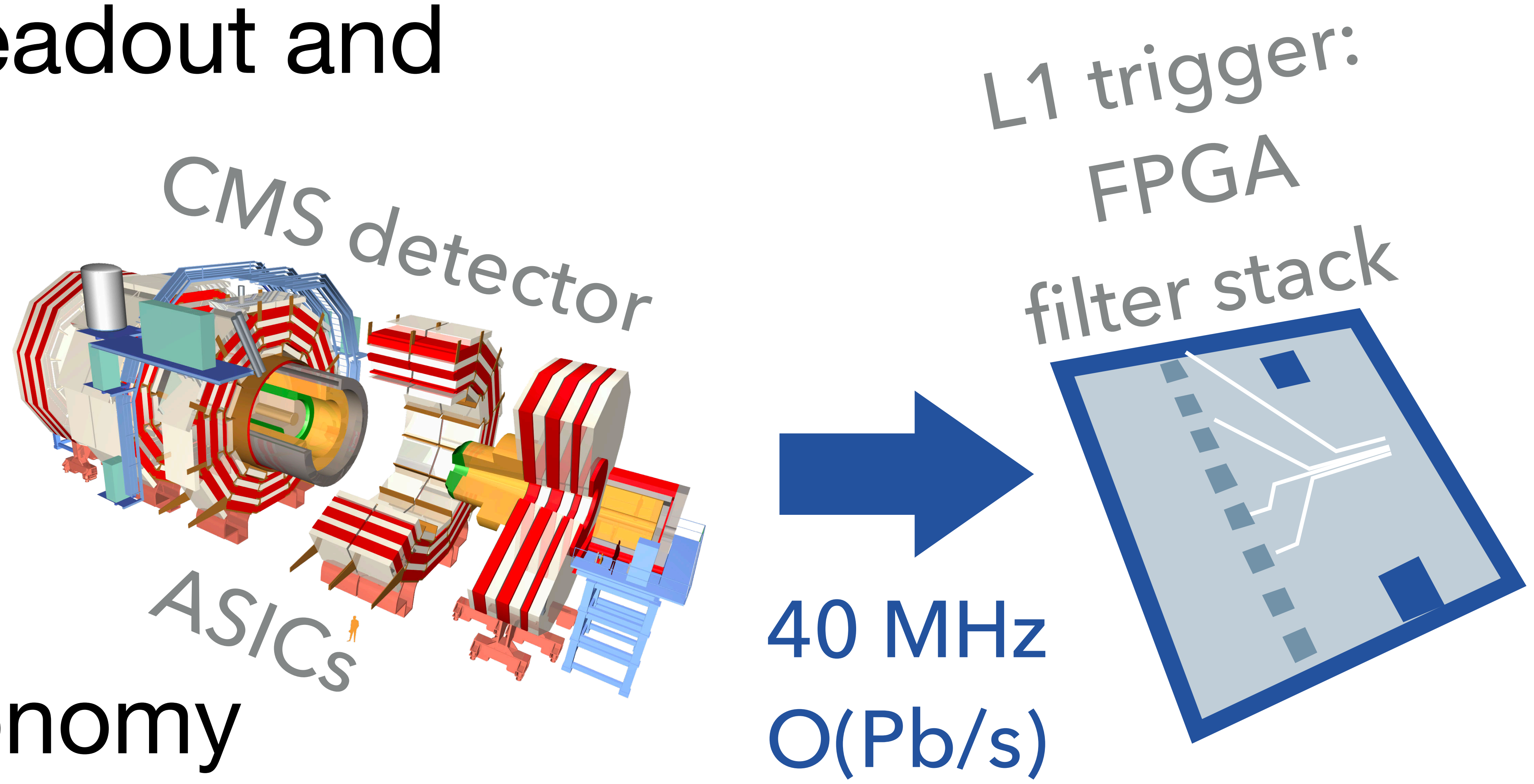
PHYS 139/239: Machine Learning in Physics

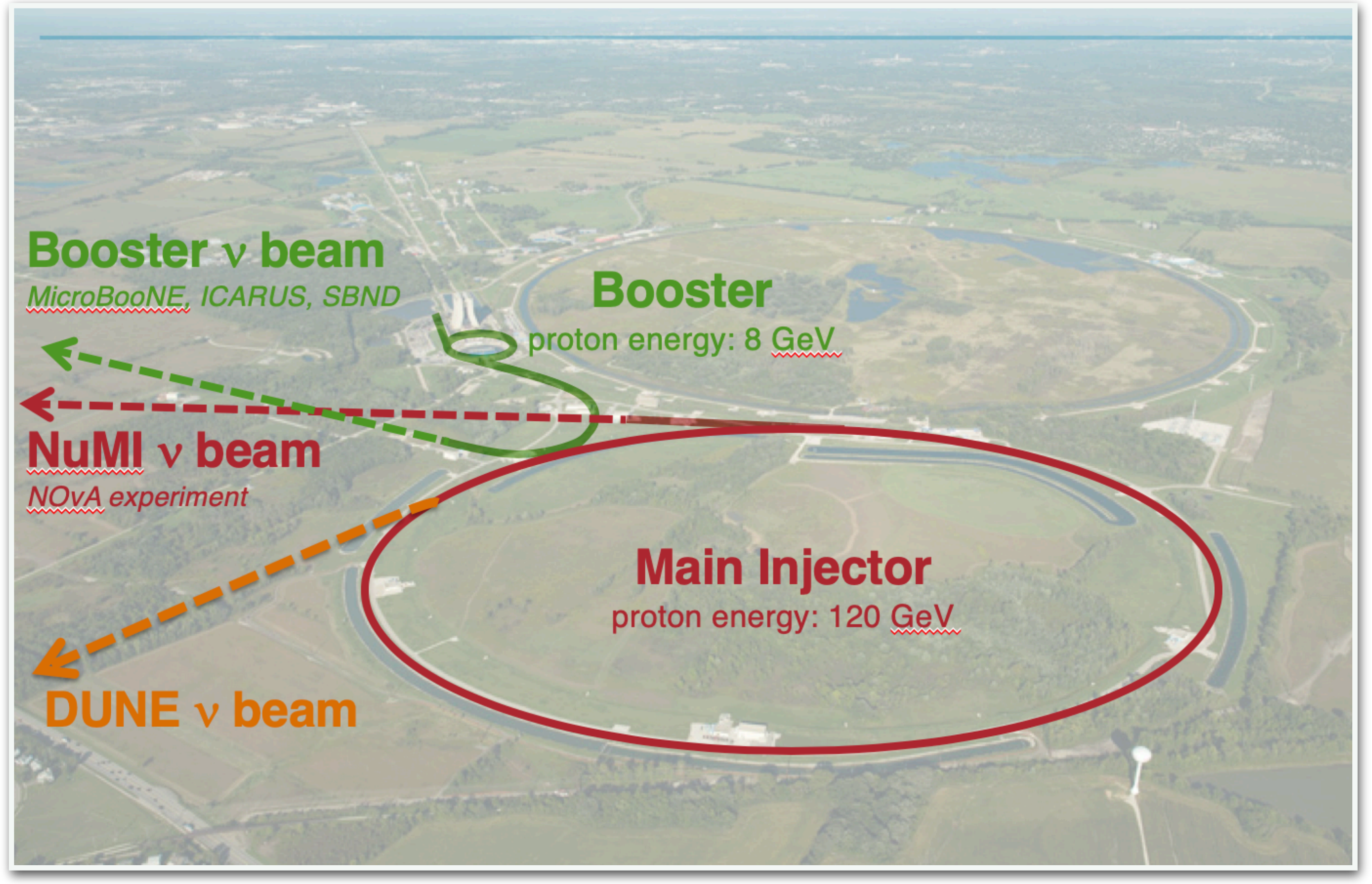
Lecture 15: Model compression

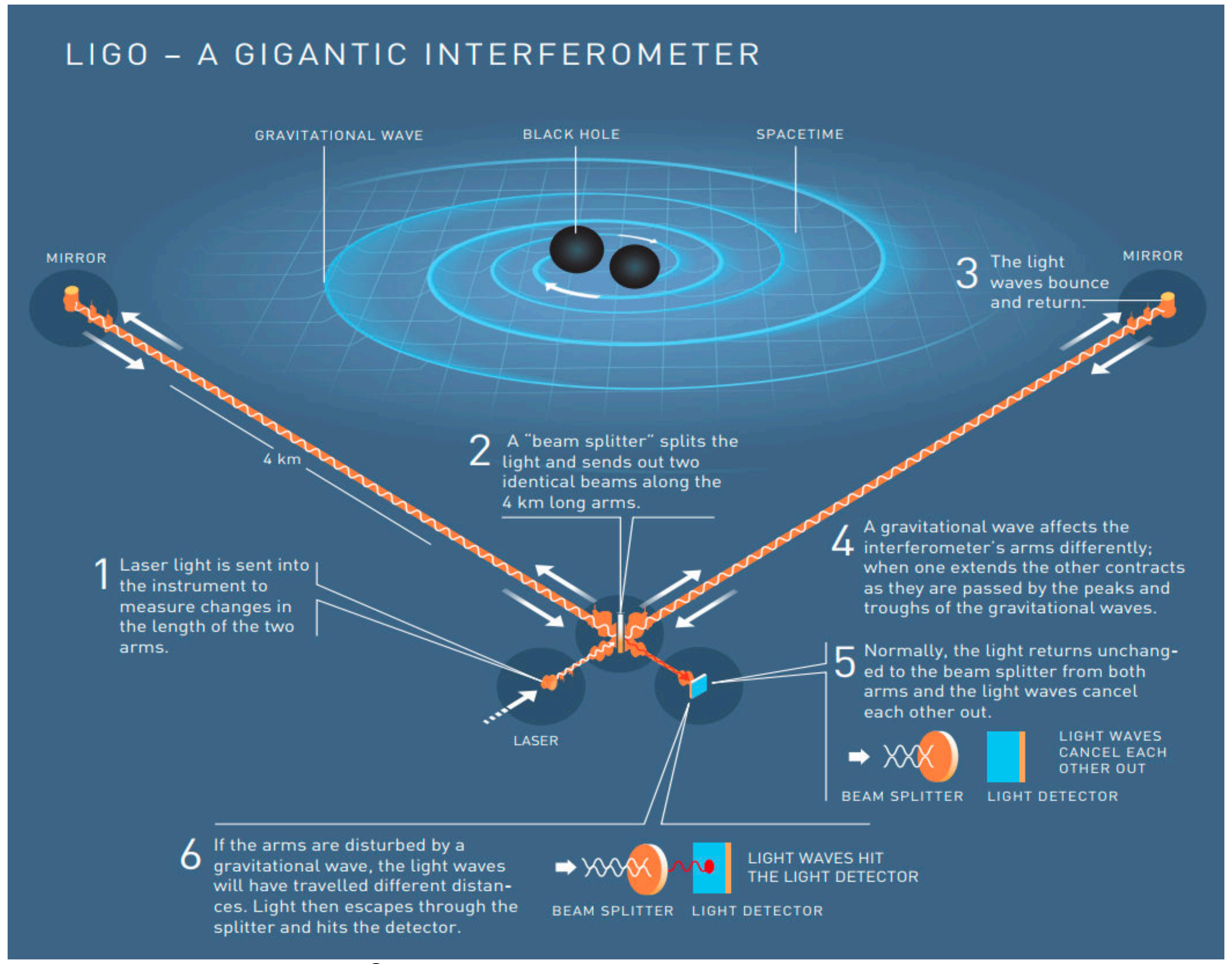
Javier Duarte — February 28, 2023

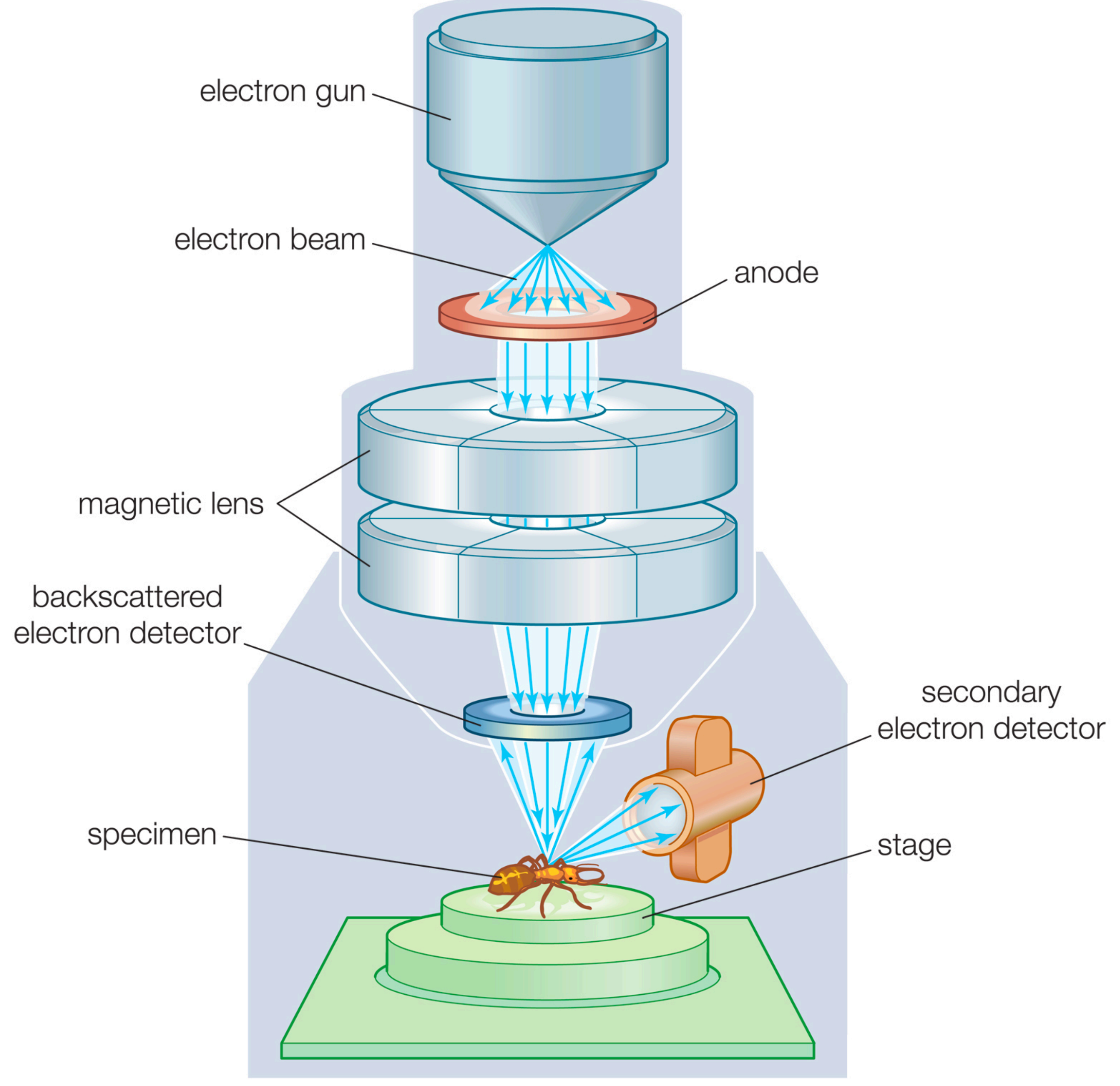
Science use-cases for real-time AI

- Collider on-detector readout and "trigger" at 40 MHz
- Accelerator control
- Neutrino physics
- Multi-messenger astronomy
- Electron & X-ray microscopy

