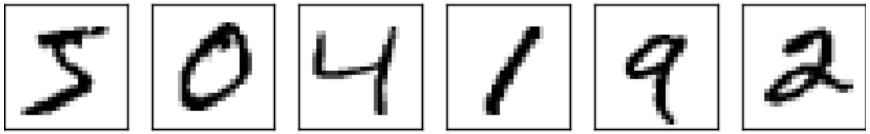
Lecture: Deep Networks and CNNs

- Topics: wrap-up of deep networks, accelerator basics,
  Diannao accelerator

- Resources: http://www.cs.utah.edu/~rajeev/cs7960/notes/
- Canvas/registration

Deep Networks for Image Classification

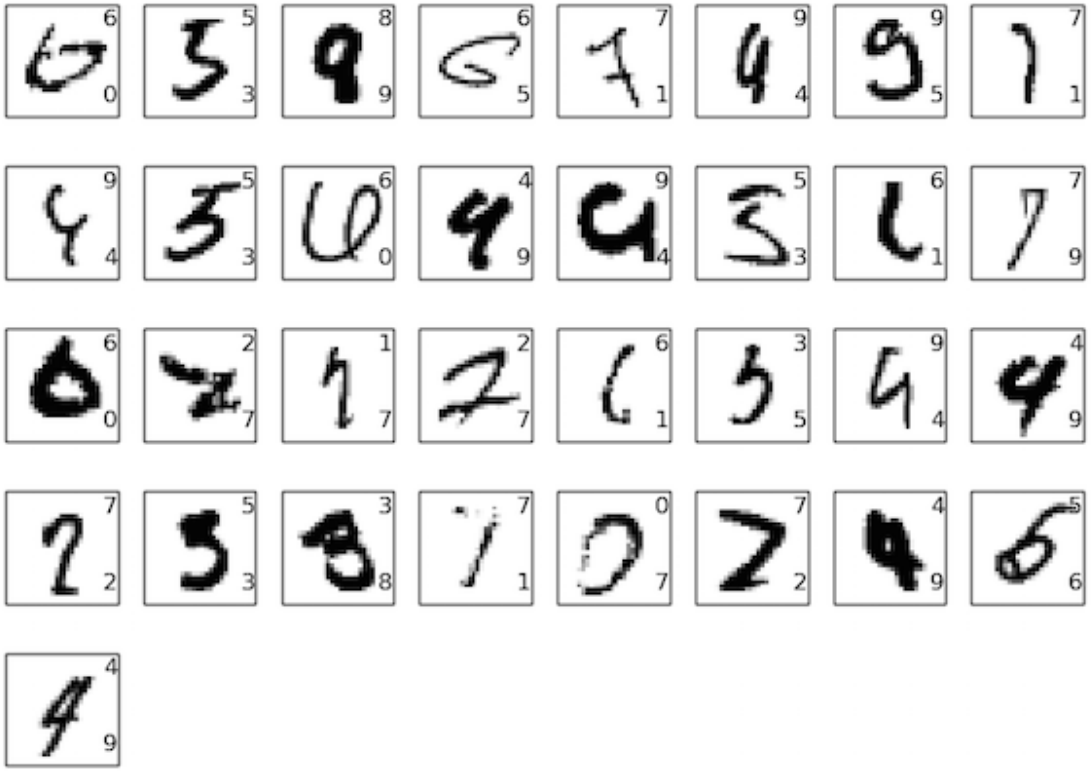
- MNIST: 784-pixel images of hand-written digits; 50K training images; 10K testing images



- ILSVRC: e.g., 1000 categories, 1.2 million training images, 150K test images, top-5 criterion

Deep Learning Success

- Modern deep learning is better than humans on both MNIST (>99%) and ILSVRC top-5 (>95%)



Accuracies

- Baseline: single hidden layer with 100 neurons: 97.8%
- CNN: added 20 5x5 filters and a 2x2 max-pool: 98.78%
- CNN: added 40 20x5x5 filters and 2x2 max-pool: 99.06%
- CNN: substitute sigmoid with ReLU: 99.23%
- CNN: expand the training data: 99.37%
- CNN: adding two fully-connected layers: 99.43%
- CNN: 1000 neurons in fully-connected layers: 99.47%
- CNN: adding dropout: 99.60%
- CNN: voting among an ensemble of 5 nets: 99.67%

(ILSVRC winners have 152 layers)

Many ways to avoid over-fitting in fully-connected networks (conv layers don't need these because the weights are shared): L2 regularization, dropout, expanded inputs.

