ADVANCED PARALLEL COMPUTING LECTURE 11 - DNN INFERENCE OPTIMIZATIONS

Holger Fröning
holger.froening@ziti.uni-heidelberg.de
Institute of Computer Engineering
Ruprecht-Karls University of Heidelberg

MODERN ML

Training: ~ $O(10^{18})$ OPs
Inference: ~ $O(10^{9})$ OPs

Image & video: classification, object localization & detection

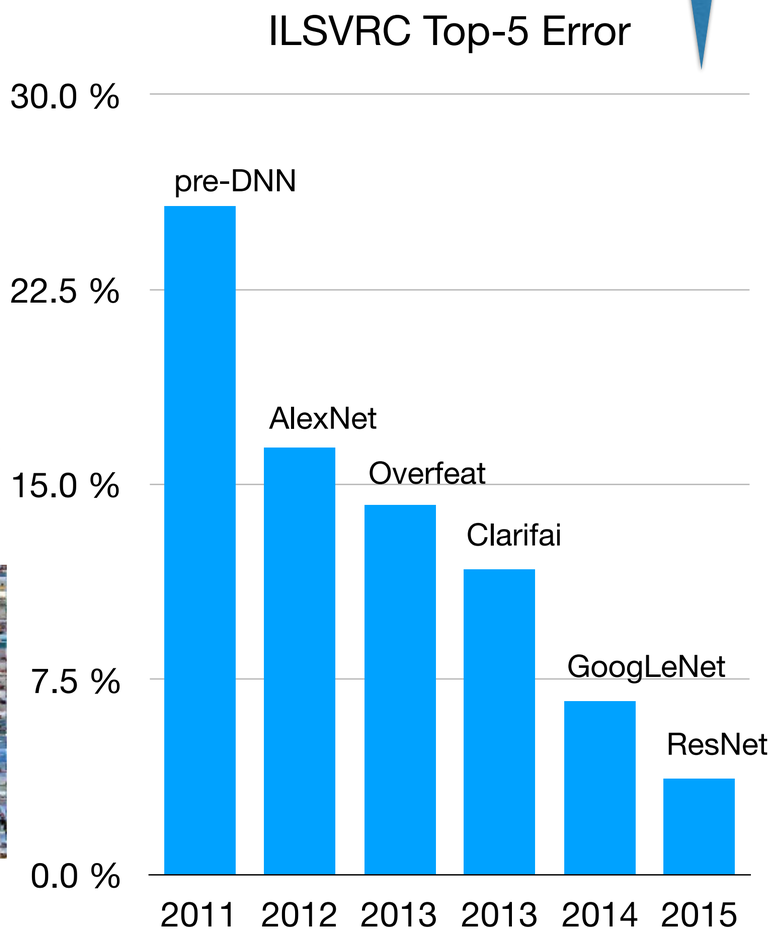
Speech and language: speech recognition, natural language processing

Medical: imaging, genetics of diseases

Various: game play, robotics

MNIST handwritten database

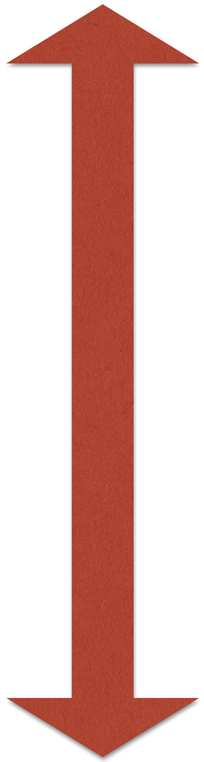
IMAGENET: 1000 classes



Artificial Neural Networks (ANNs) deliver state-of-the-art accuracy for many AI tasks
… at the cost of extremely high computational complexity

(PUBLIC) DATASET OVERVIEW

	Image size	Classes	Dataset size	SOTA error
MNIST	28x28x1	10	60,000 + 10,000	0.21% [1]
SVHN	Variable (32x32x3)	10	73,257 + 26,032 (+ 531,131)	1.69% [2]
CIFAR-10	32x32x3	10	50,000 + 10,000	96.53% [3] (accuracy)
CIFAR-100	32x32x3	100	50,000 + 10,000	75.72% [4] (accuracy)
ILSVRC2015	224x224x3	1000	14M	4.49% (TOP-5)/ 19.38% (TOP-1) [5]



Trains on my wimpy laptop in ~10min

Trains in ~10min, if you had 2k GPUs (ResNet-50, M40s)

[1] *Wan, L., Zeiler, M. D., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. ICML*

[2] *Lee, C., Gallagher, P. W., and Tu, Z. (2015). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. CoRR, abs/1509.08985.*

[3] *Graham, B. (2014). Fractional max-pooling. CoRR, abs/1412.6071.*

[4] *Clevert, D., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). CoRR, abs/1511.07289.*

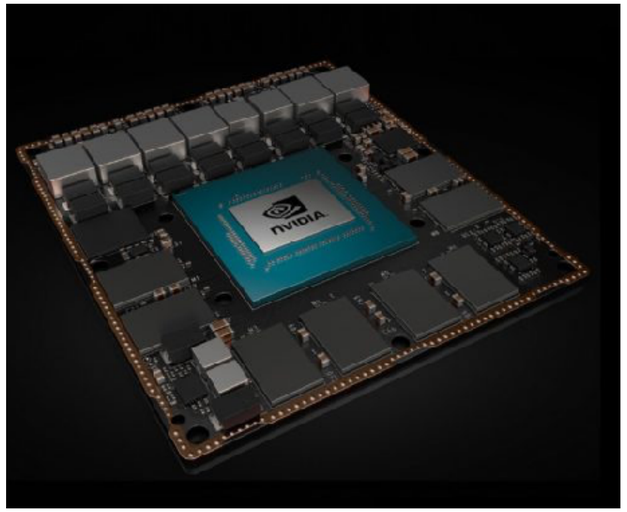
[5] *He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. CoRR, abs/1512.03385.*