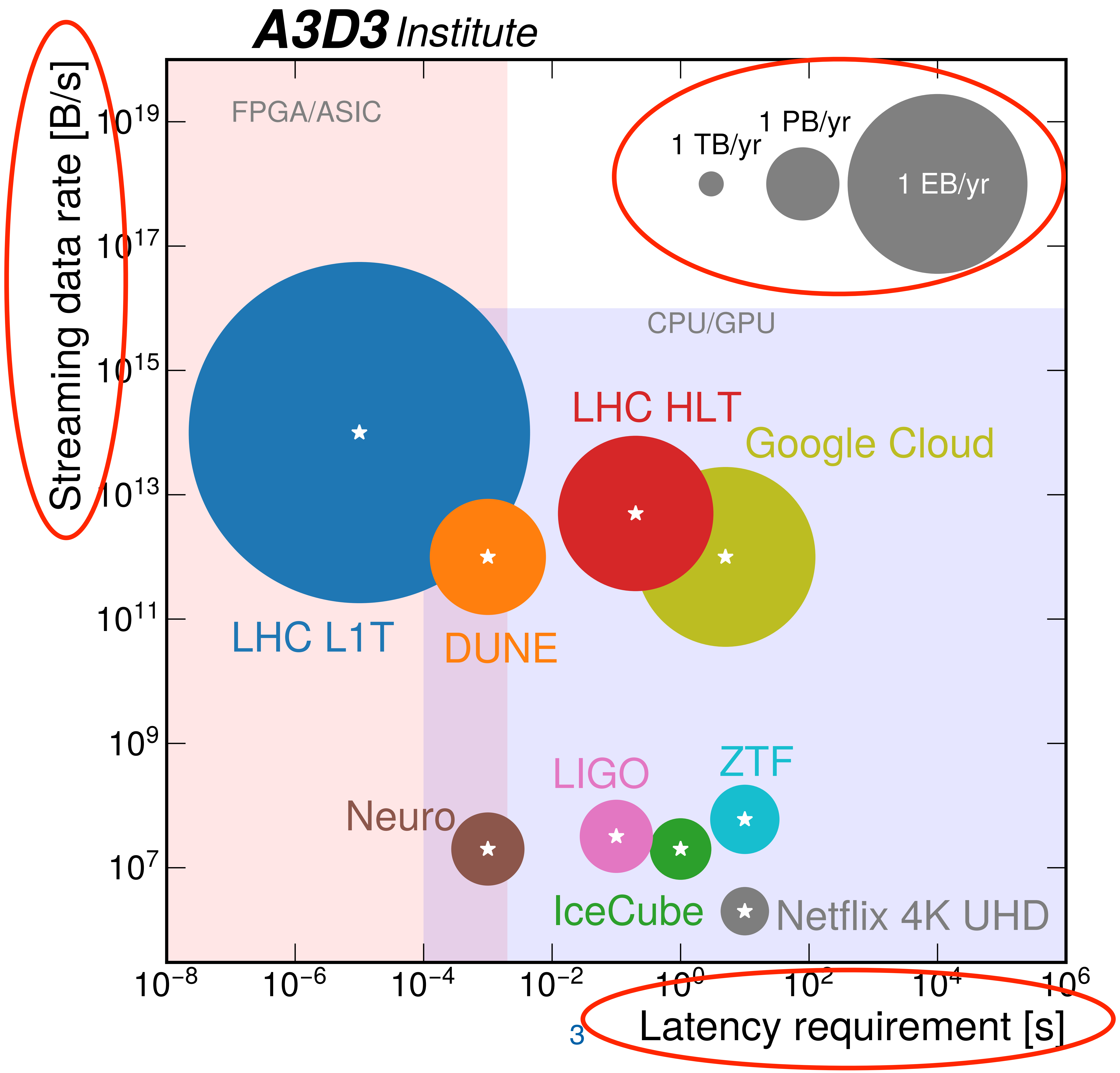






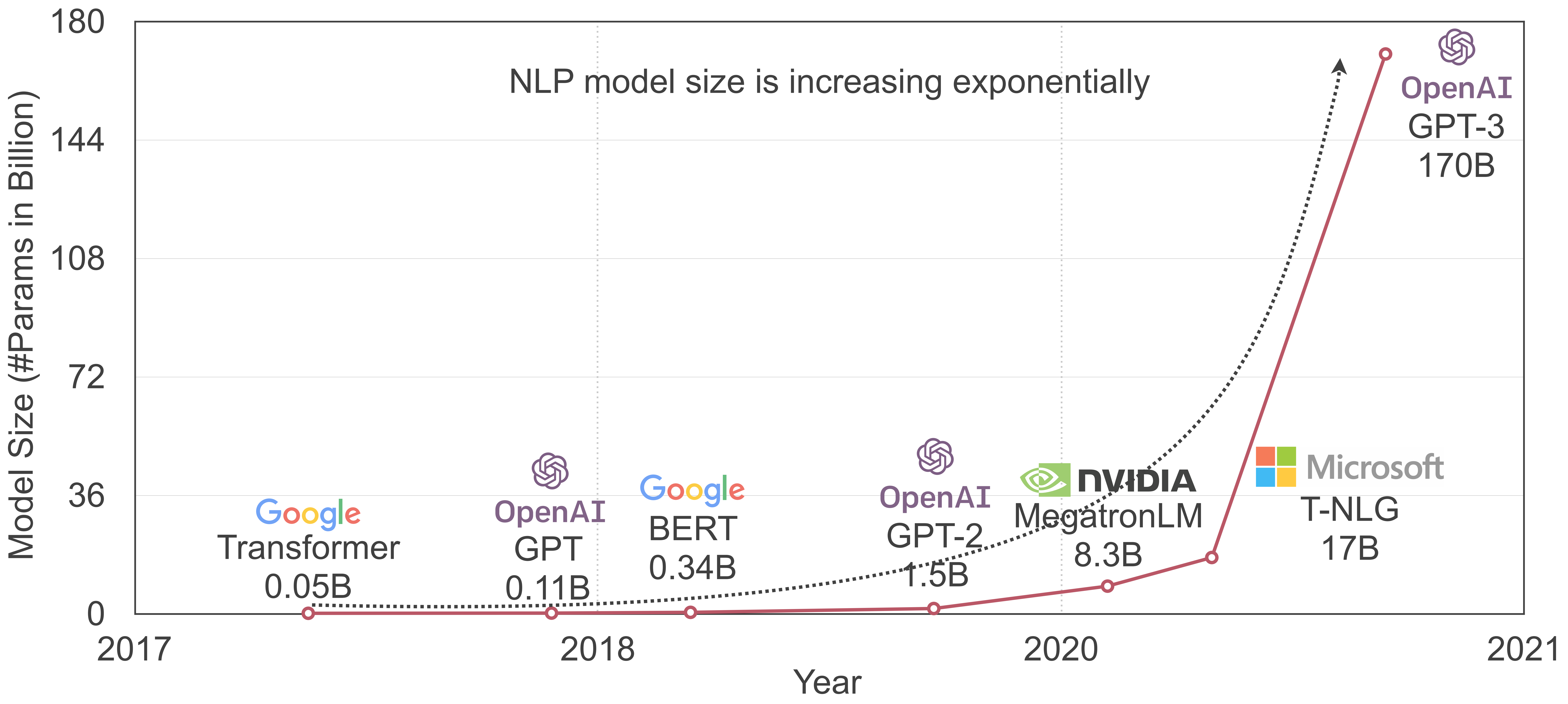


Scientific ML challenges



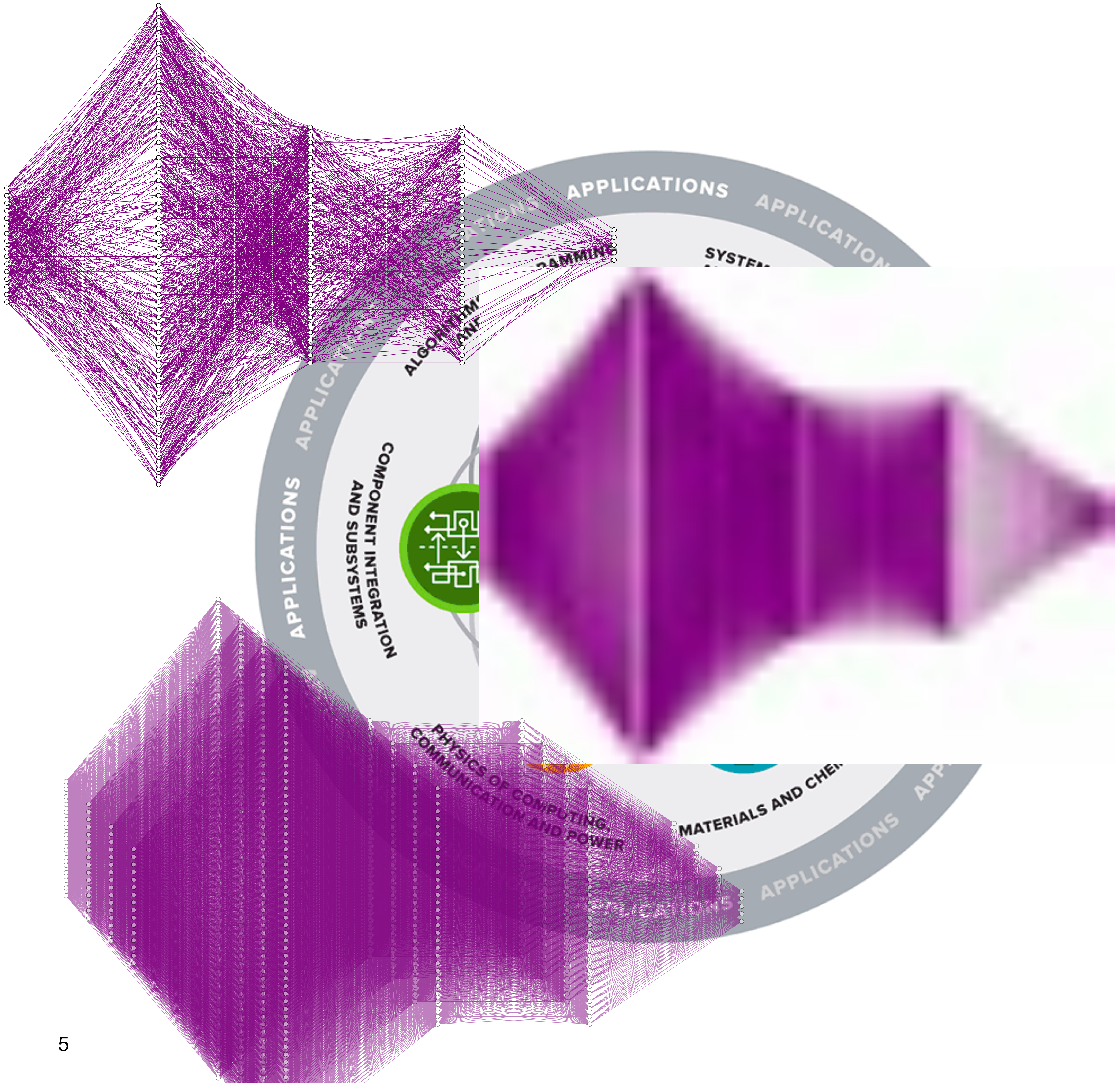
3

AI model sizes



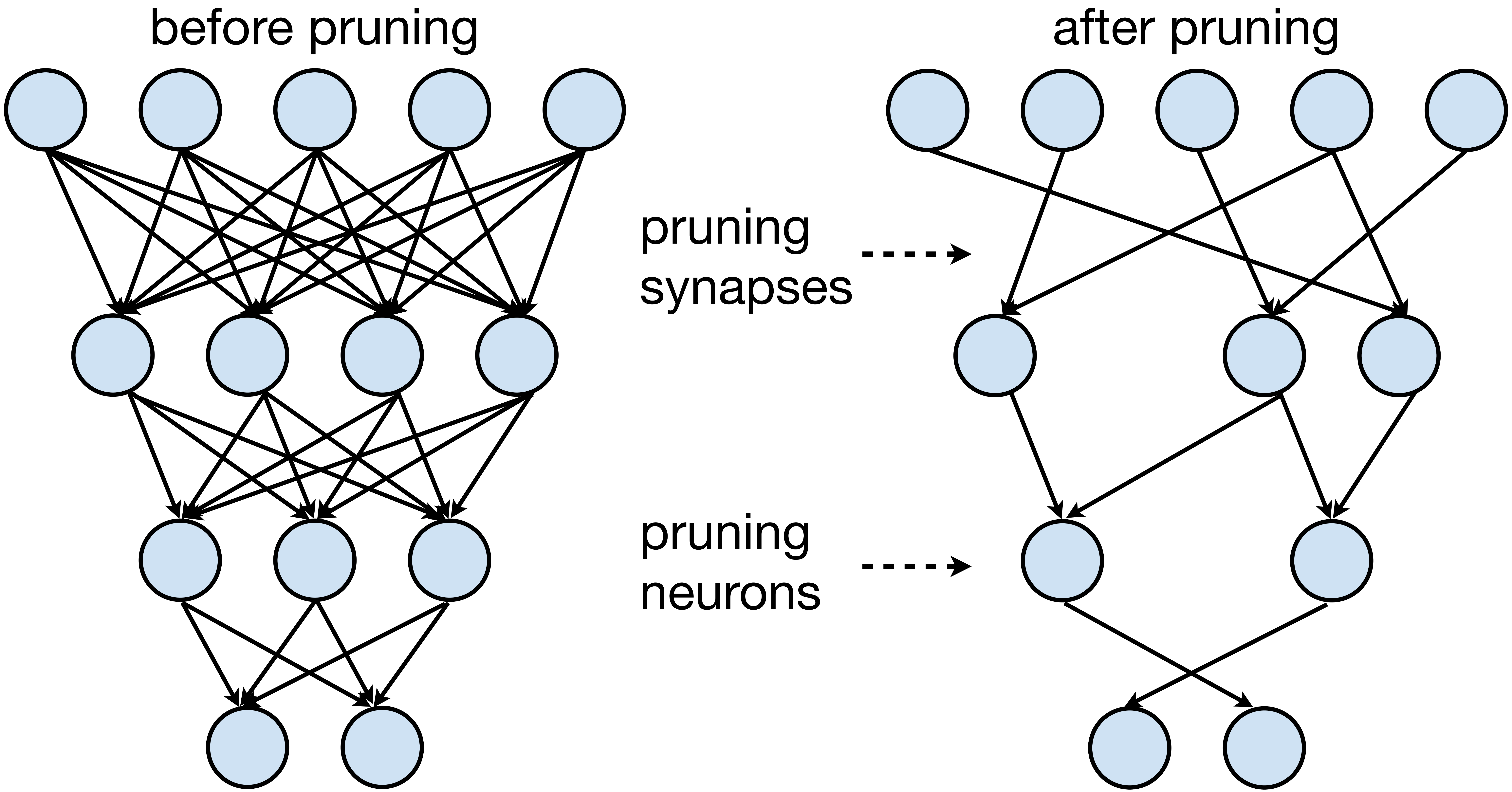
Codesign

- **Codesign**: intrinsic development loop between algorithm design, training, and implementation
- Compression
  - Maintain high performance while removing redundant operations
- Quantization
  - Reduce precision from 32-bit floating point to 16-bit, 8-bit, …
- Parallelization
  - Balance parallelization (how fast) with resources needed (how costly)



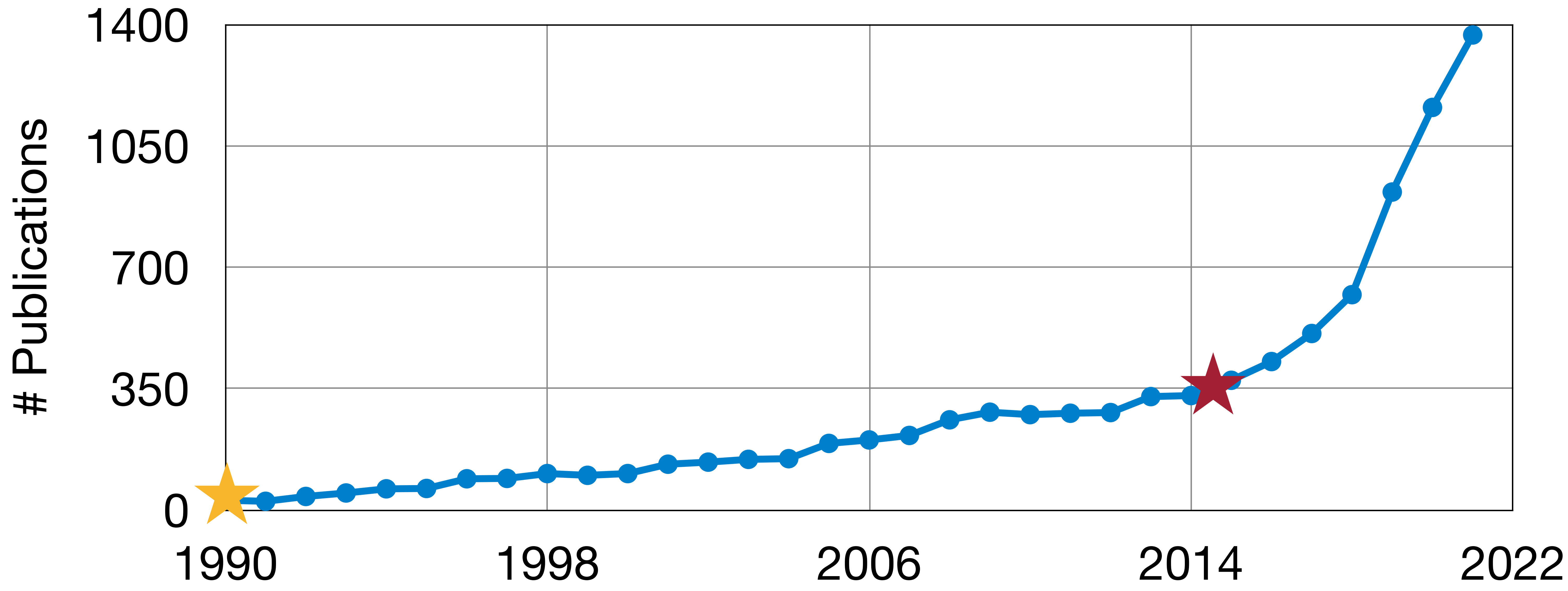
What is pruning?

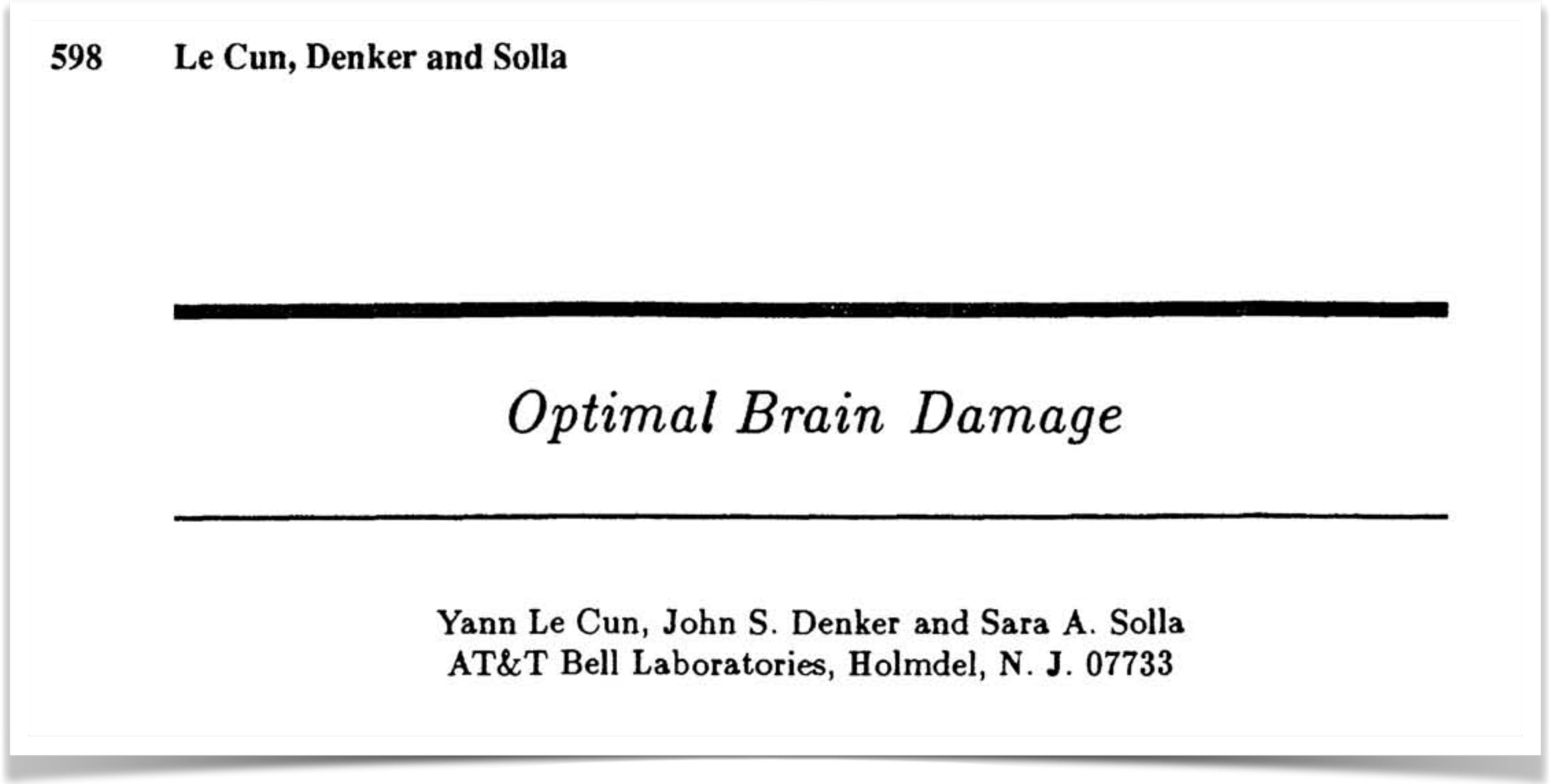
Make neural network smaller by removing synapses and neurons



Optimal Brain Damage [LeCun *et al.,* NeurIPS 1989]
Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

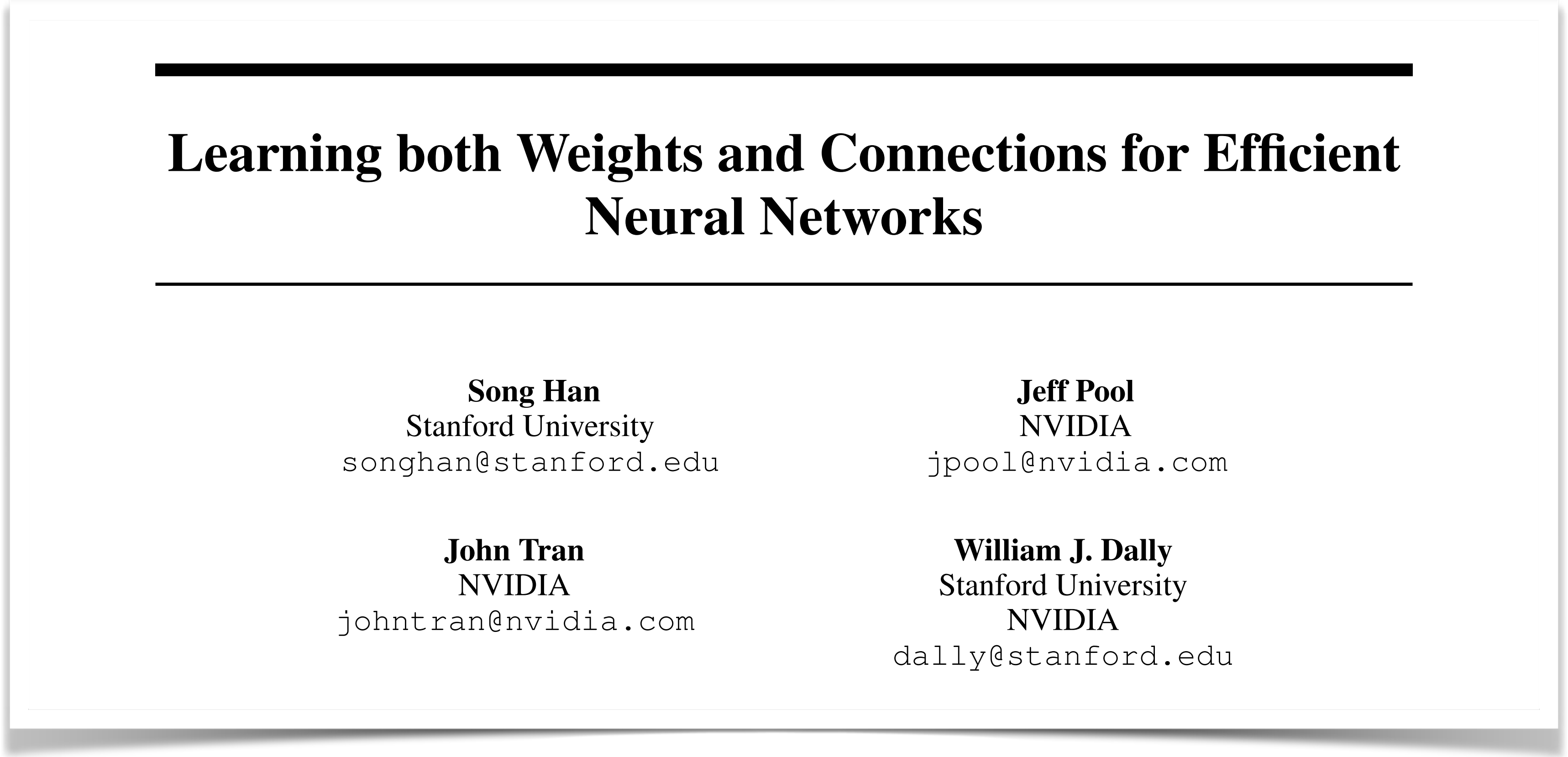
Pruning research




598 Le Cun, Denker and Solla

Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733


Learning both Weights and Connections for Efficient Neural Networks

Song Han
Stanford University
songhan@stanford.edu

Jeff Pool
NVIDIA
jpool@nvidia.com

John Tran
NVIDIA
johntran@nvidia.com

William J. Dally
Stanford University
NVIDIA
dally@stanford.edu

Souce: [Dimension.ai]