Glossary

- Sigmoid activation function: $f(x) = 1/(1+e^{-x})$ – provides a smooth step function; tanh is similar.
- ReLU activation function: f(x) = max (0,x) – it allows the output to grow larger than 1, and has a larger σ' of 0 or 1.
- Softmax activation function: $f(z_j) = e^{zj}/\Sigma_k e^{zk}$ -- it's normalizing the neuron outputs in a layer so they sum to 1 and the largest neuron sticks out
- Feature map and filter: a filter or kernel is the grid of weights; a feature map is the resulting set of values when a filter is applied to a set of inputs
- Max pooling: extracts the largest value in a 2D input grid
- L2 pooling: computes the square root of the sum of squares of the values in a 2D input grid
- L2 regularization: includes the weights in the cost function during training so we're trying to not only reduce the error, but also the values of the weights
- Expanded inputs: to avoid over-fitting, the (say) 50K training images are expanded to 250K images. Each image is shifted slightly to the left/right/top/bottom.
- Dropout: some activation functions are randomly dropped during training (to avoid overfitting).
- DNN/CNN: CNNs use shared kernels for all neurons in a feature map, while DNNs use private kernels for each neuron in a feature map.
- SVM: a mathematical approach to incrementally define hyperplanes that separate clusters – an alternative way to classify inputs into different categories.

LSTMs

- The neurons in MLPs and CNNs do not have state – every input image results in brand new computations with no memory of previous images.

- An LSTM is better suited for speech/text processing where interpreting a new syllable or pronoun may depend on past inputs.

- Recurrent neural networks (RNNs, where a layer's output feeds back as input for the next computation) have evolved into LSTMs