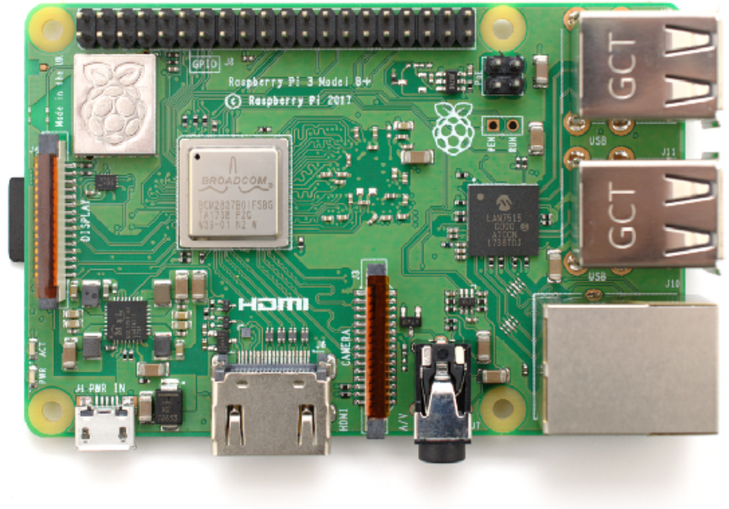
ANN OVERVIEW

	LeNet 5	AlexNet	Overfeat fast	VGG-16	GoogLeNet v1	ResNet 50	ResNet 152
Top-5 error [%]	n/a	16.4	14.2	7.4	6.7	5.3	4.5
# CONV layers	2	5	5	13	57	53	155
Weights	2.6k	2.3M	16M	14.7M	6.0M	23.5M	58M
MACs	283k	666M	2.67G	15.3G	1.43G	3.86G	11.3G
# FC layers	2	3	3	3	1	1	1
Weights	58k	58.6M	130M	124M	1M	2M	2M
MACs	58k	58.6M	130M	124M	1M	2M	2M
Total weights	60k	61M	146M	138M	7M	25.5M	60M
Total MACs	341k	724M	2.8G	15.5G	1.43G	3.9G	11.3G

FORWARD PATH ONLY. ADDITIONAL LAYERS (POOLING, BATCH NORMALIZATION, …) AND ACTIVATION FUNCTION NOT INCLUDED.

PROCESSOR ZOO





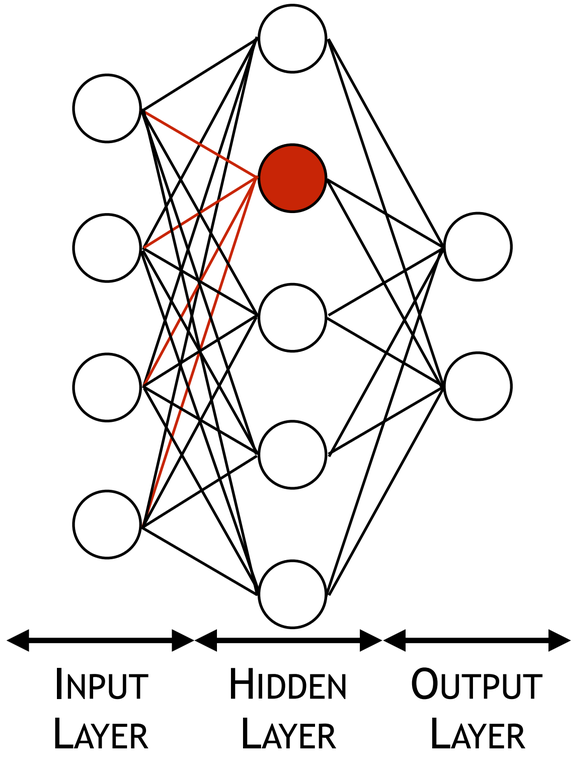
	NVIDIA Xavier	XILINX Zynq Ultrascale+ ZU19EG	Raspberry Pi 3 B+
Wattage	30W	~10W	6W
Peak GFLOP/s	1,300 (325 images/s[1])	difficult	5.6 (1.4 images/s[1])
Total memory	16GB	2GB	1GB
In-core memory	2.9MB (2.8%[2])	4.3MB (4.2%[2])	2.3MB (2.3%[2])

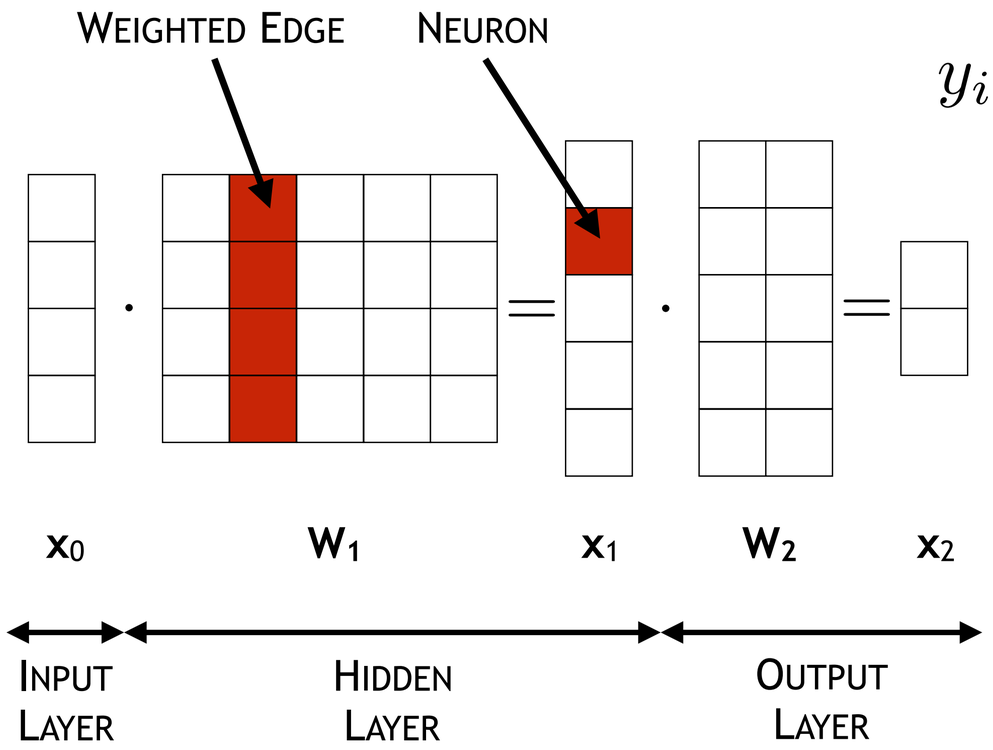
[1] based on theoretical peak GFLOP/s performance, [2] weights only, both for ResNet-50

Extreme mismatch between ANN complexity and mobile processor capability

ANN BACKGROUND

MULTI-LAYER PERCEPTRON





$$y_i = f(\sum_j (w_{j,i} \cdot x_j) + b_i)$$

$$\mathbf{x_n} = f(\mathbf{W_n} \cdot \mathbf{x_{n-1}})$$

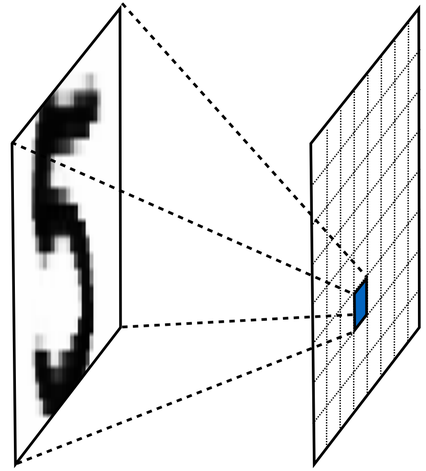
f: non-linear function (sigmoid, reLU, …)
W: weight matrix
x: input vector
b: bias vector (skipped)