Pruning formalism

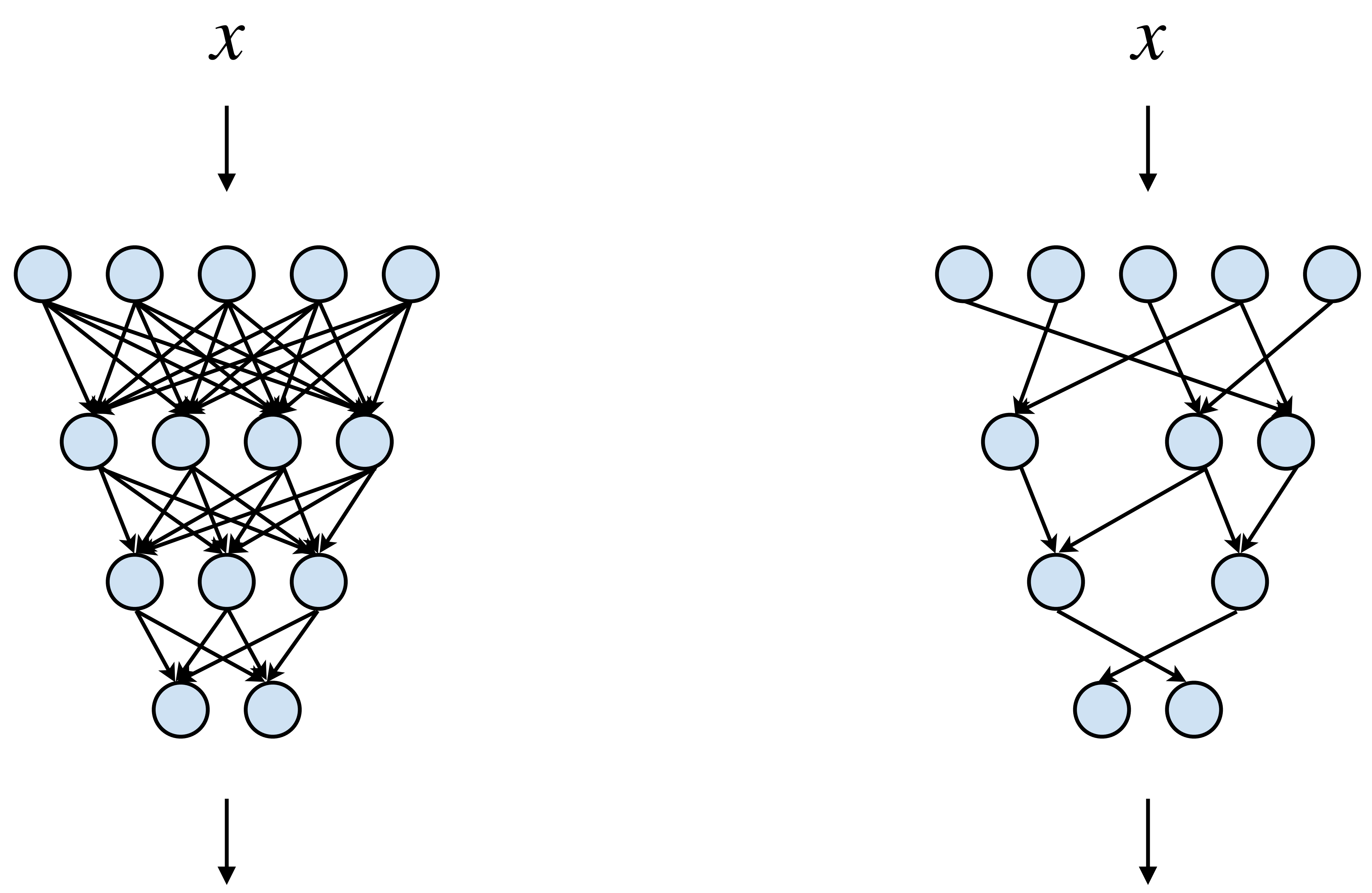
- In general, we could formulate the pruning as follows:

$$\arg\min_{\mathbf{W}_P} L(\mathbf{x}; \mathbf{W}_P)$$

subject to

$$\|\mathbf{W}_p\|_0 < N$$

  - L represents the objective function for neural network training;
  - $\mathbf{x}$ is input, $\mathbf{W}$ is original weights, $\mathbf{W}_P$ is pruned weights;
  - $\|\mathbf{W}_p\|_0$ calculates the #nonzeros in W_P, and N is the target #nonzeros.

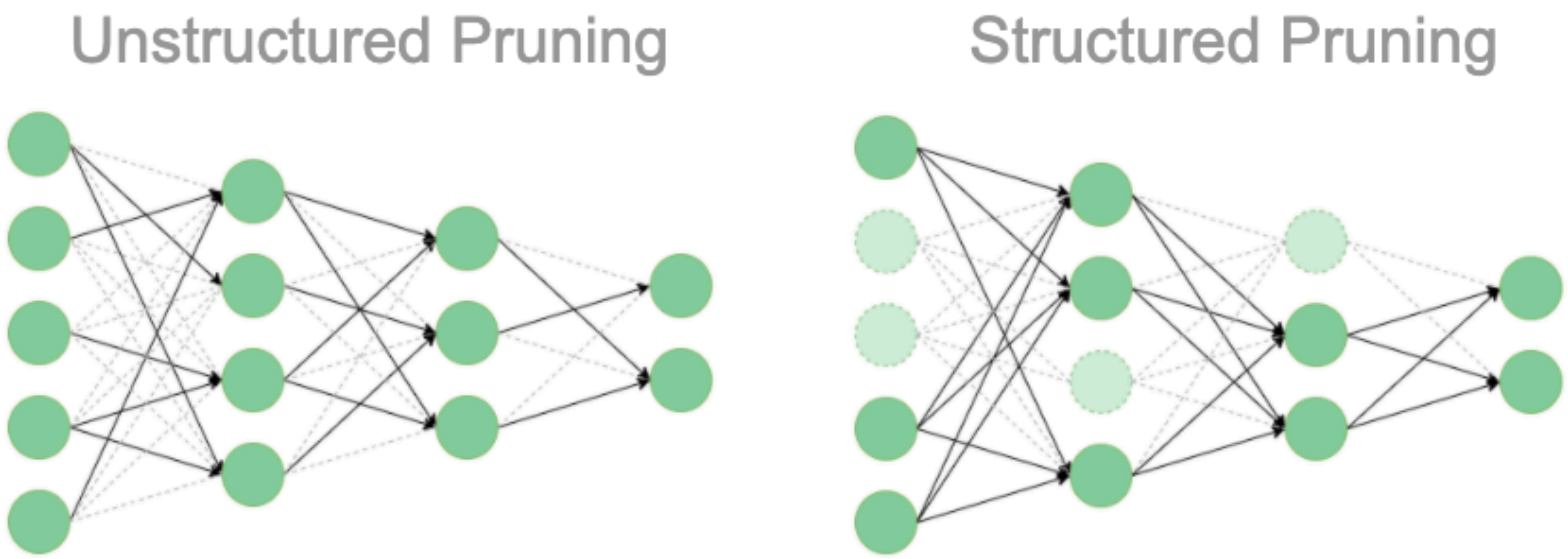


$$\arg\min_{\mathbf{W}} L(\mathbf{x}; \mathbf{W})$$

$$\arg\min_{\mathbf{W}_P} L(\mathbf{x}; \mathbf{W}_P)$$
$$s.t. \|\mathbf{W}_P\|_0 \le N$$

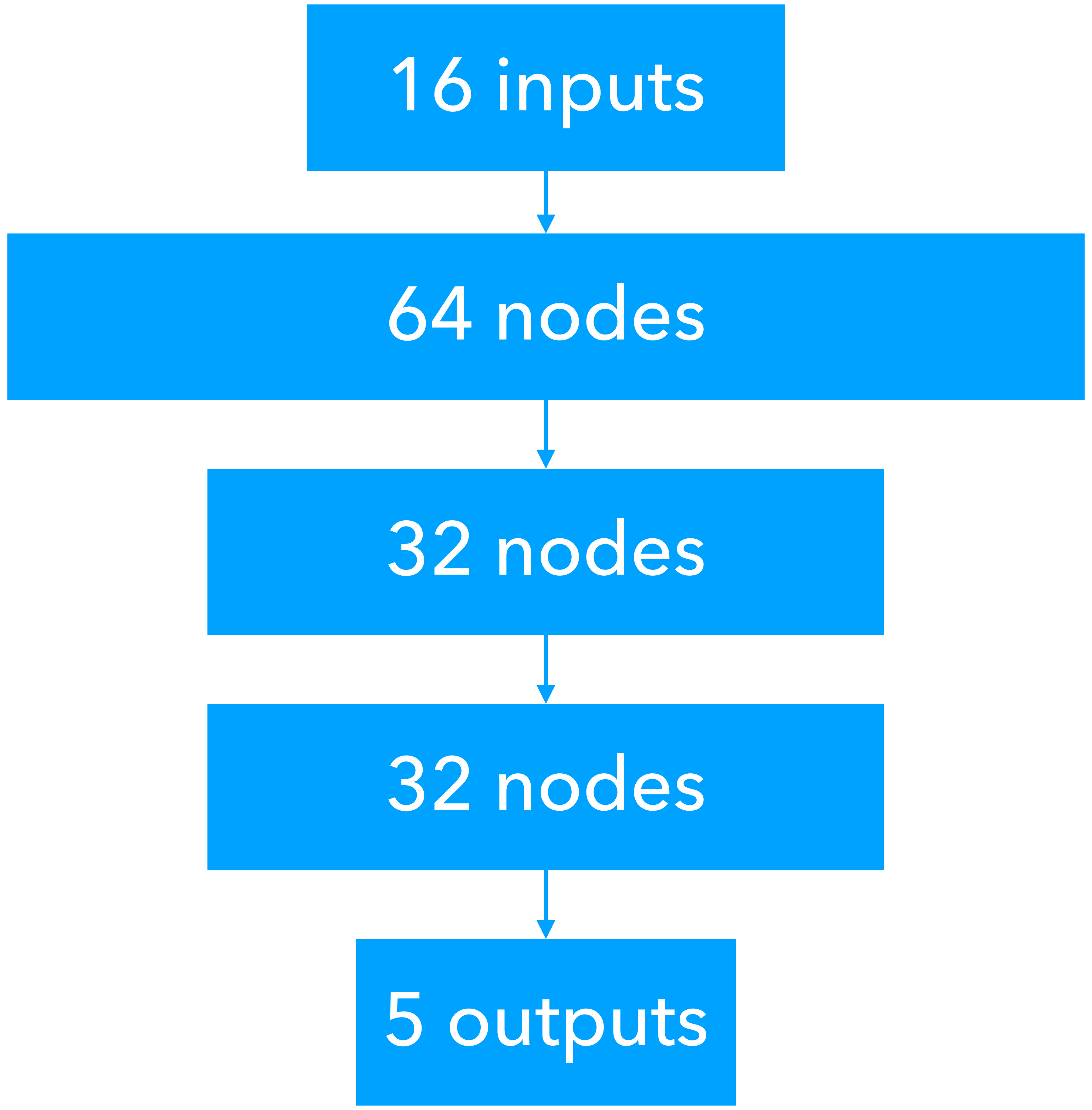
Structured vs. unstructured pruning

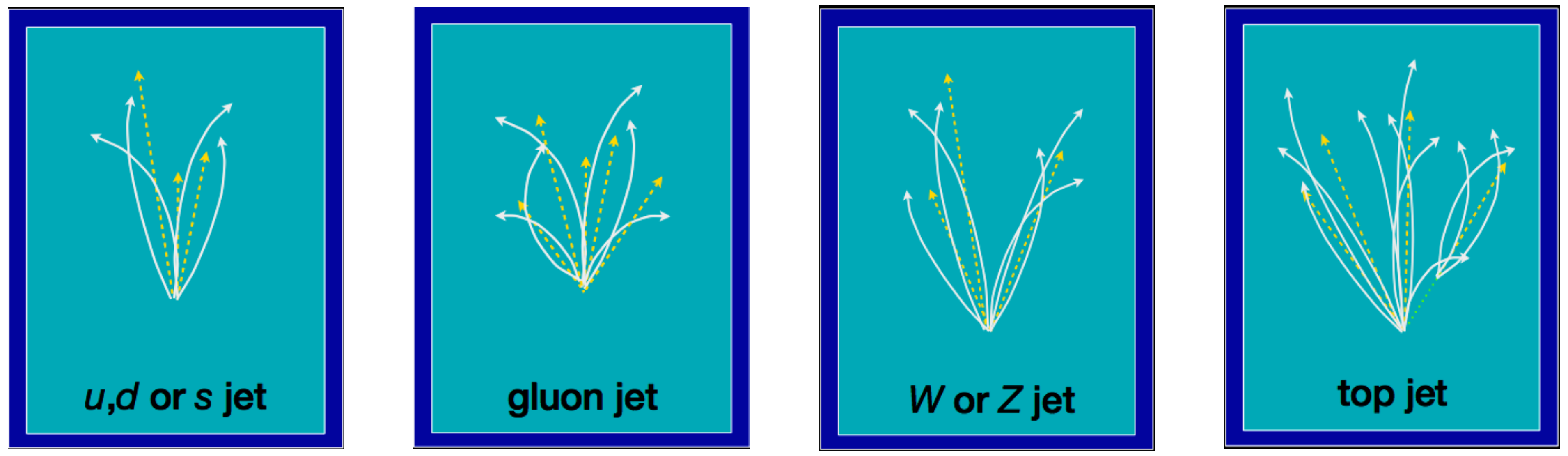
- Unstructured pruning: removing some connections regardless of placement
  - Can potentially achieve higher performance for smaller size
- Structured pruning: removing all input/output connections of particular nodes
  - More regular structure easier to support in hardware architectures