LSTMs

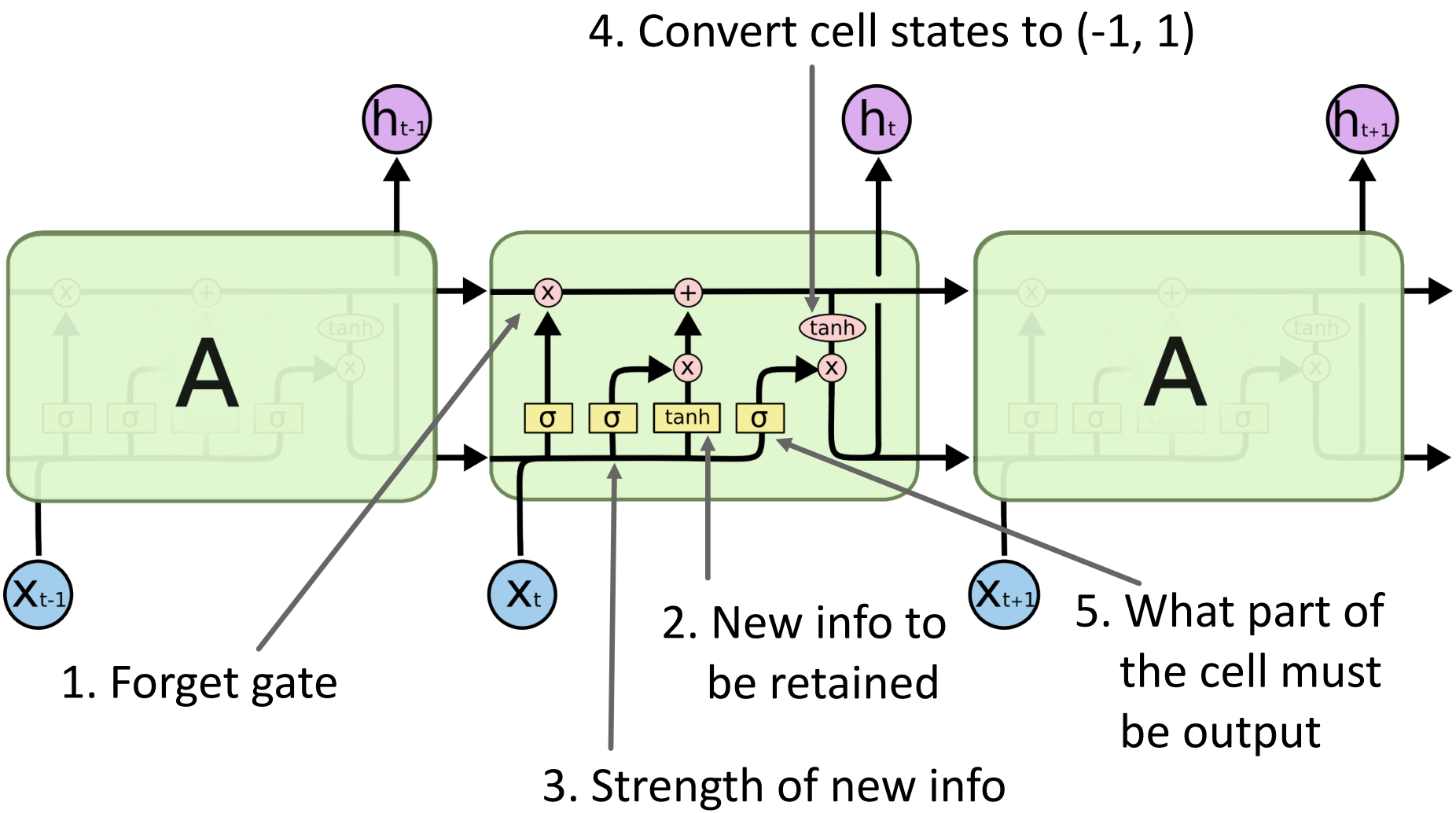


Image credit: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Accelerator Basics

- Software optimizations: loop ordering, tiling
- Hardware techniques: prefetching, SIMD units, near-data processing, …

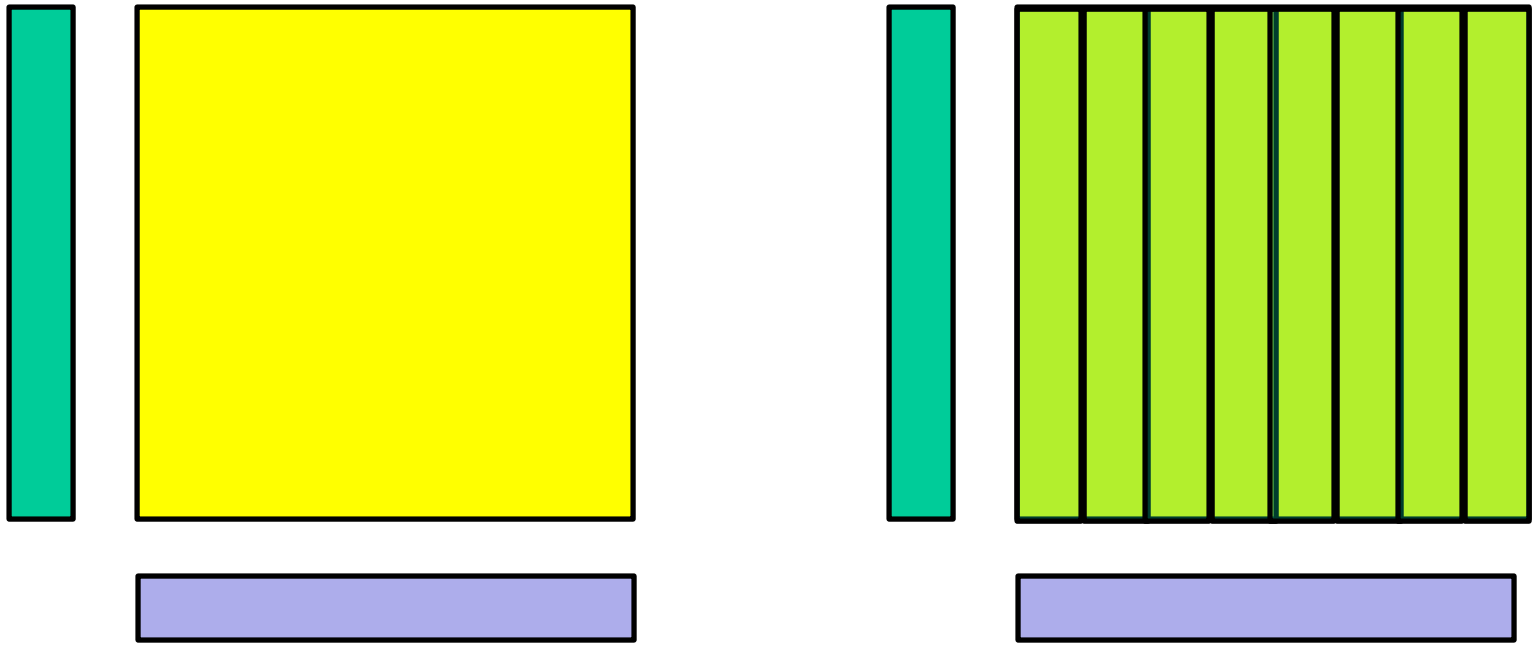
Loop Nesting

```
for n = 1 to N          Do all images
  for k = 1 to K          Do all output feature maps
    for c = 1 to C          Do all input feature maps
      for w = 1 to W          Do all horizontal pixels in output fmap
        for h = 1 to H          Do all vertical pixels in output fmap
          for r = 1 to R          Do all horizontal pixels in kernel
            for s = 1 to S          Do all vertical pixels in kernel
              out[n][k][w][h] +=
                in[n][c][w+r-1][h+s-1] *
                filter[k][c][r][s];
```