MNIST: 28x28 images in 10 classes => MLP with 28x28 inputs & 10 outputs

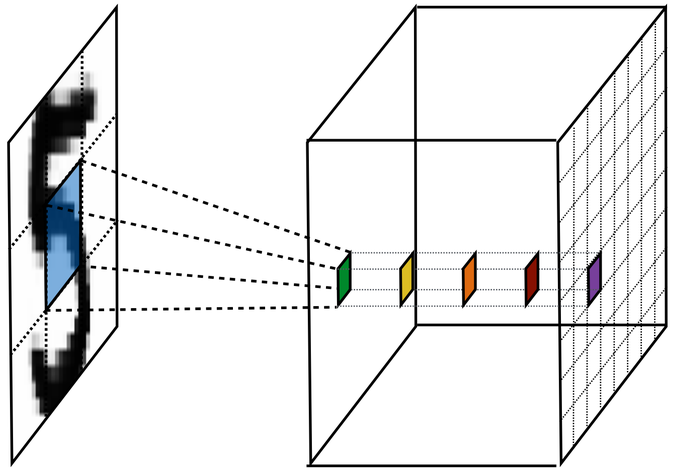
Foundation: fully-connected layer => matrix-vector operation

Parallel training/inference in batches: matrix-matrix operation (x becomes matrix)

CONVOLUTIONAL LAYERS



Fully-connected layer



Convolutional layer

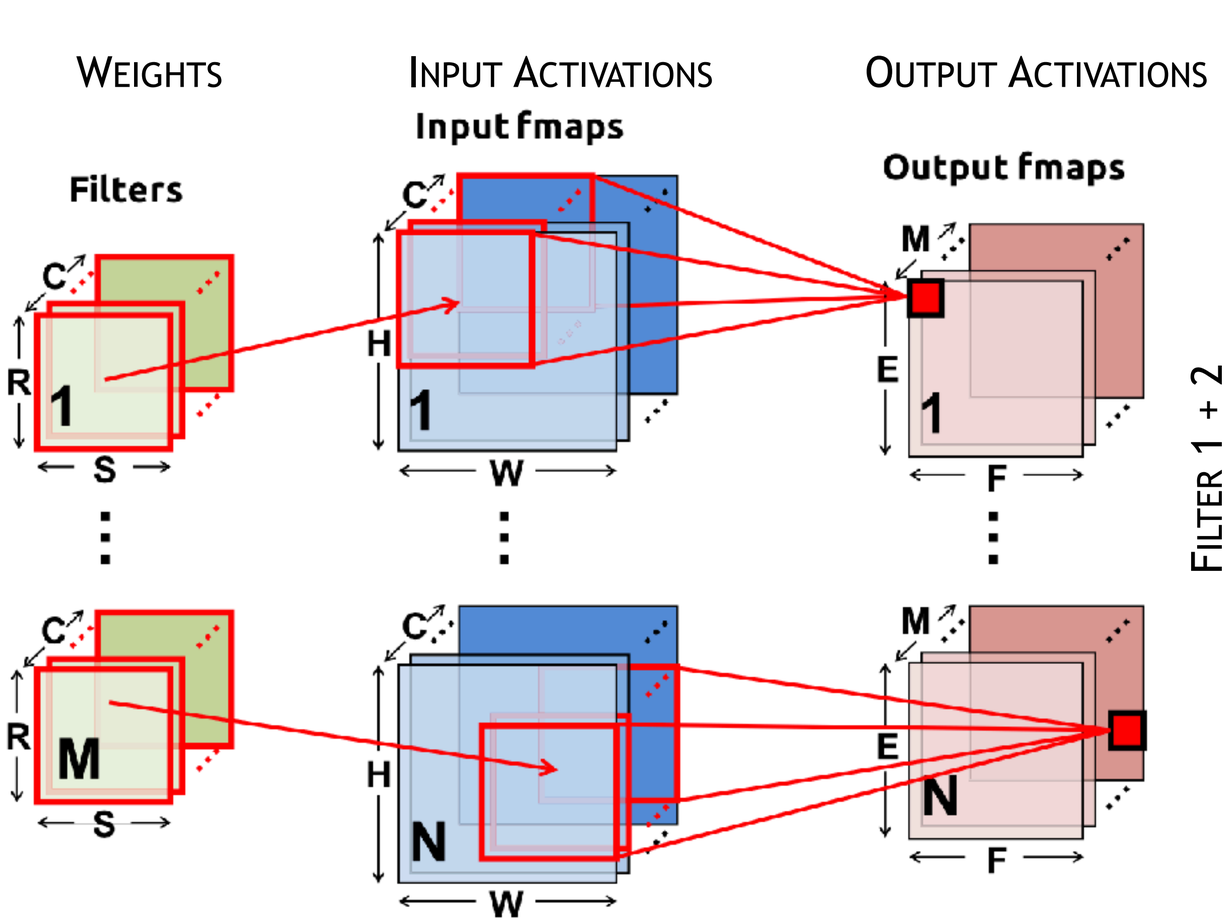
Receptive field: spatially local correlation (patches)