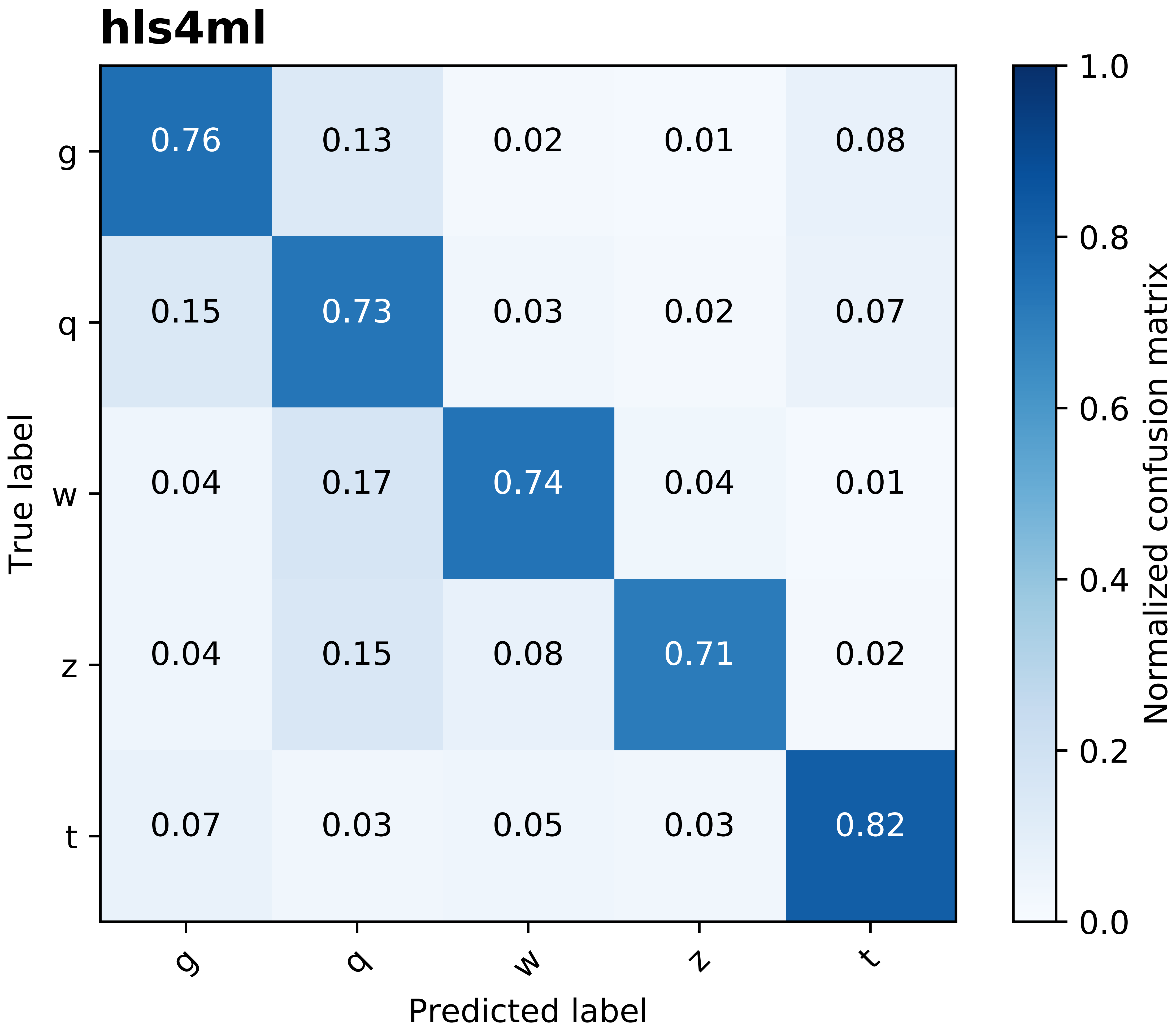


Benchmark: Jet tagging MLP

Small NN benchmark correctly identifies particle “jets” 70-80% of the time







Iterative magnitude-based pruning arXiv:1804.06913

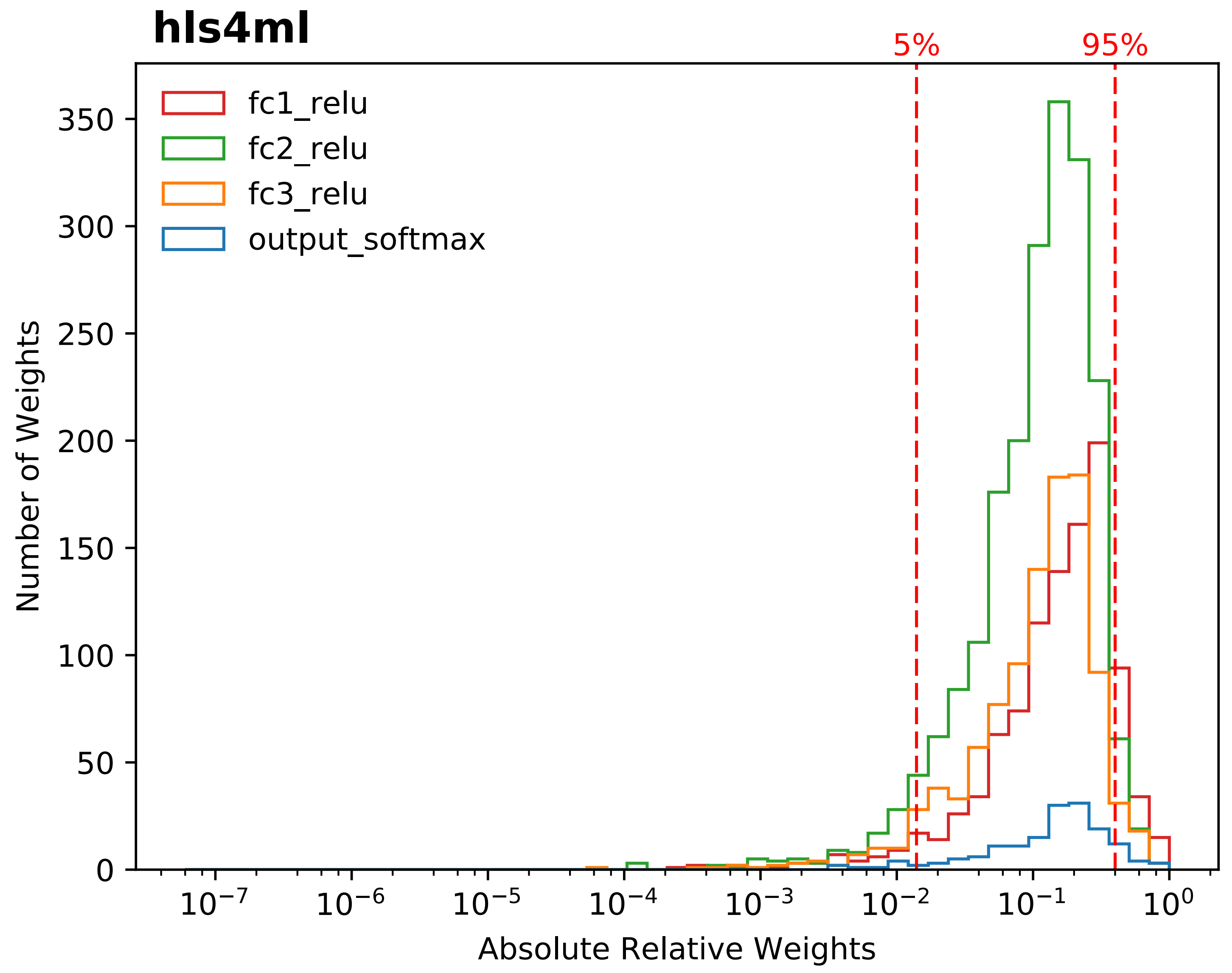
- Train with L_1 regularization (down-weights unimportant synapses)

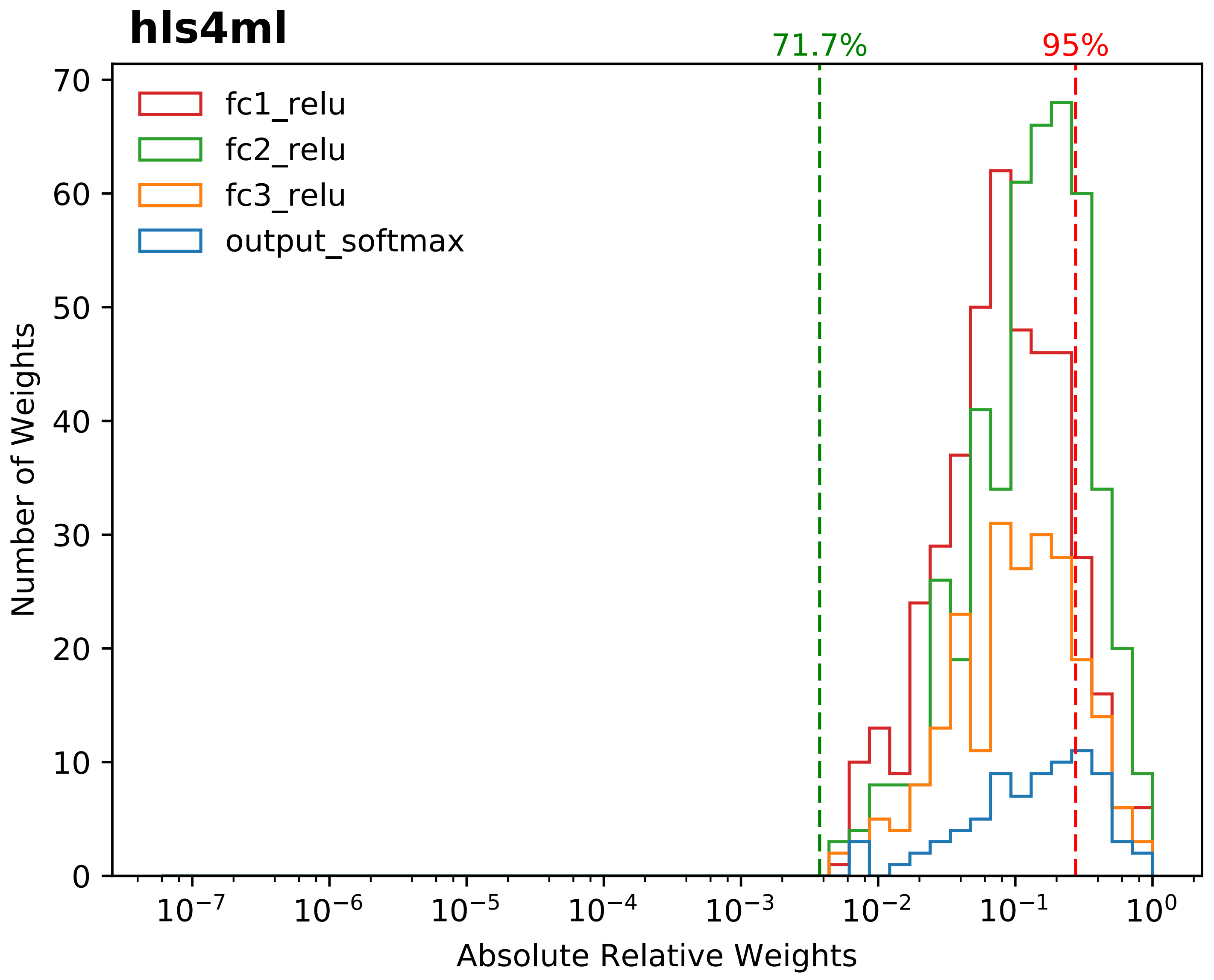
$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove smallest weights

70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.

- Iterate





Iterative magnitude-based pruning

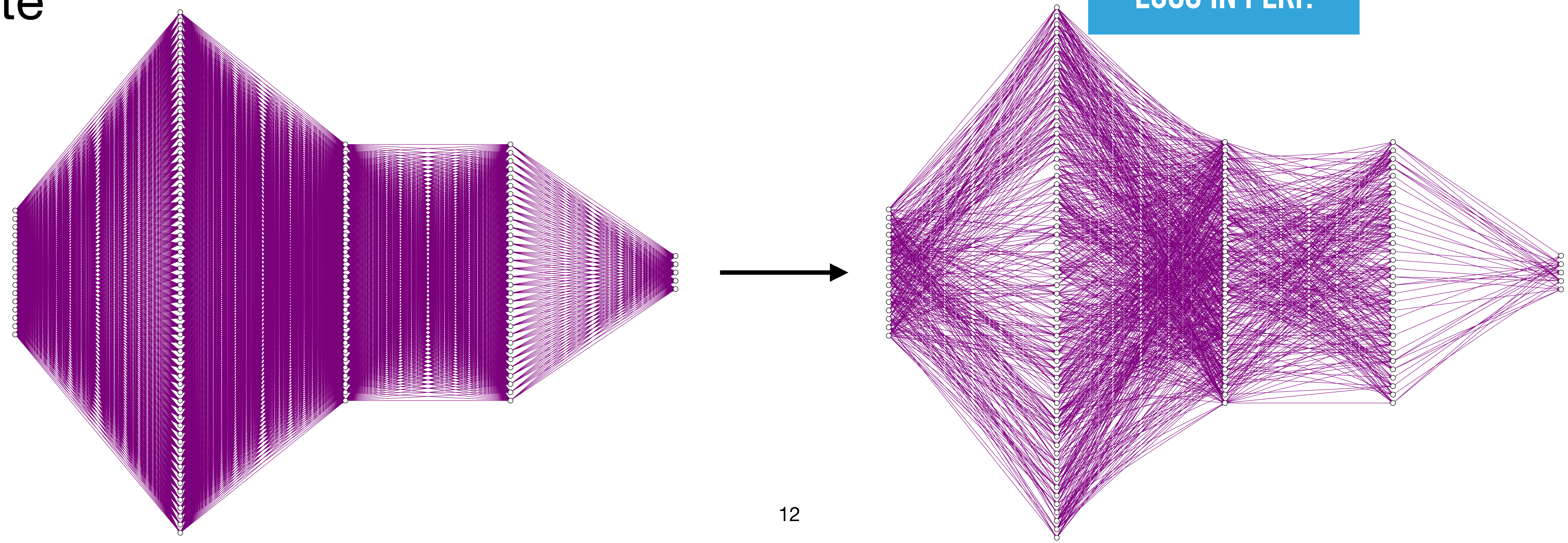
[arXiv:1804.06913](https://arxiv.org/abs/1804.06913)

- Train with L_1 regularization (down-weights unimportant synapses)

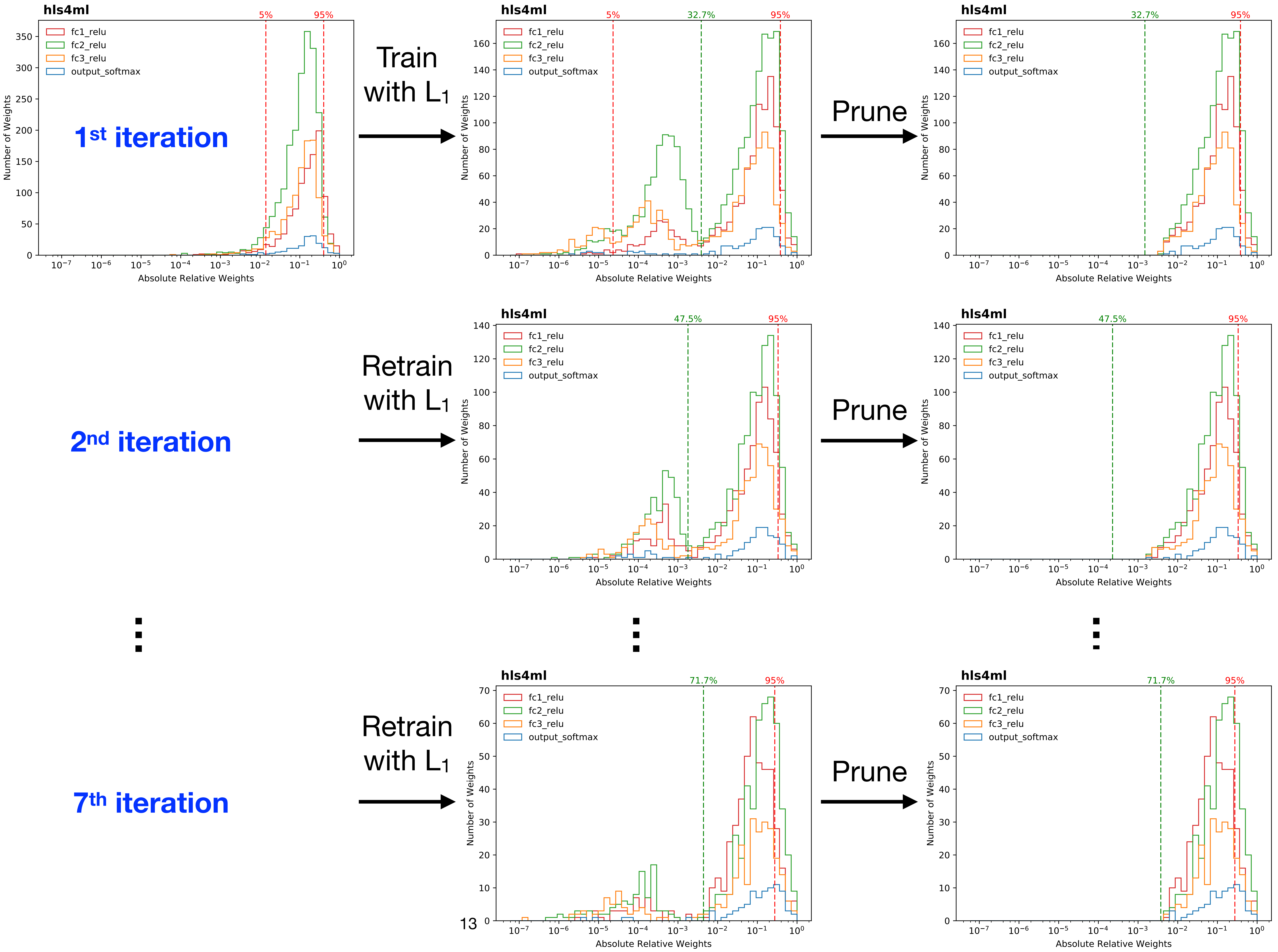
$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove smallest weights
- Iterate

70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.



Pruning

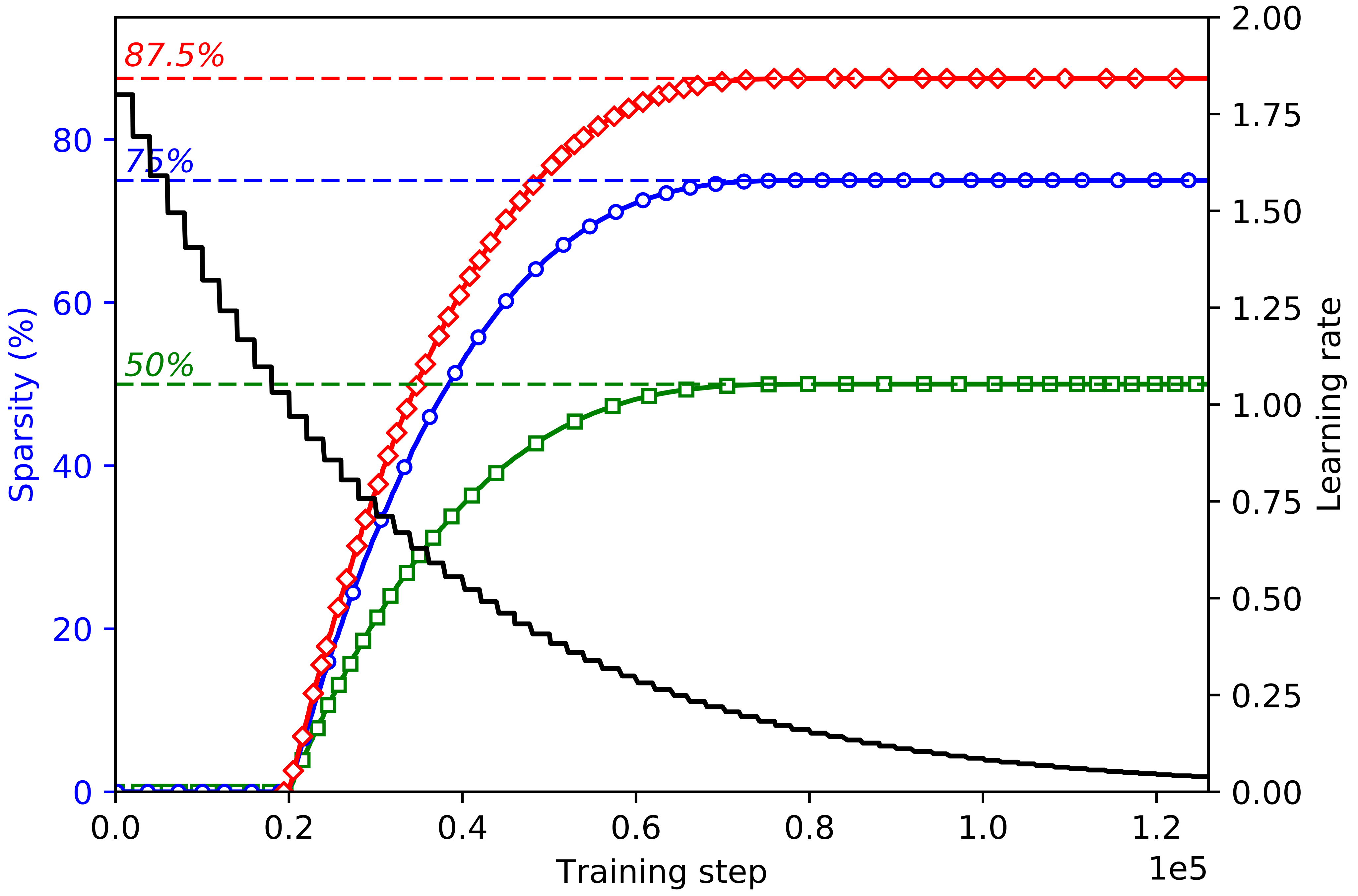


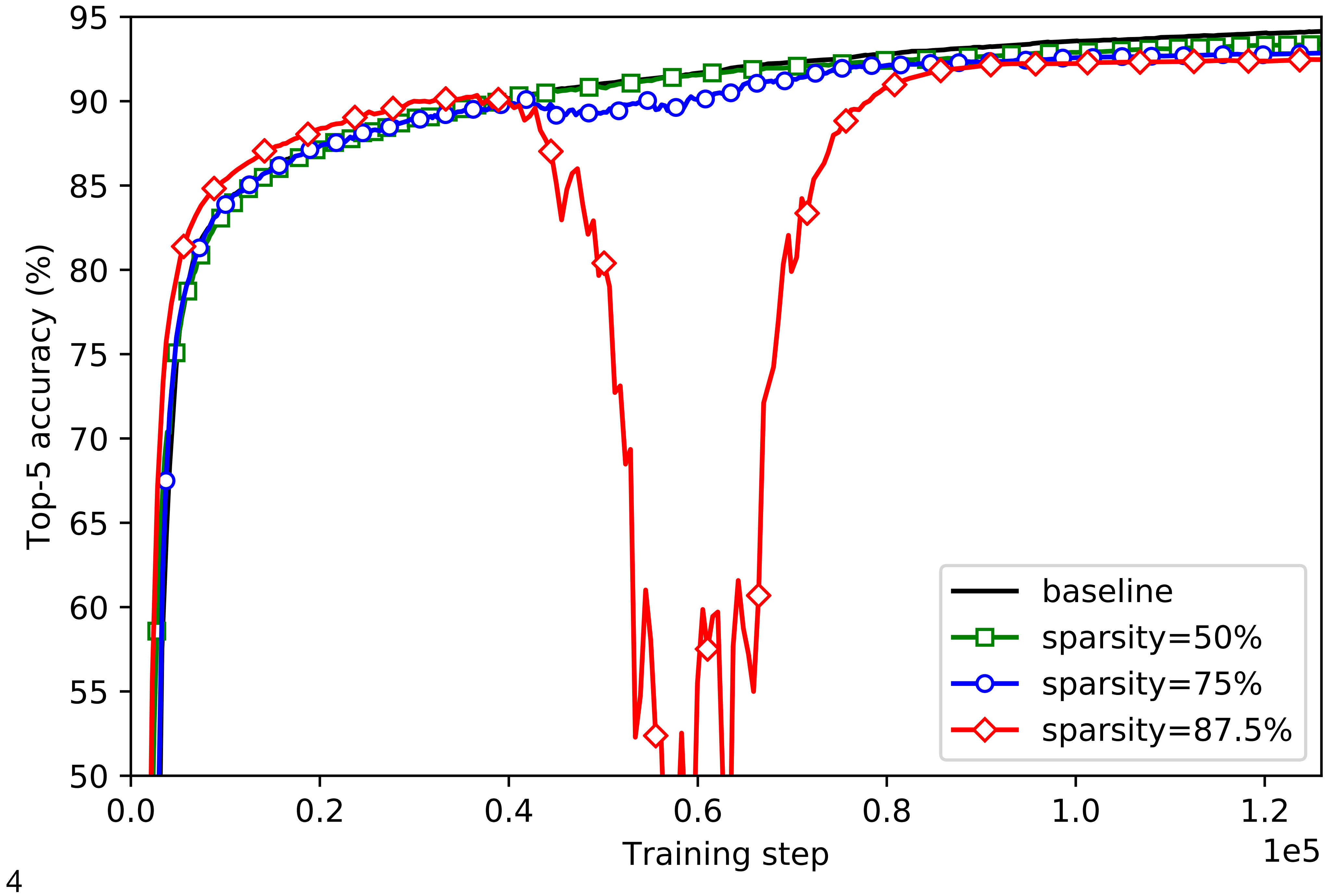
1st iteration
2nd iteration
7th iteration
Train with L1
Retrain with L1
Retrain with L1
Prune
Prune
Prune
hls4ml
fc1_relu
fc2_relu
fc3_relu
output_softmax
5%
95%
32.7%
47.5%
71.7%
Number of Weights
Absolute Relative Weights