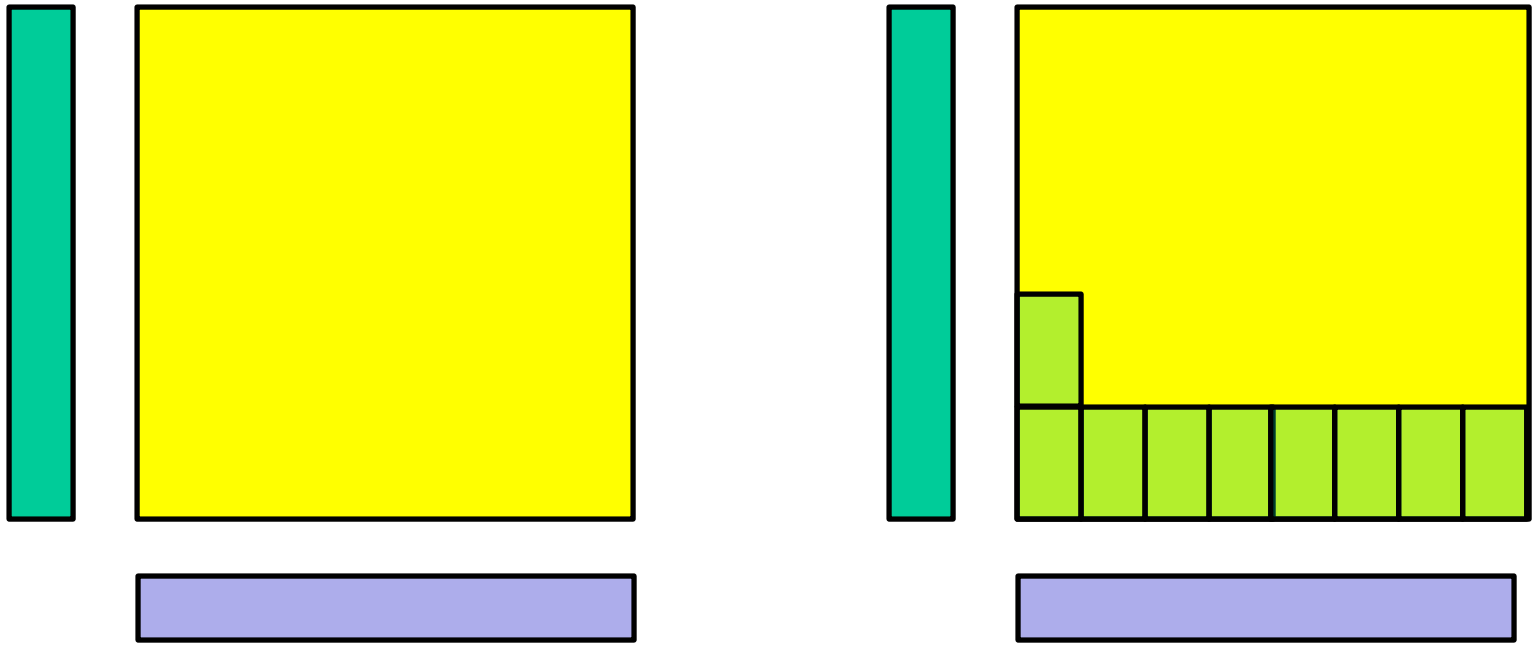
Figure 3: 7-dimensional CNN loop nest.