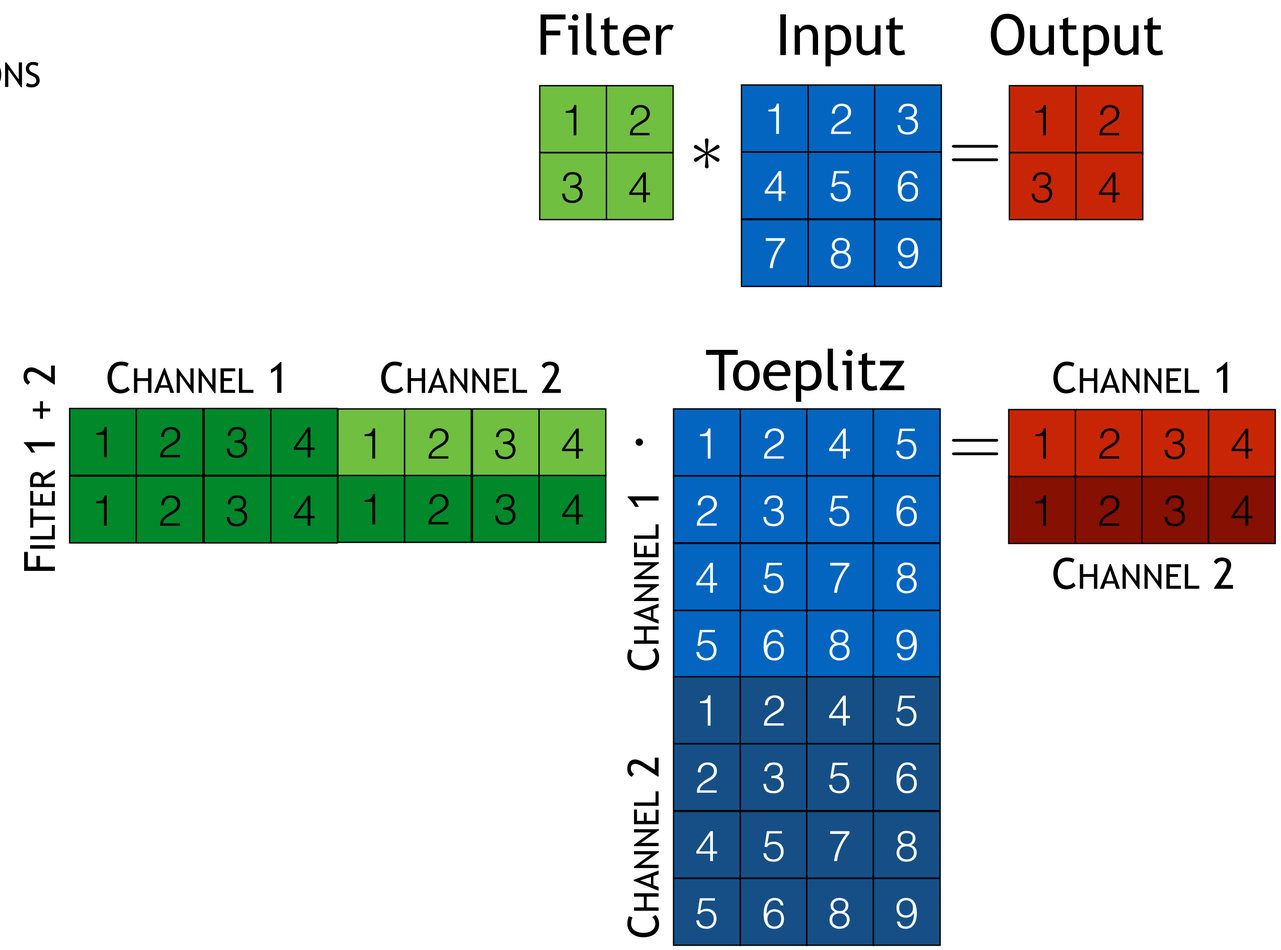
Shared weights: as each filter is applied to all patches of the input

3D layers: “depth” of one layer is the number of filters (kernels) learned

=> Even higher computational intensity

CONVOLUTION OPERATION [10]

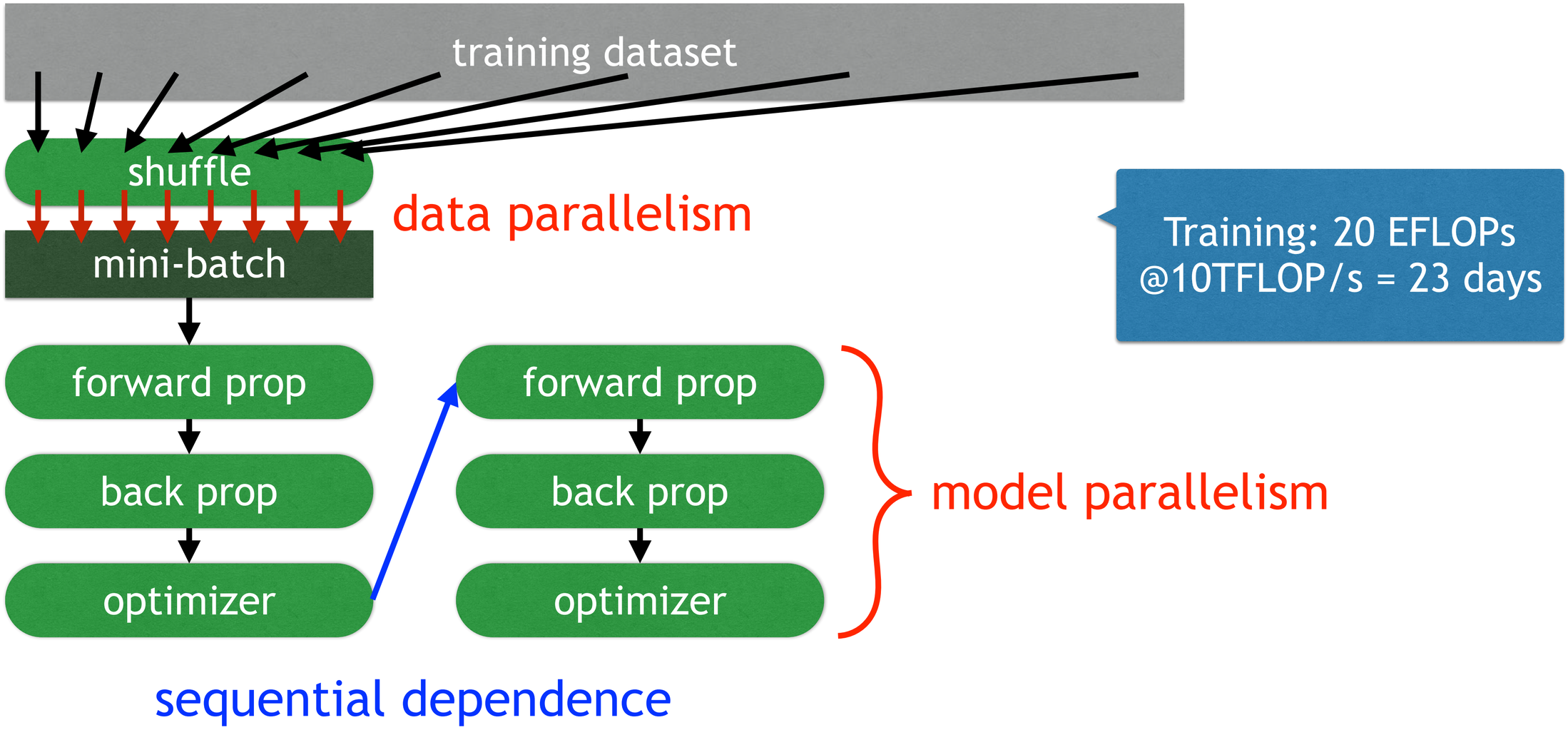




Convolutions increase data reuse, but are usually still mapped to matrix operations

V. Sze, T.-J. Yang, Y.-H. Chen, J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, December 2017.

TRAINING OF DEEP NEURAL NETWORKS



BACKWARD PATH

Exemplary error function E as loss L

- i^{th} training example
- $t^{(i)}$: true answer
- $y^{(i)}$: result from NN

$$E = \frac{1}{2} \cdot \sum_i (t^{(i)} - y^{(1)})^2$$

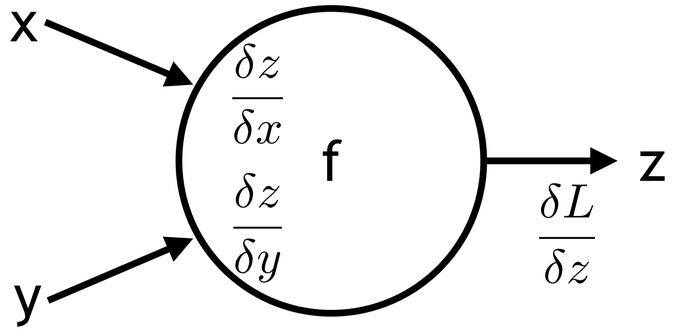
Gradient descent

- Multiply steepness with learning rate
- Stochastic gradient descent: use randomization and noise to find global minimum