Pruning APIs

arXiv:1710.01878

- TensorFlow API: https://www.tensorflow.org/model_optimization/guide/pruning
    - Sparsity schedule for gradual pruning
- Similar API for PyTorch: https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

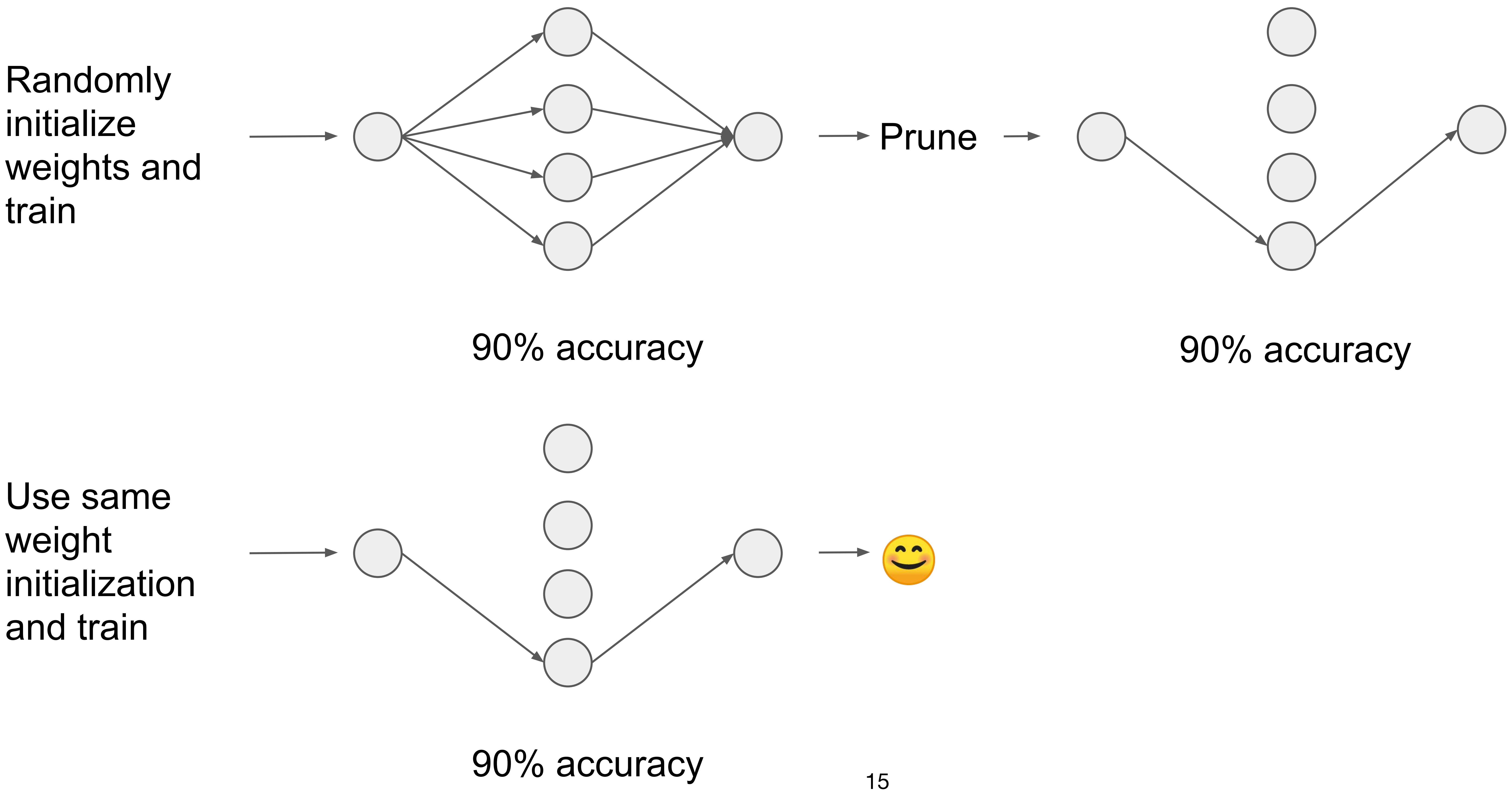




Aside: Lottery Ticket Hypothesis

arXiv:1803.03635

- A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations

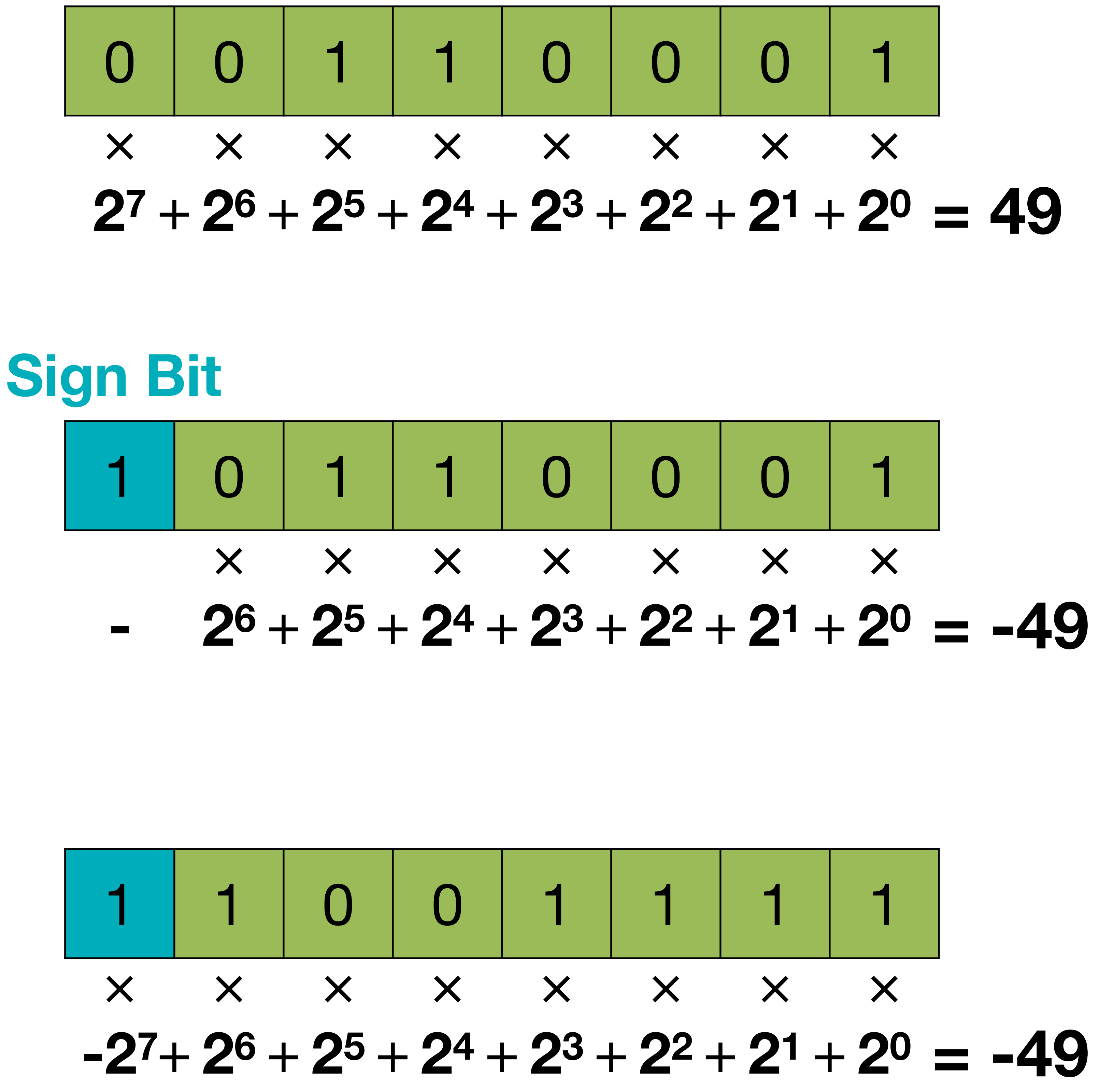


Numeric data types: integer

- Unsigned Integer
  - *n*-bit Range: $[0,\ 2^n - 1]$

- Signed Integer
  - Sign-Magnitude Representation
    - *n*-bit Range: $[-2^{n-1} - 1,\ 2^{n-1} - 1]$
    - Both 000…00 and 100…00 represent 0

  - Two's Complement Representation
    - *n*-bit Range: $[-2^{n-1},\ 2^{n-1} - 1]$
    - 000…00 represents 0
    - 100…00 represents -2^{n-1}



Numeric data types: fixed-point number

Integer . Fraction

"Decimal" Point

0	0	1	1	0	0	0	1
×	×	×	×	×	×	×	×

$$-2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 3.0625$$

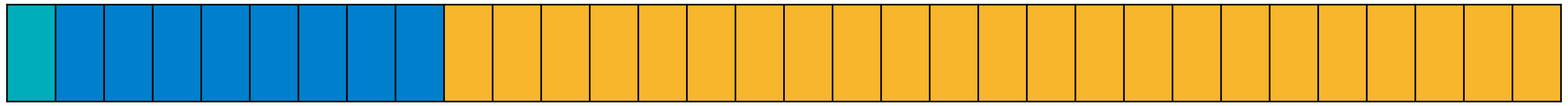
0	0	1	1	0	0	0	1
×	×	×	×	×	×	×	×

$$(-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0) \times 2^{-4} = 49 \times 0.0625 = 3.0625$$

(using 2's complement representation)

Numeric data types: floating-point number

Example: 32-bit floating-point number in IEEE 754



Sign **8 bit Exponent** **23 bit Fraction**

$$(-1)^{\text{sign}} \times (1 + \textbf{Fraction}) \times 2^{\textbf{Exponent}-127}$$

Exponent Bias = 127 = $2^{8-1}-1$

(significant / mantissa)

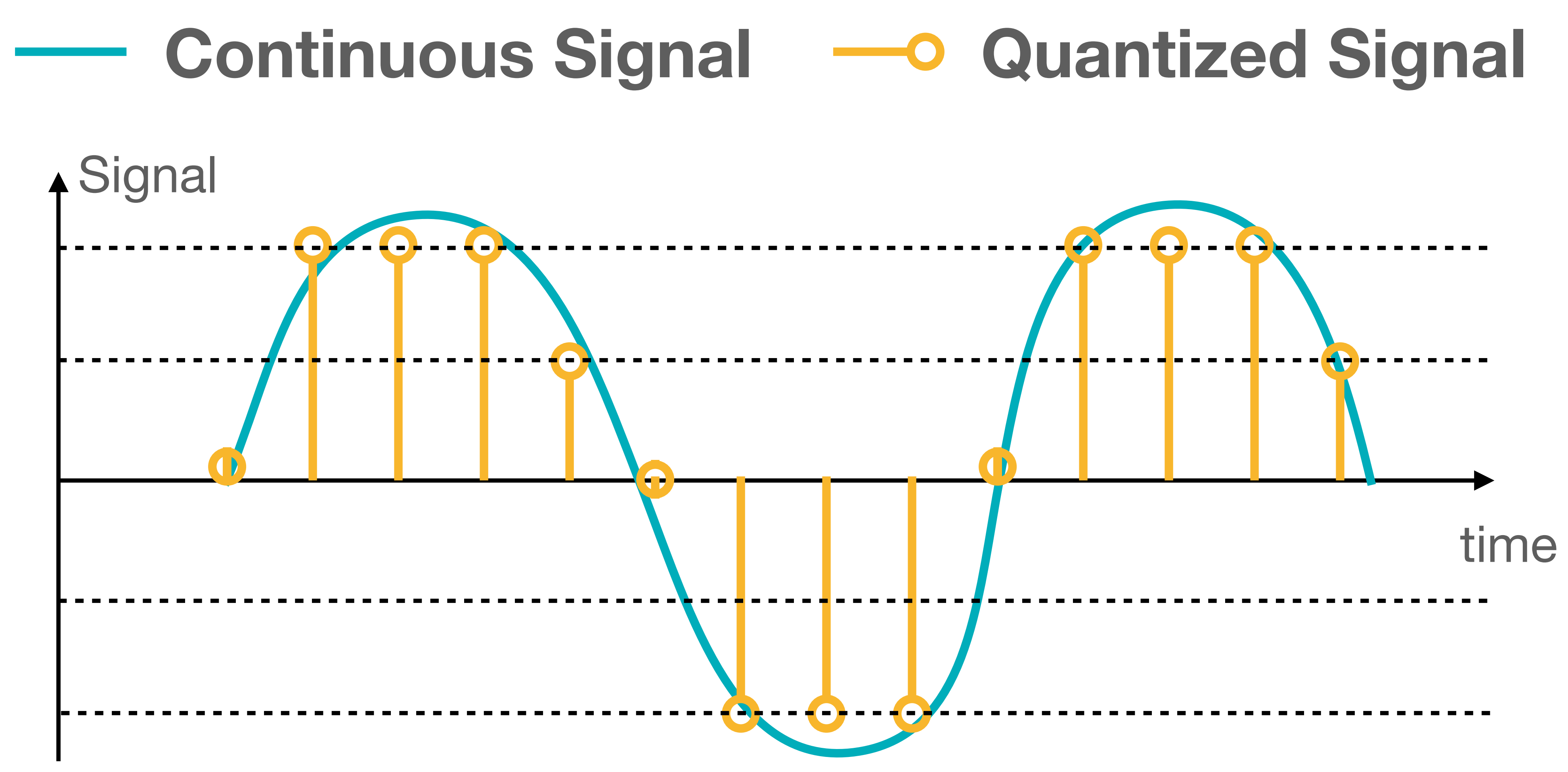
$$0.265625 = 1.0625 \times 2^{-2} = (1 + \underline{0.0625}) \times 2^{\underline{125}-127}$$

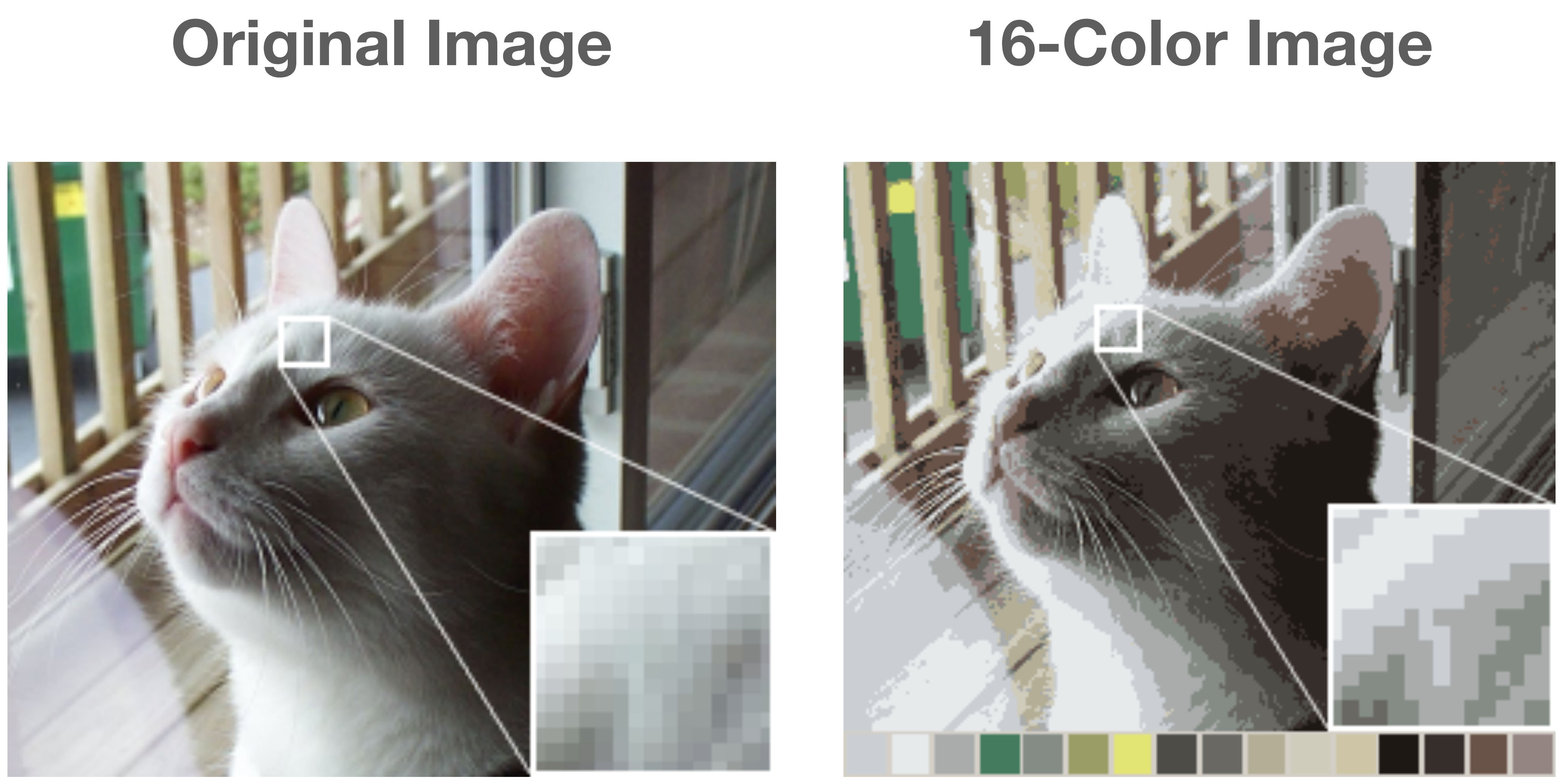
0	0	1	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

125 0.0625

What is quantization?

Quantization is the process of constraining an input from a continuous or otherwise large set of values to a discrete set.





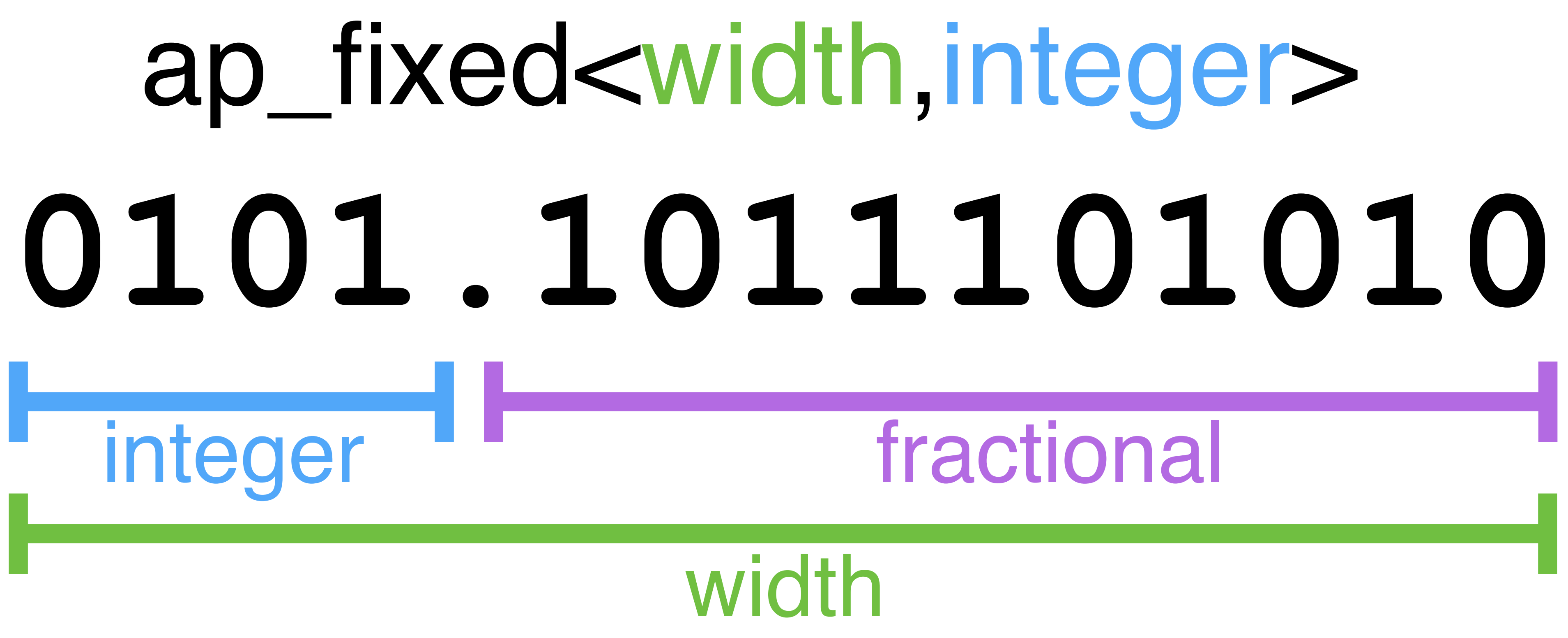
Images are in the public domain.

Wikipedia: Quantization

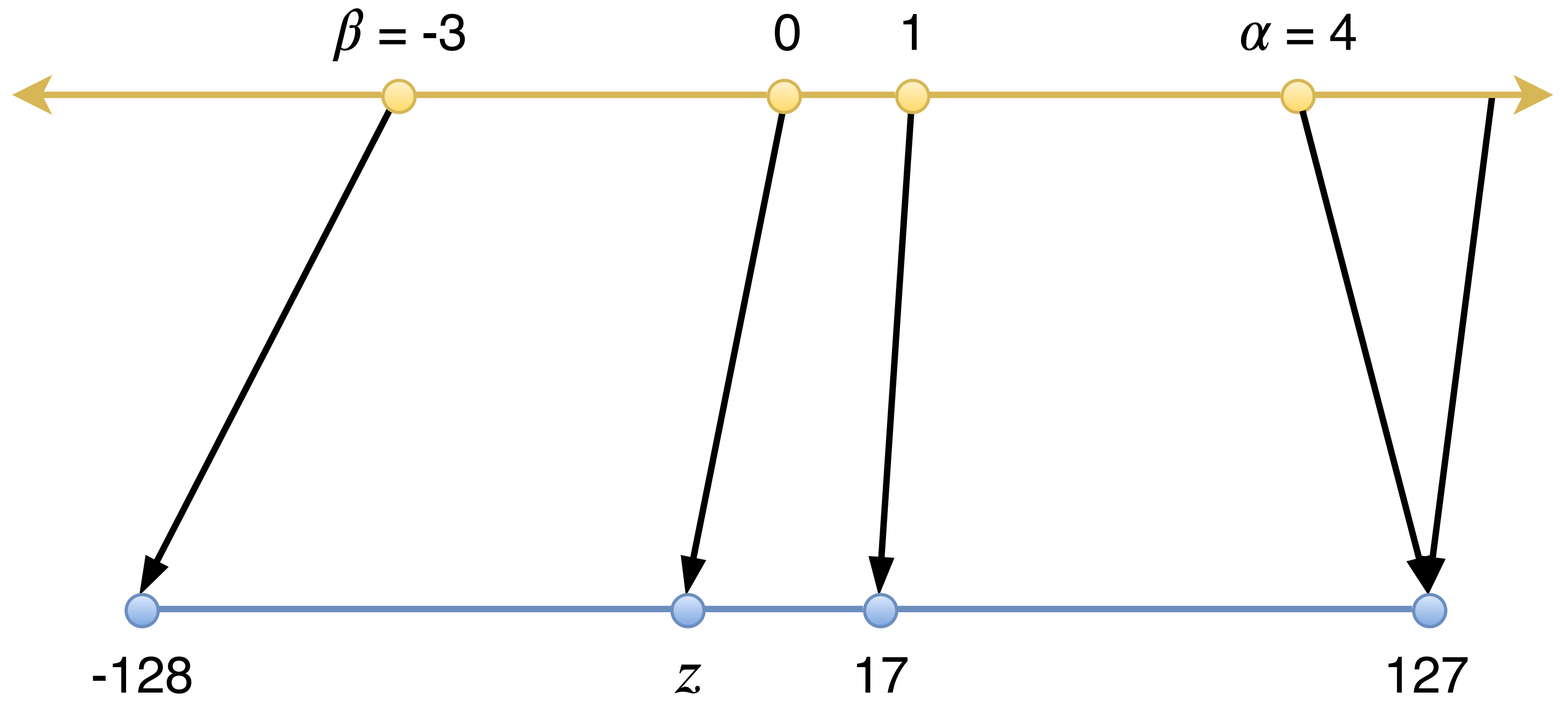
Quantization types

arXiv:2004.09602

- Quantization: using reduced precision for parameters and operations
- Fixed-point precision



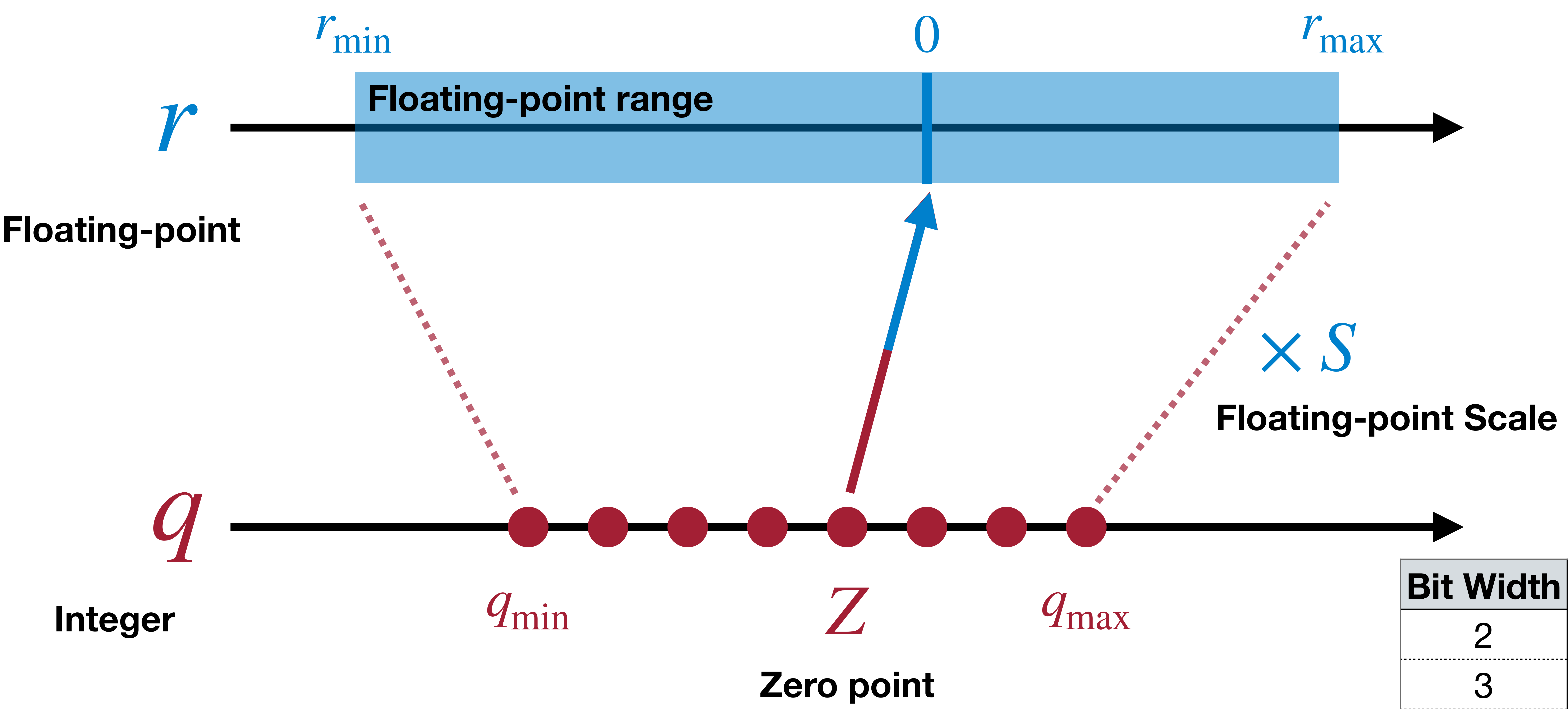
- Affine integer quantization



Affine integer quantization

arXiv:2004.09602

An affine mapping of integers to real numbers $r = S(q - Z)$



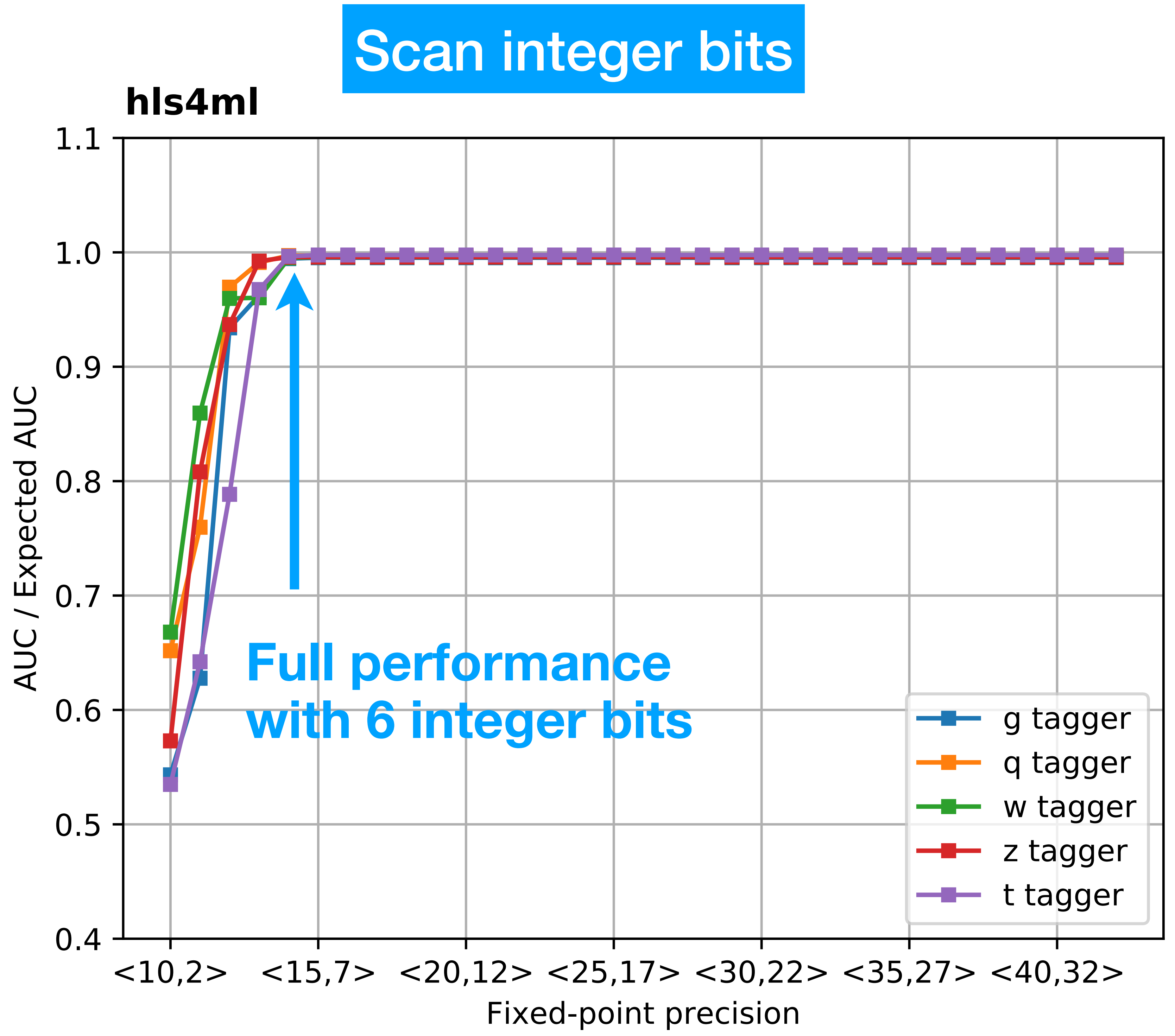
Bit Width	q_{min}	q_{max}
2	-2	1
3	-4	3
4	-8	7
N	-2^{N-1}	$2^{N-1}-1$

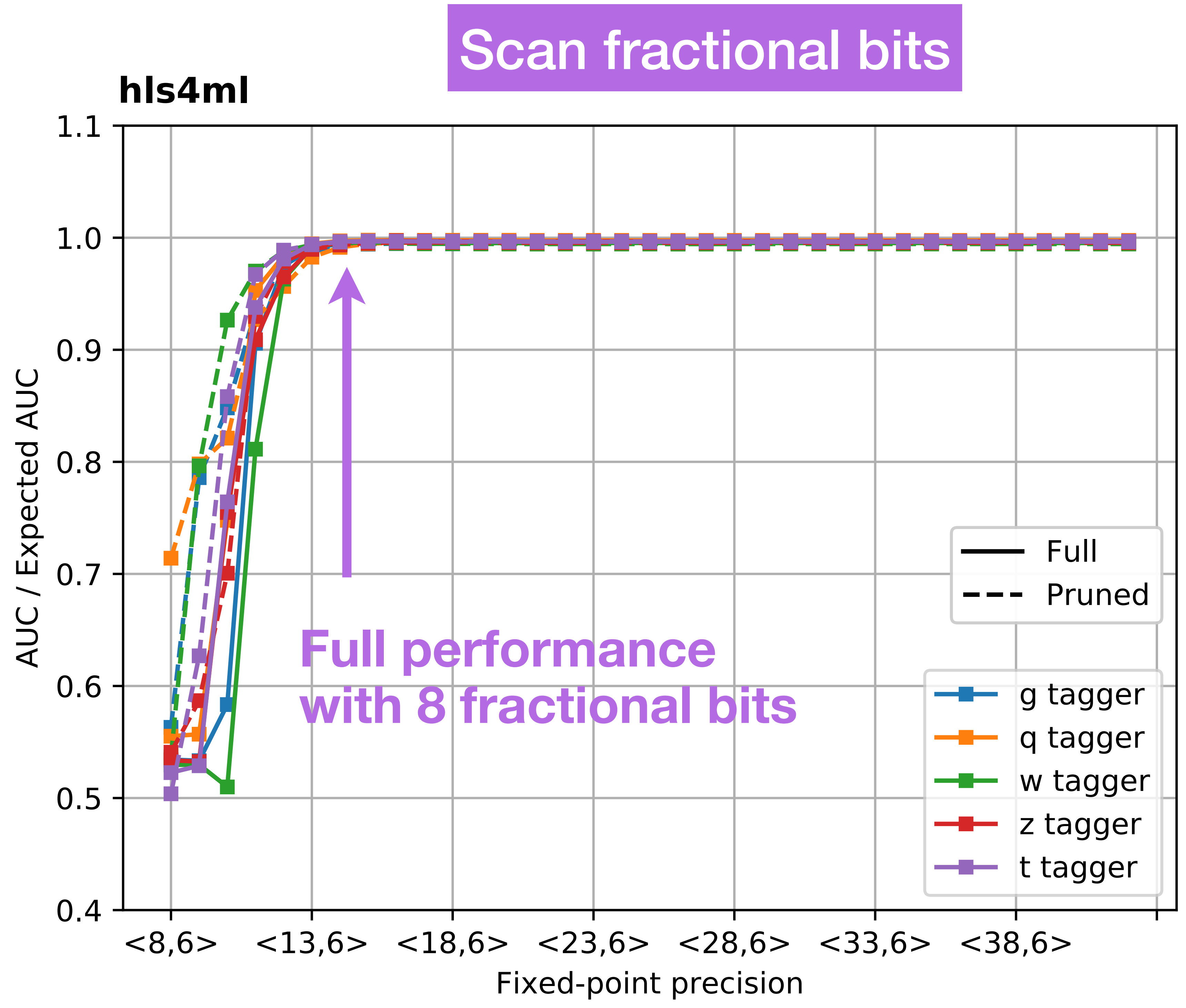
Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [Jacob *et al.*, CVPR 2018]

Post-training quantization vs. quantization-aware training

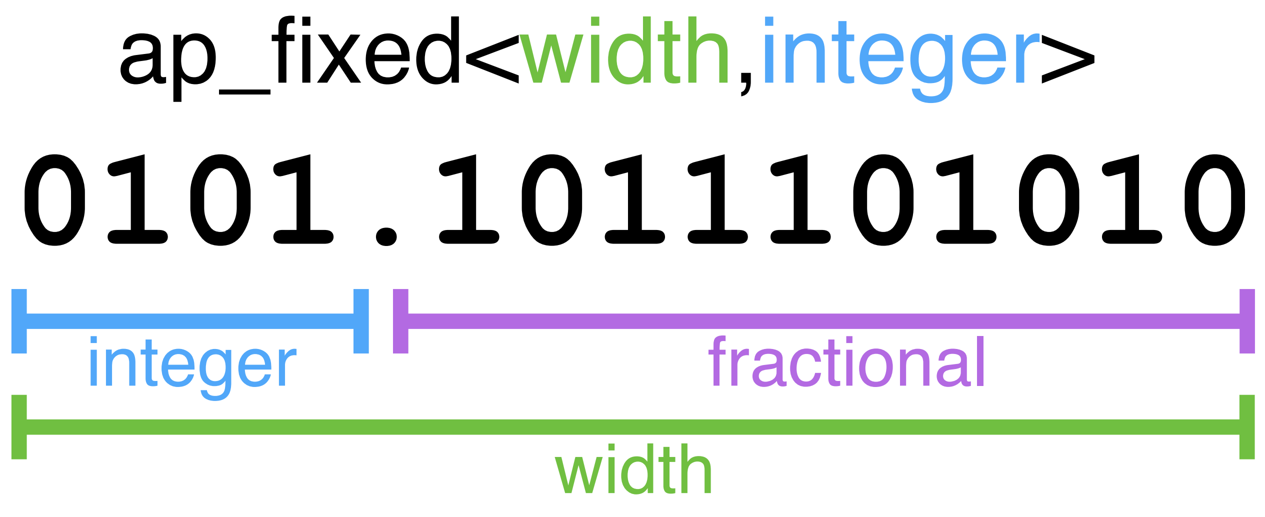
PTQ	QAT
Usually fast	Slow
No re-training of the model	Model needs to be trained/finetuned
Plug and play of quantization schemes	Plug and play of quantization schemes (requires re-training)
Less control over final accuracy of the model	More control over final accuracy since *q-params* are learned during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