Backpropagation

- Propagate error backwards through the network: partial derivatives & chain rule
- Usually an operation based on a Jacobian (J) matrix multiplication

$$\frac{\delta L}{\delta x} = \frac{\delta L}{\delta z} \cdot \frac{\delta z}{\delta x}$$



$$\frac{\delta L}{\delta y} = \frac{\delta L}{\delta z} \cdot \frac{\delta z}{\delta y}$$

$$J_{i,j} = \frac{\delta f_i}{\delta x_j}$$

PLETHORA OF ANN PROCESSORS

Research

Eyeriss, EIE/ESE, FPGAConvNet, NeuroCube, *DianNao, NEURAghe, FINN, … (lost track)

Commercial

Google TPU/Edge TPU

NVIDIA GPU specializations (TensorCores)

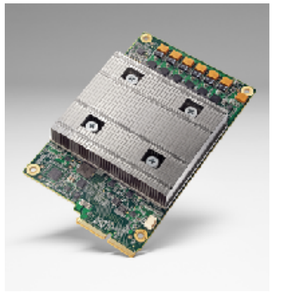
Intel Mobileye EyeQ5 & Nervana, Loihi (neuromorphic)

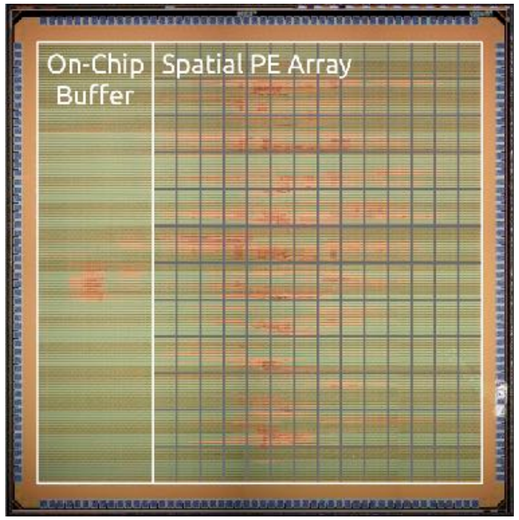
More: Movidius MyriadX VPU, ARM Trillium, GraphCore IPU, CEVA NeuPro, IBM TrueNorth, …

Basically any smartphone manufacturer

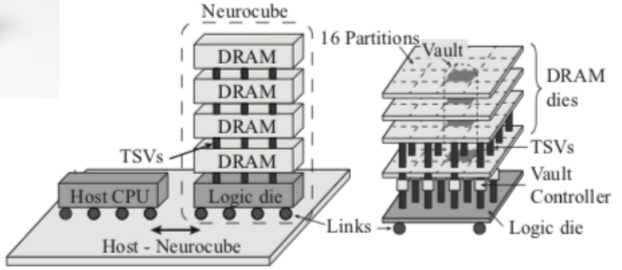
Architecture: array of processing elements (PE) most promising (PE array)

Main goal: minimize data movements





http://eyeriss.mit.edu



Neurocube

DNNS SIMPLICITY WALL

Simplicity wall: DNNs spend most of their time in matrix multiplications

- Predictability, static loop-trip counts, little control overhead

Safe optimizations: use without restraints, no implication towards model's/workload's accuracy

- Shorter communication paths
- Data reuse to minimize data volume being transferred
- => Dedicated architectures

Unsafe optimizations: potential implications towards model's/ workload's accuracy

- Reduce number of operations & model size: compression, pruning
- Reduce precision of operations and operands: quantization (fixed point, binarization)

Brandon Reagen; Robert Adolf; Paul Whatmough; Gu-Yeon Wei; David Brooks; Margaret Martonosi, "Deep Learning for Computer Architects," in Deep Learning for Computer Architects , 1, Morgan & Claypool, 2017, doi:10.2200/S00783ED1V01Y201706CAC041

QUANTIZATION AS UNSAFE OPTIMIZATION

QUANTIZATION PRIMER

Quantizer Q: piece-wise constant function

Input values in given quantization interval mapped to corresponding quantization level

Apply to activations/weights(/gradients)

$$Q(x) = q_l, \text{ if } x \in (t_l, t_{l+1}]$$

quantization level l (L total)

quantization intervals

Uniform quantization if all levels are equidistant

$$q_{i+1} - q_i = \Delta, \forall i$$

where Δ is a constant quantization step

Keep activation function in mind when quantizing

$$Q(x) = \begin{cases} +1 : x \geq 0 \\ -1 : x < 0 \end{cases}$$

Example for binary quantization (sign function)