Post-training quantization





- General strategy: avoid overflows in integer bit then scan the decimal bit until reaching optimal performance

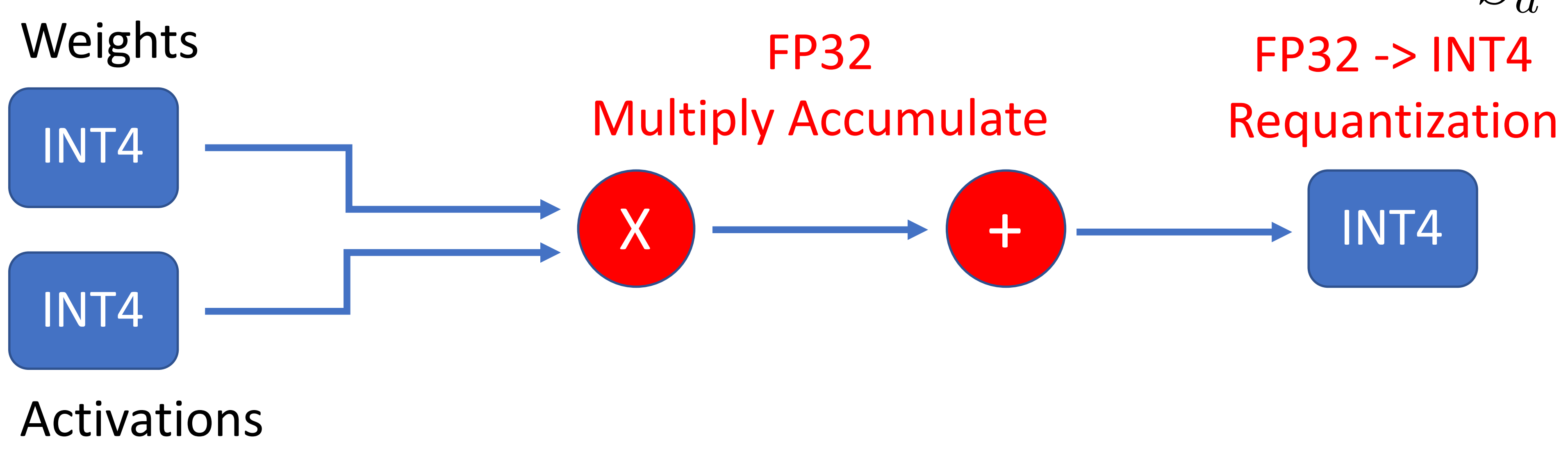


Quantization-aware training: how does it work?

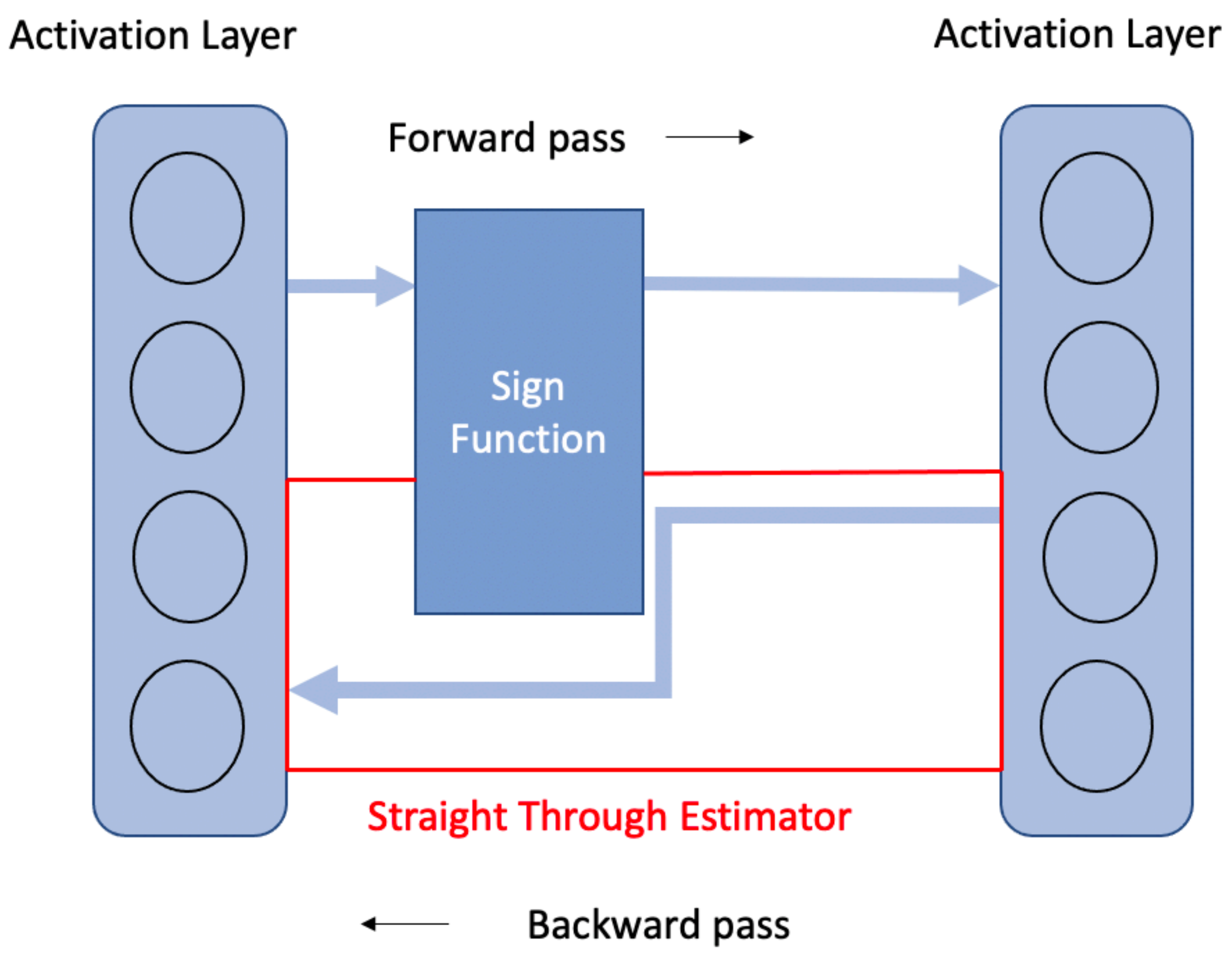
- Fake quantization: using 32-bit floating-point math under the hood
- Straight-through estimator: during backpropagation, ignore quantization operation (treat as identity)

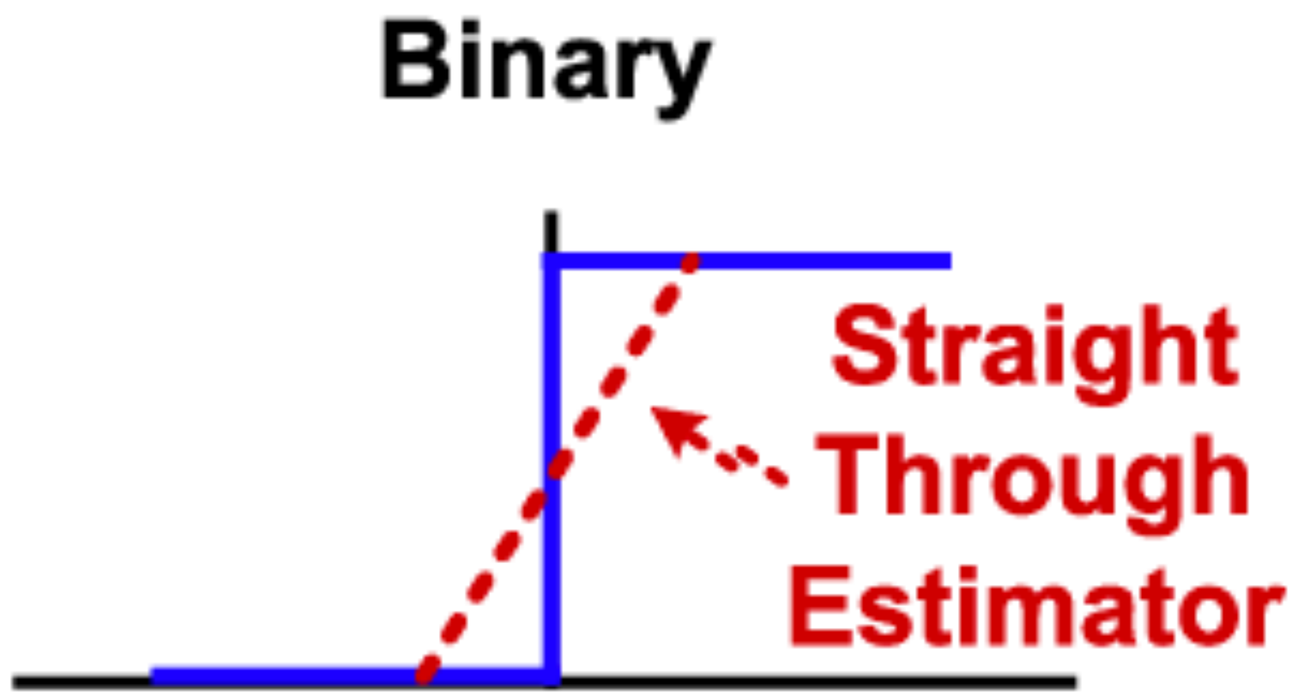
$$W^{\text{fp32}} = (S_w, W^{i4})^{\text{fp32}} \qquad a^{\text{fp32}} = W^{\text{fp32}} h^{\text{fp32}} \qquad a^{i4} = Int(\frac{a^{\text{fp32}}}{S_a})$$

Fake Quantization:



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

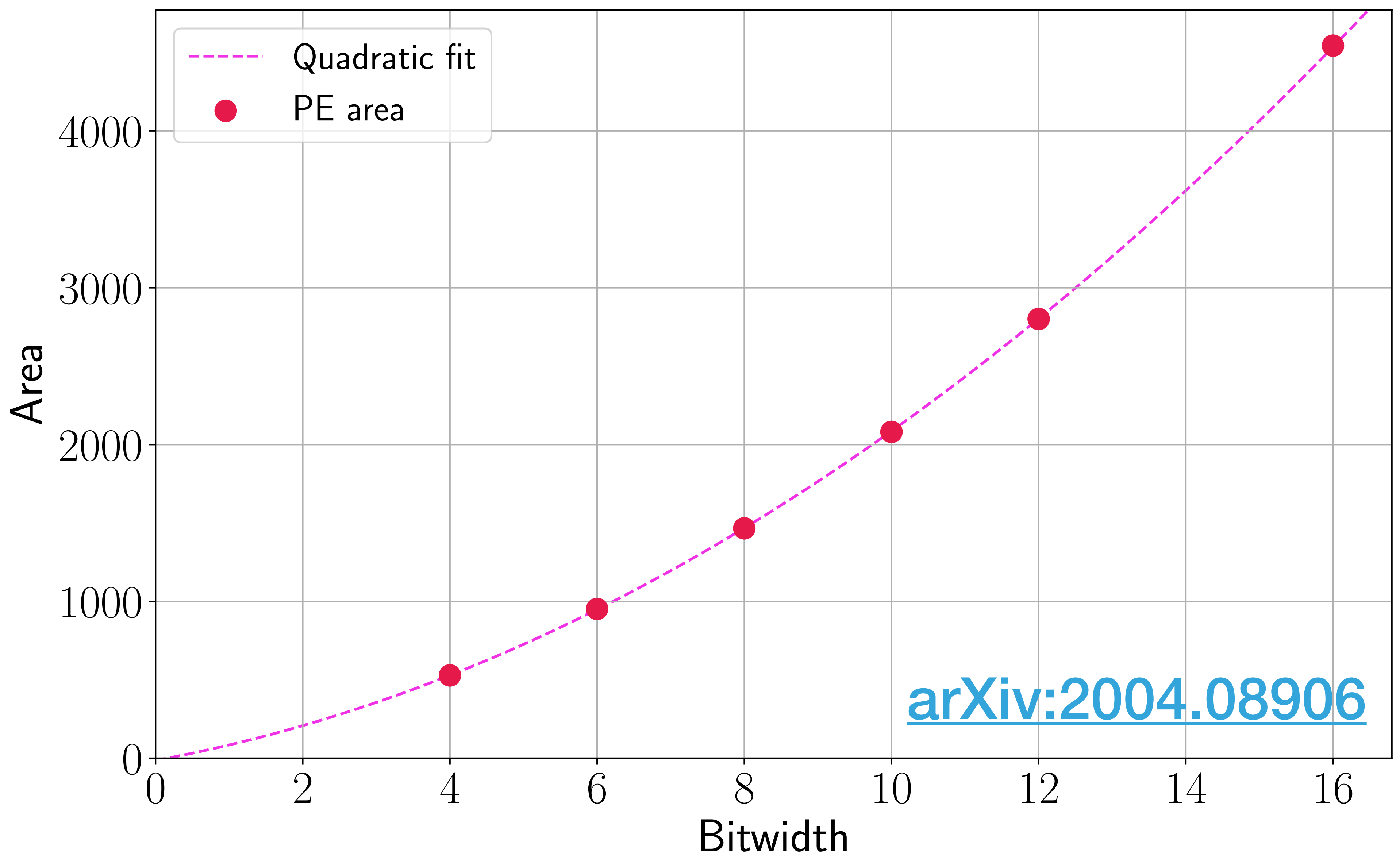


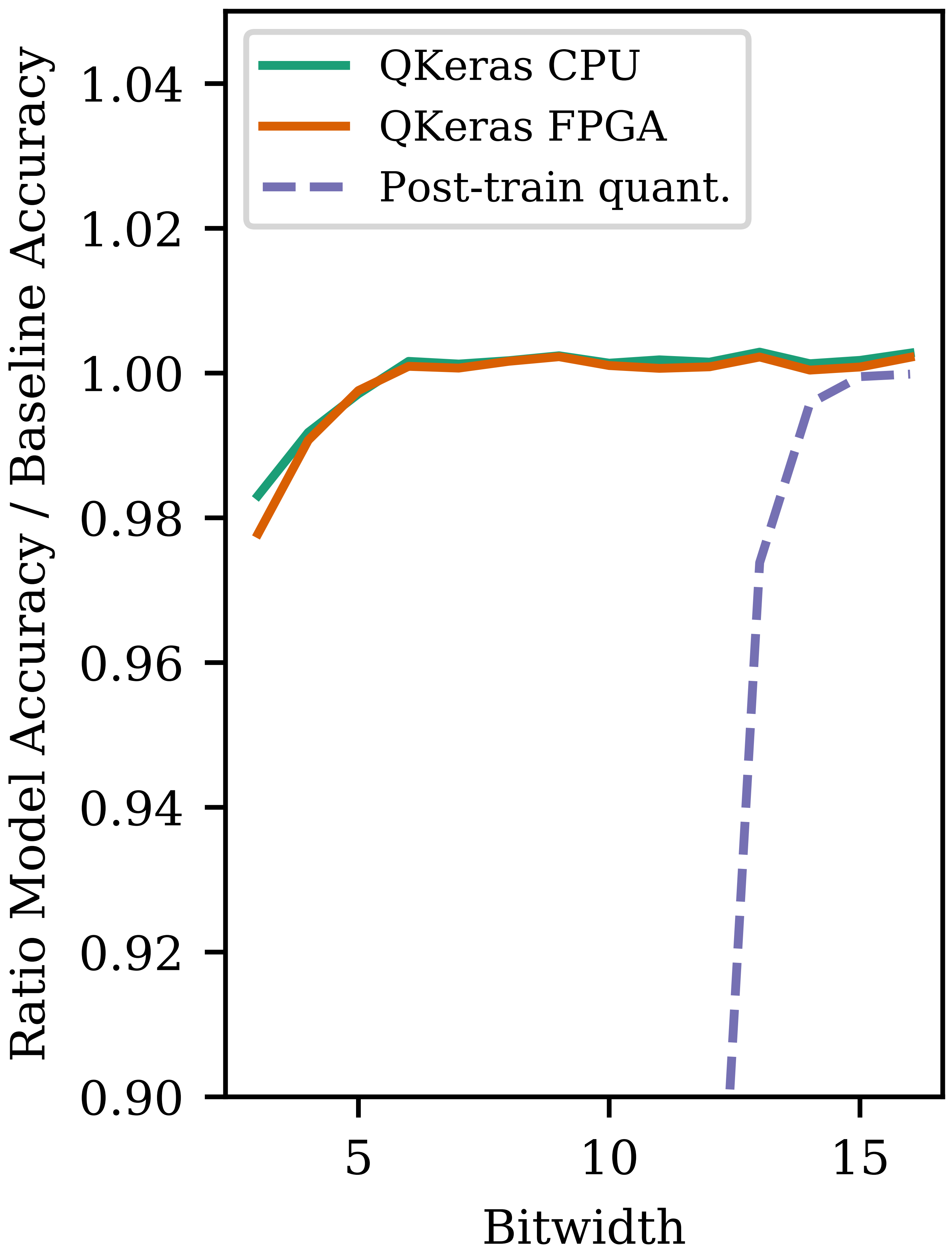


Quantization-aware training

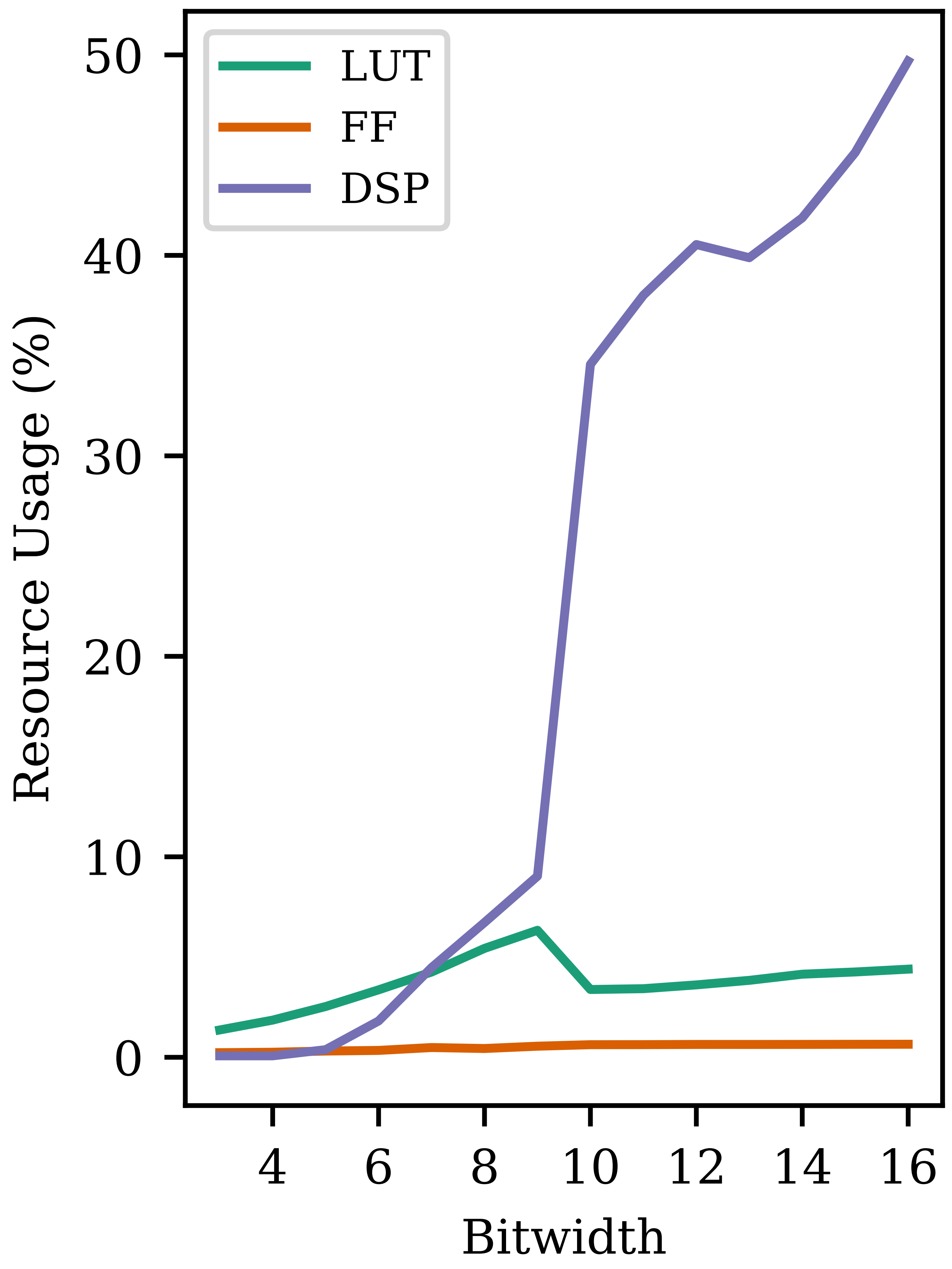
arXiv:2006.10159

- Full performance with 6 bits instead of 14 bits
- Much smaller fraction of resources
- Area & power scale quadratically with bit width