EXAMPLE QUANTIZATION USING K BITS

Real number

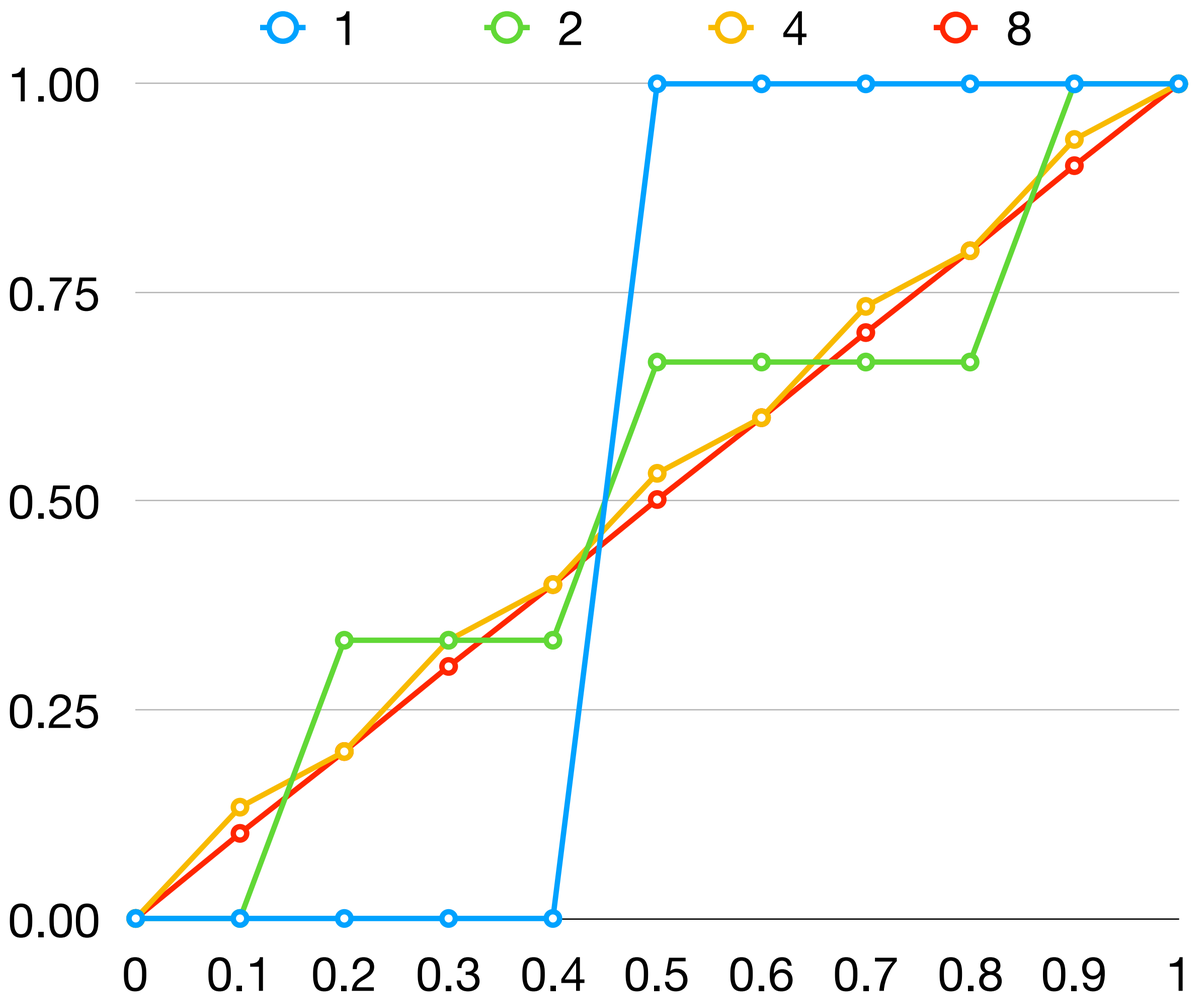
$$a_i \in [0, 1]$$

k-bit fixed-point integer

$$a_i^q \in [0, 1]$$

Quantizer

$$a_i^q = \frac{1}{2^k - 1} round((2^k - 1)a_i)$$



RELATED WORK QUANTIZATION

SW quantization concepts

	Weights	Activations
BNN [3]	[-1,+1]	[-1,+1]
XNOR [2]	[-S,+S]	[-1,+1]
DoReFa [1]	[-S,+S]	[0,+1]
TWN [7]	[-S,0,+S]	float32
TTQ [8]	[-Sn,0,+Sp]	float32
HWGQ [9]	XNOR [2]	2bit

AlexNet/ImageNet accuracy for state-of-the-art quantization

A-W	DC	BNN	XNOR	DoReFa	TWN	TTQ	HWGQ
32-32	80.3	80.2	80.2	80.3	80.3	80.3	81.5
32-8/5	80.3	-	-	-	-	-	-
32-2	-	-	-	-	76.8	79.7	-
32-1	-	-	-	76.3	-	-	-
2-1	-	-	-	-	-	-	76.3
1-1	-	50.4	69.2	69.3	-	-	-

Key observation: DNNs contain plenty of redundancy
Nonuniform quantization outperforms uniform quantization

NON-UNIFORM QUANTIZATION

Uniform quantization has a limited model capacity

- Easy to store ($\log_2 L$ bits without the quantization levels)
- Easy to compute if activations and weights quantized identically (again $\log_2 L$ bits)

Non-uniform quantization improves model capacity

- Storage: $\log_2 L$ bits plus the levels
- Computation: q_l has to be used

Trainable quantization levels (scaling factors) to to adapt to weights/activations

- Full-precision weights required for training
- Re-training usually helps to partially recover from quantization error

$$w_l^i = \begin{cases} W_l^p : w_l > \Delta_l \\ 0 : |w_l| \leq \Delta_l \\ -W_l^n : w_l < -\Delta_l \end{cases}$$

$$\Delta_l = t \cdot max(|w|); t \in [0, 1]$$

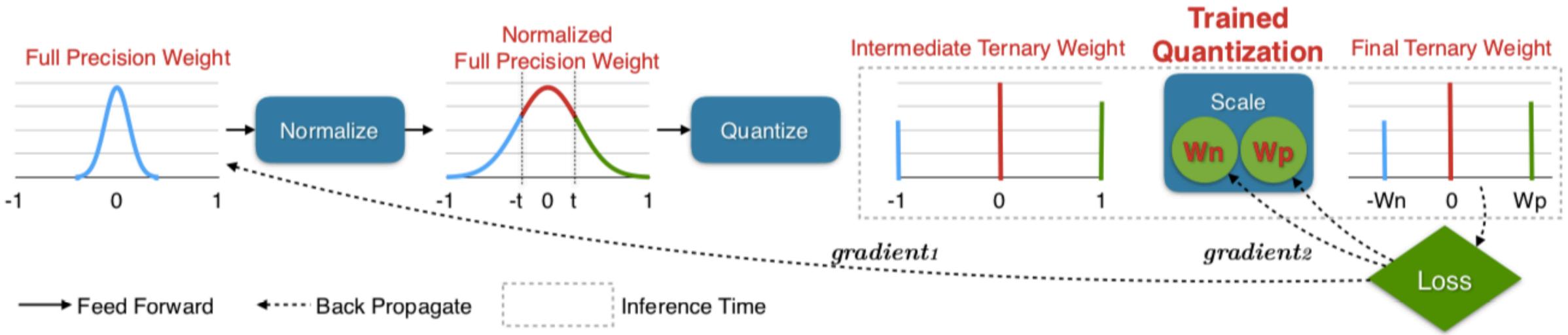
TRAINED TERNARY QUANTIZATION [8]

Train full-precision weights & train scale factors for ternary weights (hyperparameter t)

1. Normalization: weights in range [-1, +1]
2. Quantization by thresholding: {-t, 0, +t}
3. Learning ternary assignments: gradient to full-resolution weights
4. Learning ternary values: gradient to scaling factors

$$w_l^i = \begin{cases} W_l^p : w_l > \Delta_l \\ 0 : |w_l| \leq \Delta_l \\ -W_l^n : w_l < -\Delta_l \end{cases}$$