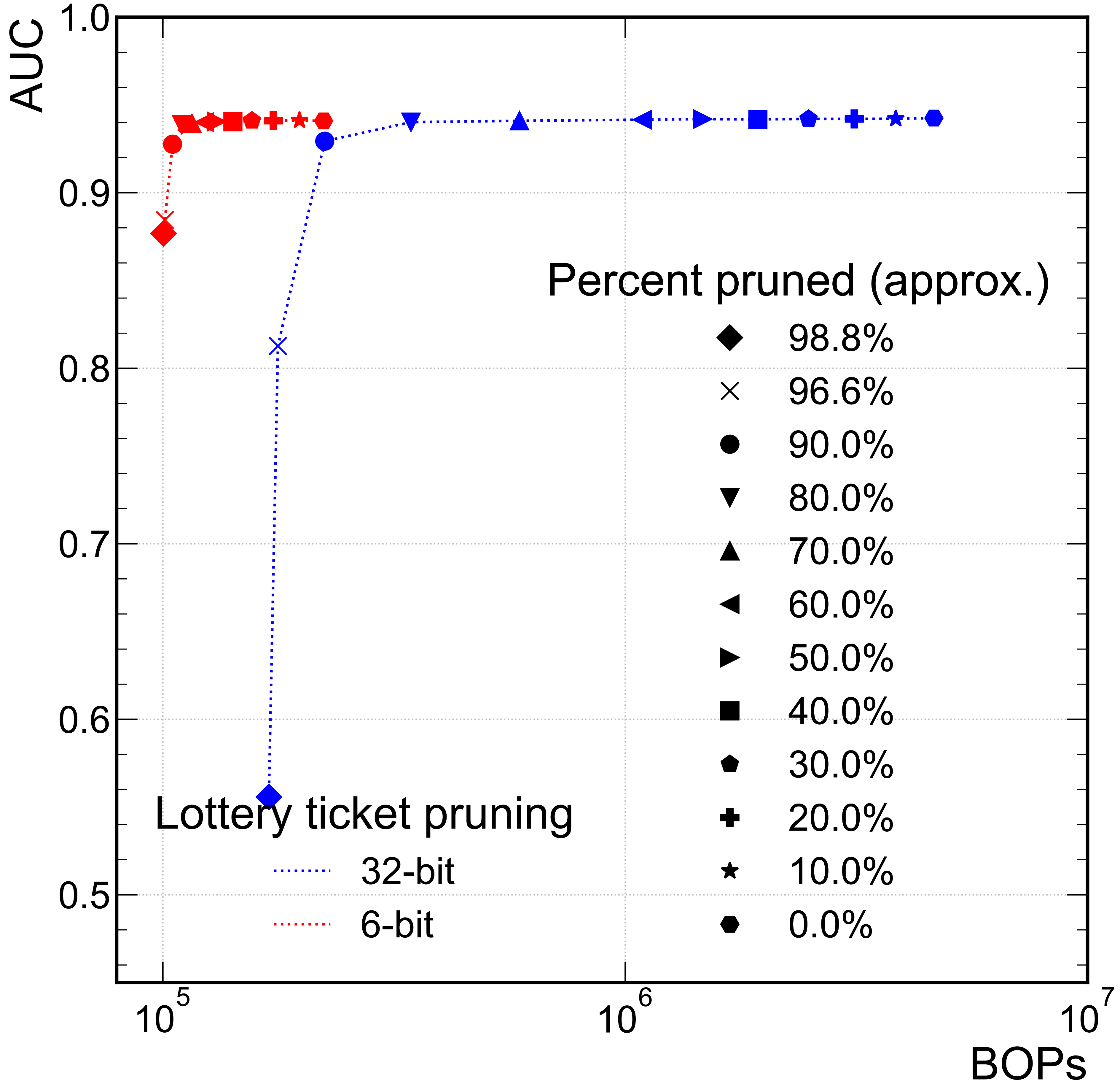
Xilinx VU9P



Pruning + quantization-aware training

arXiv:2102.11289

- Quantization-aware pruning (QAP): iterative pruning can further reduce the hardware computational complexity of a quantized model
- After QAP, the 6-bit, 80% pruned model achieves a factor of 50 reduction in BOPs compared to the 32-bit, unpruned model

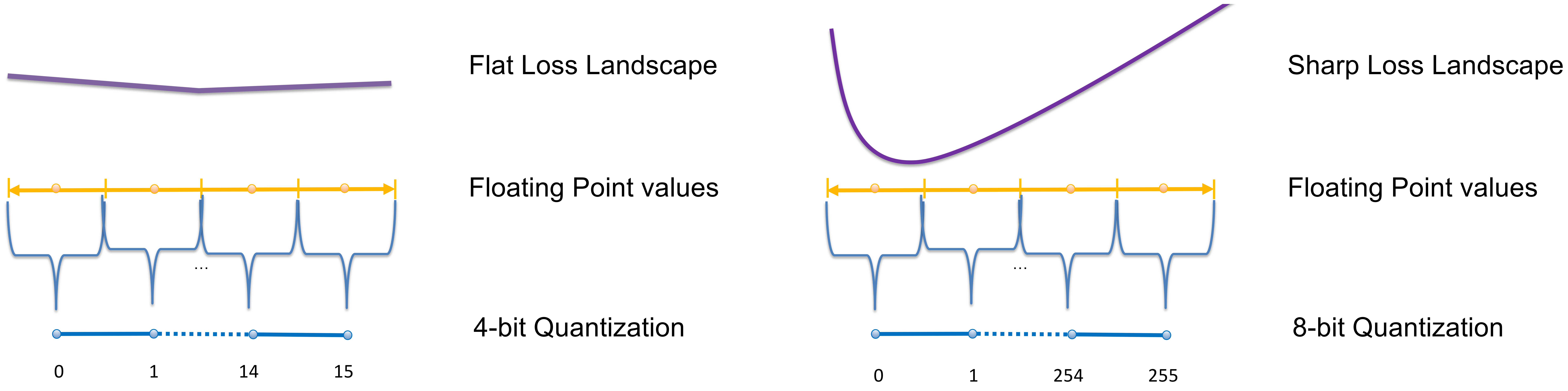


- Study using Brevitas

Bit operations (BOPs) definition: arXiv:1804.10969

Hessian-aware quantization (HAWQ) arXiv:2011.10680

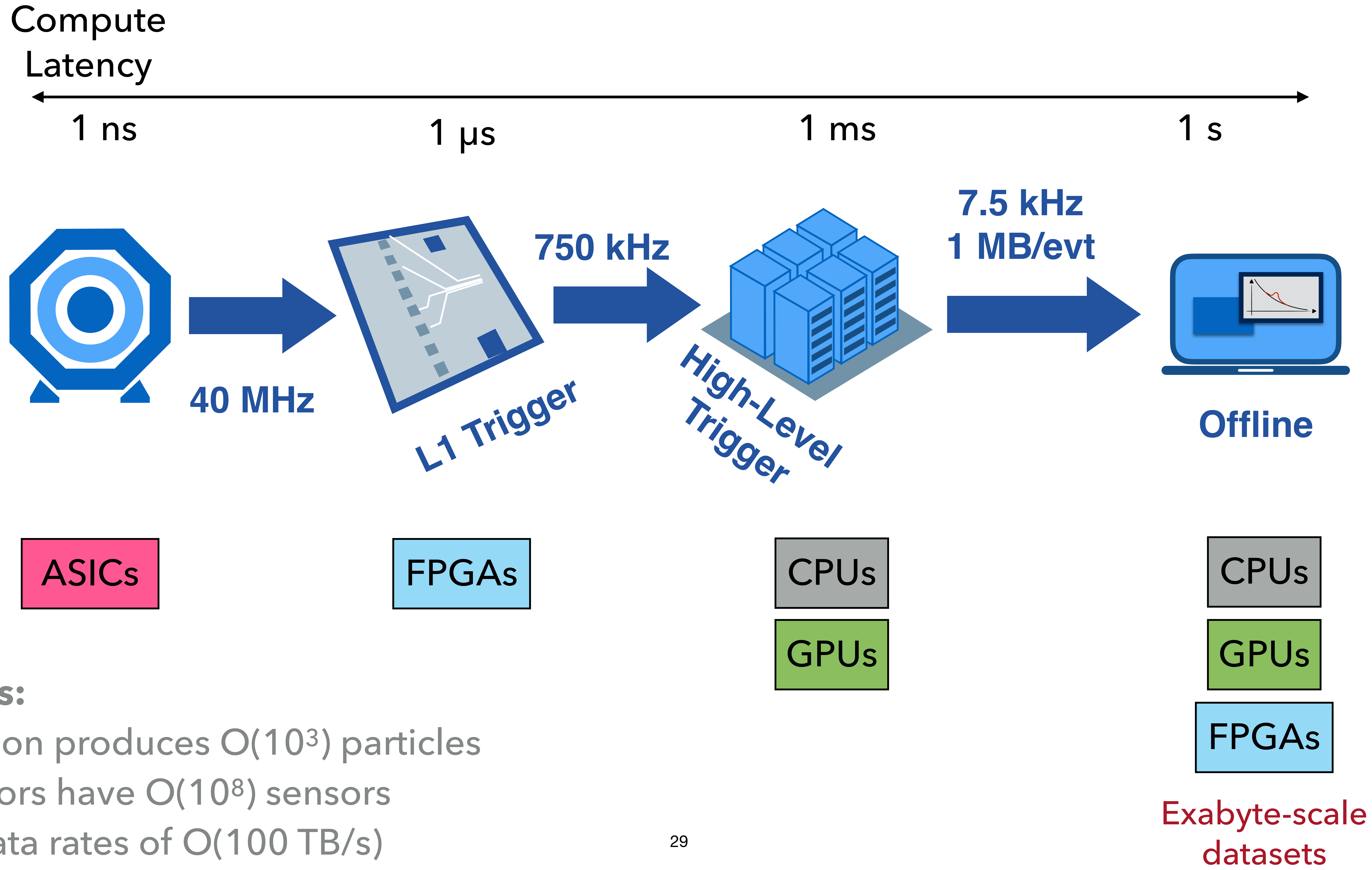
- Hessian of loss can provide additional guidance about quantization!
- Flat loss landscape: Lower bit width
- Sharp loss landscape: Higher bit width



COMPUTING HARDWARE ALTERNATIVES

Image: [Microsoft]

LHC event processing



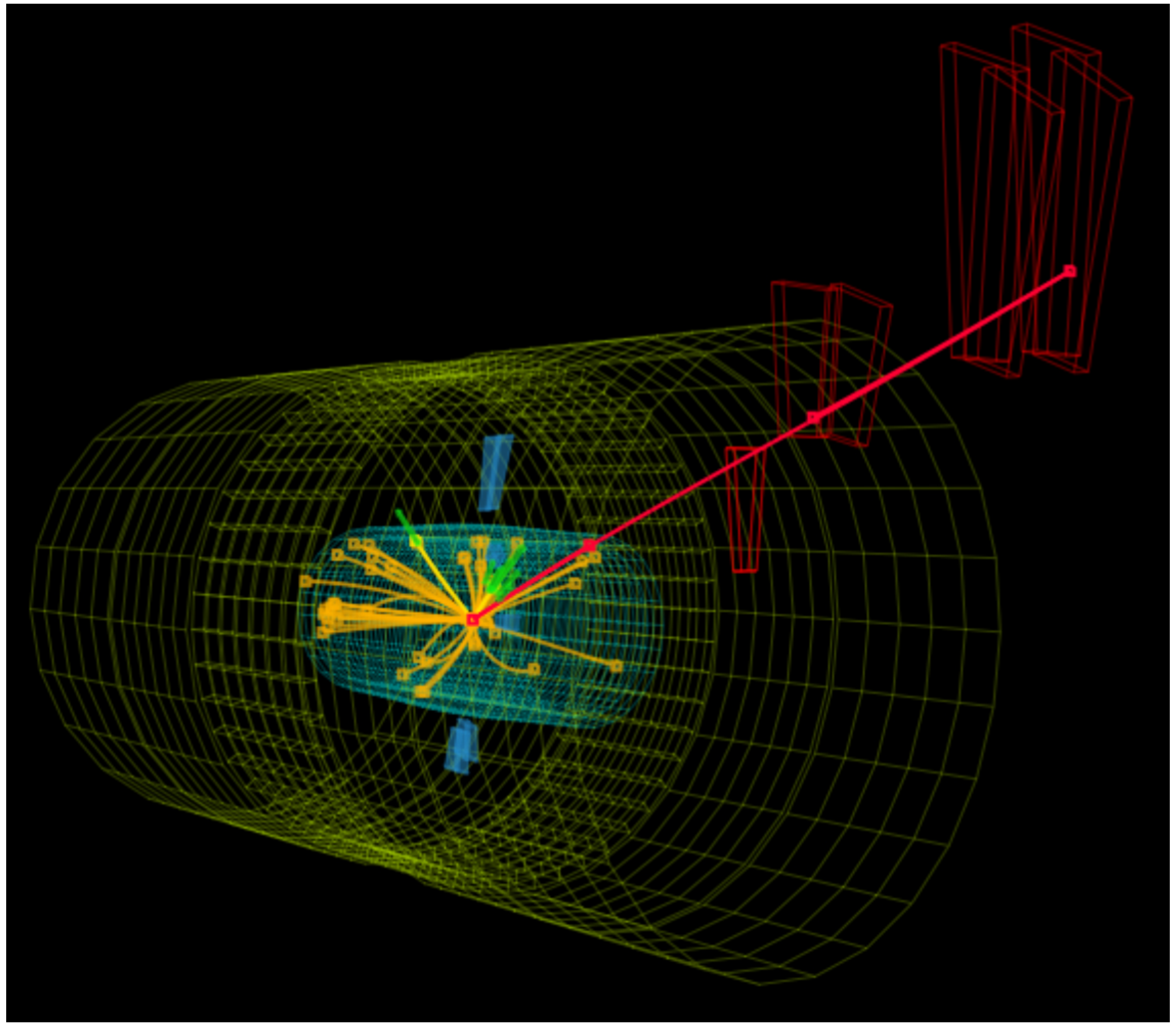
Challenges:

Each collision produces $O(10^3)$ particles

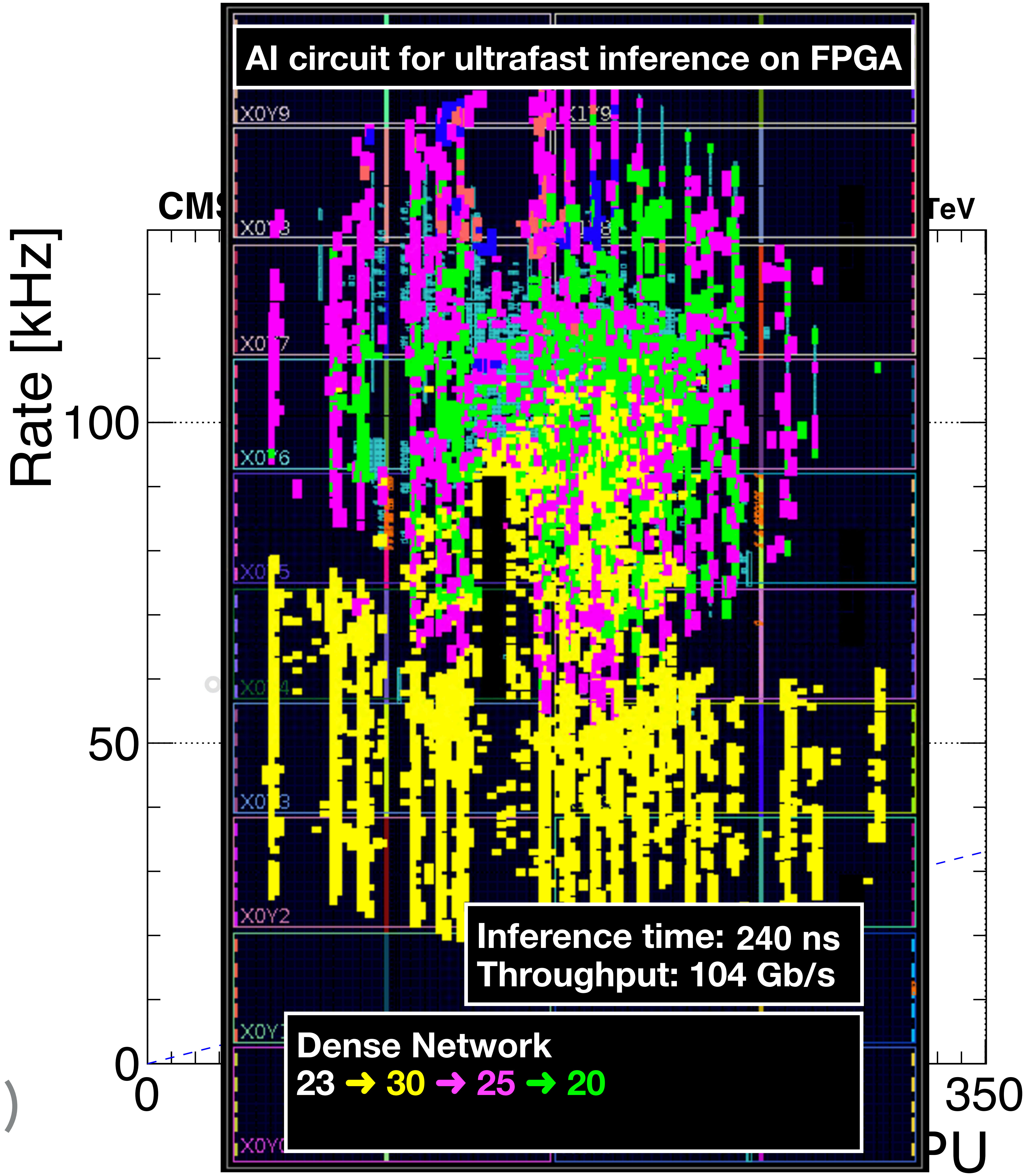
The detectors have $O(10^8)$ sensors

Extreme data rates of O(100 TB/s)

APPLICATION: MEASURE MUONS AT 40 MHZ



- NN measures muon momentum
  - 3× reduction in the trigger rate for NN!
- Fits within L1 trigger latency (240 ns!) and FPGA resource requirements (less then 30%)



Next time

- Knowledge distillation