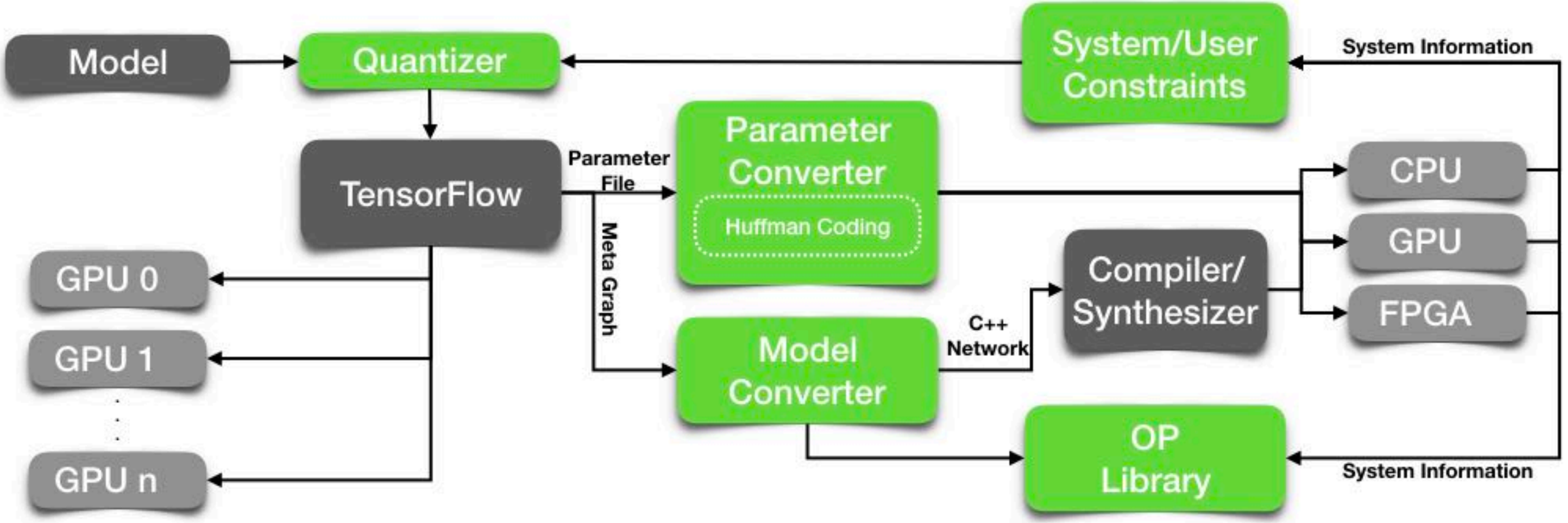
$$\Delta_l = t \cdot max(|w|); t \in [0, 1]$$



Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. CoRR, abs/1612.01064, 2016. URL http://arxiv.org/abs/1612.01064

DEEPCHIP’S QUANTIZATION CONCEPT

DEEPCHIP: SOFTWARE ARCHITECTURE



Günther Schindler, Matthias Zöhrer, Franz Pernkopf, and Holger Fröning, Towards Efficient Forward Propagation on Resource-Constrained Systems, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2018).

QUANTIZATION

Weight quantization to ternary values according to TTQ [8]

- Scale factors $\{W^p, W^n\}$: independent + asymmetric, trained using SGD
- Hyperparameter t => trading among accuracy and space

$$w_l^i = \begin{cases} W_l^p : w_l > \Delta_l \\ 0 : |w_l| \leq \Delta_l \\ -W_l^n : w_l < -\Delta_l \end{cases}$$

$$\Delta_l = t \cdot max(|w|); t \in [0, 1]$$

Bounding activations, quantization to fixed point (flexible bit-width k, [1])

- Bounded ReLU => $0 <= a_i <= 1$
- cf. TTQ using floating point

$$a_i = \begin{cases} 0 : \tilde{a}_i \leq 0 \\ \tilde{a}_i : 0 < \tilde{a}_i < 1 \\ 1 : \tilde{a}_i \geq 1 \end{cases}$$

$$a_i^q = \frac{1}{2^k - 1} round((2^k - 1)a_i)$$

PARAMETER CONVERTER

Space-efficient data structures

Run-length encoding of weight matrix

- Non-zero values + signs
- Only sign and distance vector stored
- => Reduced cardinality

Compression using Huffman coding (not shown)

$$\mathbf{W}_l^T = \begin{pmatrix} 0 & W_l^p & W_l^p & 0 & W_l^n \\ W_l^n & 0 & 0 & W_l^p & 0 \\ W_l^p & W_l^n & 0 & W_l^n & W_l^n \end{pmatrix}$$

$$\mathbf{I}_l^p = \begin{pmatrix} 1 & 2 & - \\ 3 & - & - \\ 0 & - & - \end{pmatrix} \qquad \mathbf{I}_l^n = \begin{pmatrix} 4 & - & - \\ 0 & - & - \\ 1 & 3 & 5 \end{pmatrix}$$

Signs $\mathbf{s}_l = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$

Indices $\mathbf{i}_l = \begin{pmatrix} 1 & 2 & 4 & 5 & 8 & 10 & 11 & 13 & 14 \end{pmatrix}$

Distance $\mathbf{d}_l = \begin{pmatrix} 1 & 1 & 2 & 1 & 3 & 2 & 1 & 2 & 1 \end{pmatrix}$

OPERATOR LIBRARY - REDUCE & SCALE

Saving complexity

1. Reduced precision
2. Sparsity
3. Only partial sums and two multiplications

$$c = \sum_{i=1}^{N} a_i \cdot b_i, \quad a_i, b_i \in \mathbb{R} \quad \forall i$$

$$c = W_l^p \cdot \sum_{i \in \mathbf{i}_l^p} a_i + W_l^n \cdot \sum_{i \in \mathbf{i}_l^n} a_i, \quad where$$

$$\mathbf{i}_l^p = \{i | b_i = W_l^p\} \quad \text{and} \quad \mathbf{i}_l^n = \{i | b_i = W_l^n\}$$

MULT vs. ADD

Instruction	Cycles [ARM] (normalized)	Energy (pJ) [13]
float32 FMA	8.0	4.6
int16 FMA	3.0	1.6
int16 ADD	1.5	0.05

AlexNet/ImageNet

	Baseline	BNN	INT8	DeepChip
Top-5 Accuracy [%]	78.3	56.4	-[1]	79.0
Sparsity [%]	0.0	0.0	0.0	63.0
Inference Rate [FPS]	4	22	7	8
Memory [MB]	244	24	61	25

[1] *Authors claim no change in accuracy*

N-ARY QUANTIZATION (NAQ)

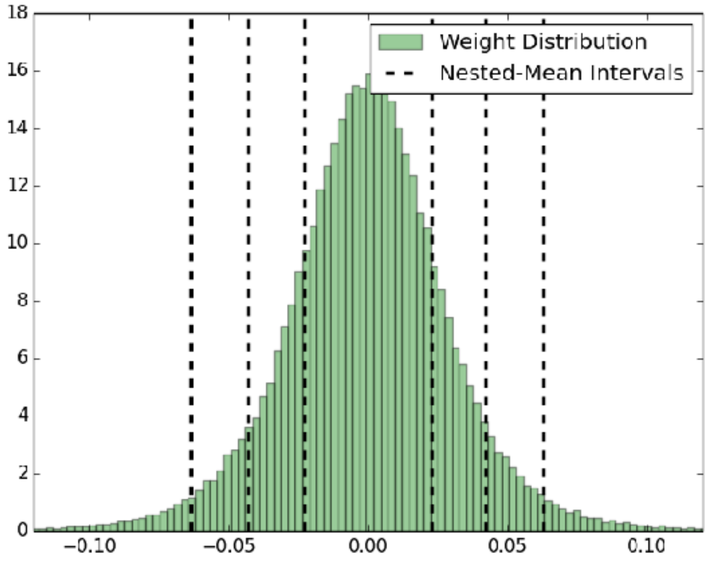
Up to now: all good for ConvNet+SVHN, AlexNet+ImageNet, ResNet-44+CIFAR-10

I.e., complex model + simple data, or simple model + complex data

But: quantization depends on complexity(data) & complexity(model)

Non-uniform n-ary weight representations

Multiple scale factors, nested means for clustering



ResNet-18/ImageNet

	Weights [bit]	Activations [bit]	Training	Top-5 [%]
LQNet [16]	2	2	2.3x	85.9
DeepChip - ternary	2	2	1.2x	86.7
LQNet [16]	3	32	1.7x	88.8
DeepChip - quinary	3	32	2.0x	89.0

DISCUSSION

FLOATING/FIXED POINT ARITHMETICS

Fixed point

add: energy, area ~ N [bits]

mult: energy, area ~ N^2 [bits]

- dynamic range is small

+/- less complexity - slightly?

Floating point

energy, area: complex

+ huge dynamic range

- distribution: [-1,1] contains half the numbers

- more complex

bfloat16 (Google/Intel): introduced because of dynamic range

	Sign S	Exponent E	Significand F	Range
float64	1b	11b	52b	~2x10$^{+/-308}$
float32	1b	8b	23b	~2x10$^{+/-38}$
float16	1b	5b	10b	~1x10$^{-4/+5}$
bfloat16	1b	8b	7b	=float32
POSIT	1b		variable	

$$v = (-1)^S \cdot (1 + F) \cdot 2^{(E-bias)}$$

$$P_S = A_S \oplus B_S$$

$$P_E = A_E + B_E$$

$$P_F = A_F \cdot B_F$$

THERE IS MORE THAN FLOATING/FIXED POINT ARITHMETIC

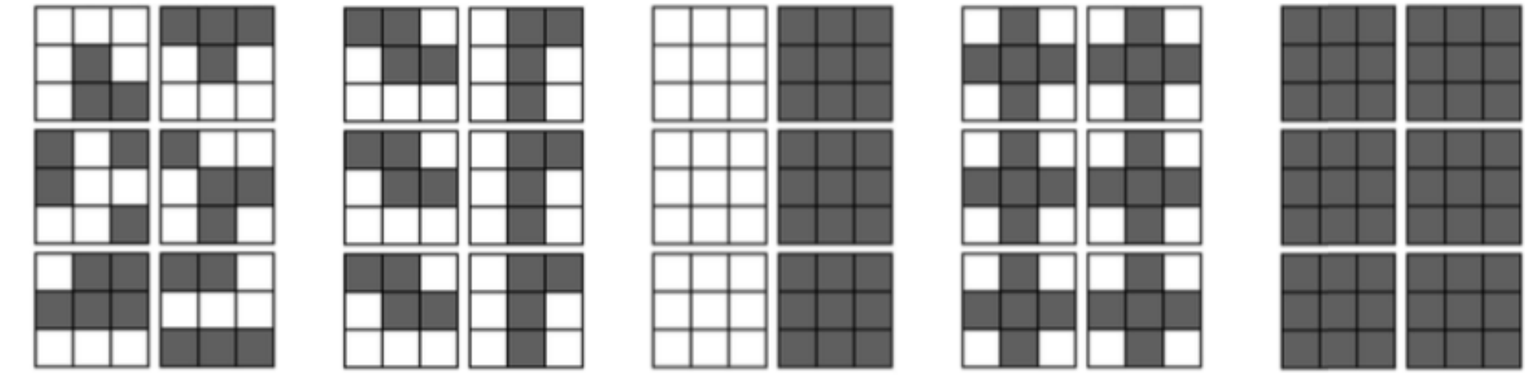
PRUNING

Pruning connections achieves impressive theoretical compression rates

Issues of resulting fine-grained sparsity

- Compression rates are typically reported without indexing overhead
- Load imbalance (lack of structured parallelism)
- Random memory accesses (predictability issues result in poor caching effectivity)

=> Structured pruning



(a) Weight pruning (b) Column pruning (c) Channel pruning (d) Shape pruning (e) Layer pruning

BEYOND DATA PARALLELISM

Data parallelism (DP): replicate models, distribute data based on mini-batch size

- Training: forward + gradient + backwards + optimizer
- Optimizer: hyperparameters, learning rate, etc.
- Communication: all2all & allreduce, possibly asynchronous (Hogwild! et al.)
- Scalability: mini-batch size times processor count = effective mini-batch size

Model parallelism (MP): layer-wise parallel approach

- Communication: dependencies between layers
- Scalability: too much communication

Main reason to not use DP: model size > GPU memory

NEUROMORPHIC COMPUTING

- ANNs somewhat inspired by the brain
  - Convolutions
  - Attention module for time series (Google Transformer)
  - ANN neurons != biological neurons
  - Human brain’s estimated compute capability of 1EF is nonsense (20W btw)
- Neuromorphic computing: spiking neural networks (SNN)
  - Neurons have much more similarities with the biological brain
- Main issue with neuromorphic computing?
  - SGD not applicable as SNNs are not differentiable
  - Probably: many small things are better than few complex ones

MY TAKE ON MACHINE LEARNING

Its computational complexity demands for dedicated methods & tools

- Safe: optimized data re-use, minimal data movement distances
- Unsafe: quantization, pruning, (lossy) compression

Ultimately there will be (integrated) ML compute stacks

- ALUs, architecture, ISA, IR, compiler, DSL

ML is not a single application, it a spectrum of applications

- Behavior = data + model (cf. graph computations)
- Even true for ANNs

Fundamental technique that any computer scientist should understand

- Possibilities and limitations

SUMMARY

- Artificial NNs are universal functional approximators
  - Deep (many layers)
  - Thin (few parameters per layer)
  - Multi-branch (Inception, ResNet, DenseNet)
- Stochastic Gradient Descent (SGD) drives the success of ANNs
  - Robust: tolerant against mistakes (hyperparameters, model architecture)
  - Mind the mini-batch size (computational intensity (+), reduces variance of updates (+), memory requirement (-), convergence problem if too large (-))
  - Mind the scalability issues
- ANNs drive computer architecture
  - Various commercial (and academic) examples
  - Prime example for: App+Data defines Architecture