Source: "SCNN: An Accelerator for Compressed-Sparse Convolutional Neural Networks," A. Parashar et al., ISCA 2017

Data Access Pattern – Classifier Layer

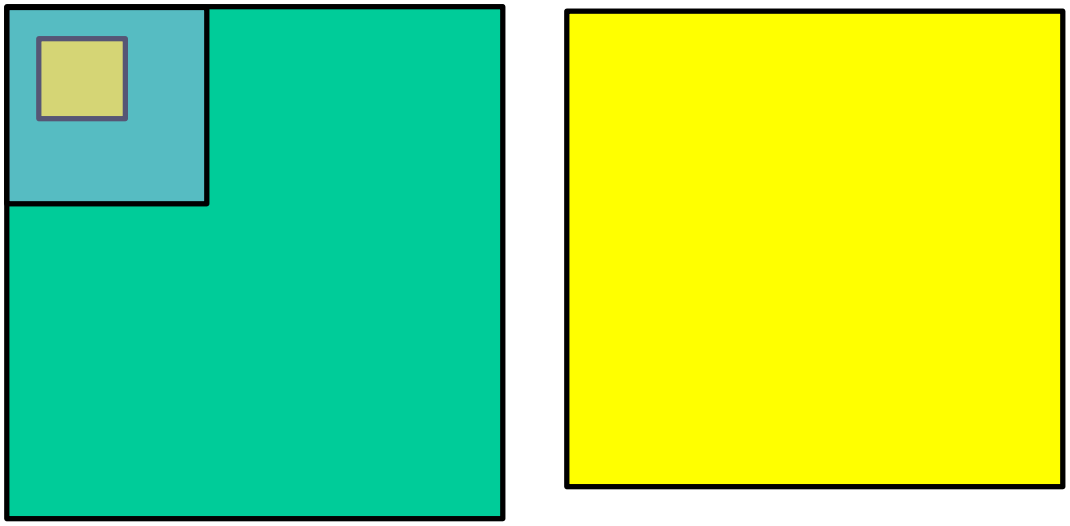


Tiling to Reduce Data Fetches



Tiling – Convolution Layer

2x2 kernel



DianNao Intro

- Deep learning: many layers, large amounts of data moving between layers, many synaptic weights
- Most prior work did little to address the high cost of bringing data to/from memory – DianNao uses tiling, buffering, prefetching to reduce these costs – DaDianNao does even better
- Focus only on inference for now
- Programmable, so it can adapt to algorithm tweaks

Spatially Unfolded Design

- Implement the full neural network with dedicated latches, multipliers, adders, and sigmoid units
- Reasonable for small networks, e.g., 90-10-10 network can be implemented with 974x lower energy than a general-purpose core

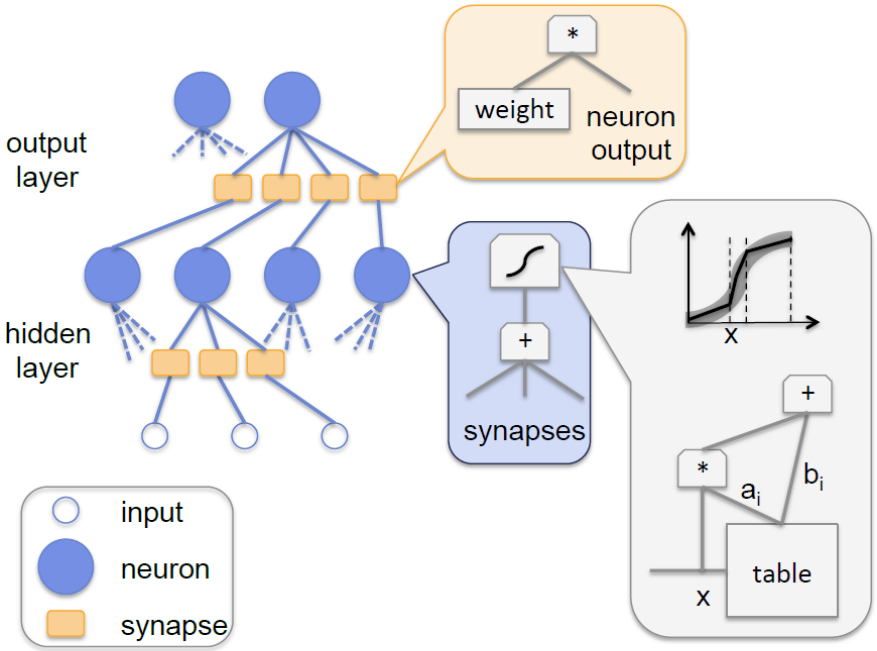


Figure 9. *Full hardware implementation of neural networks.*

Scalability

Spatially unfolded approach does not scale well

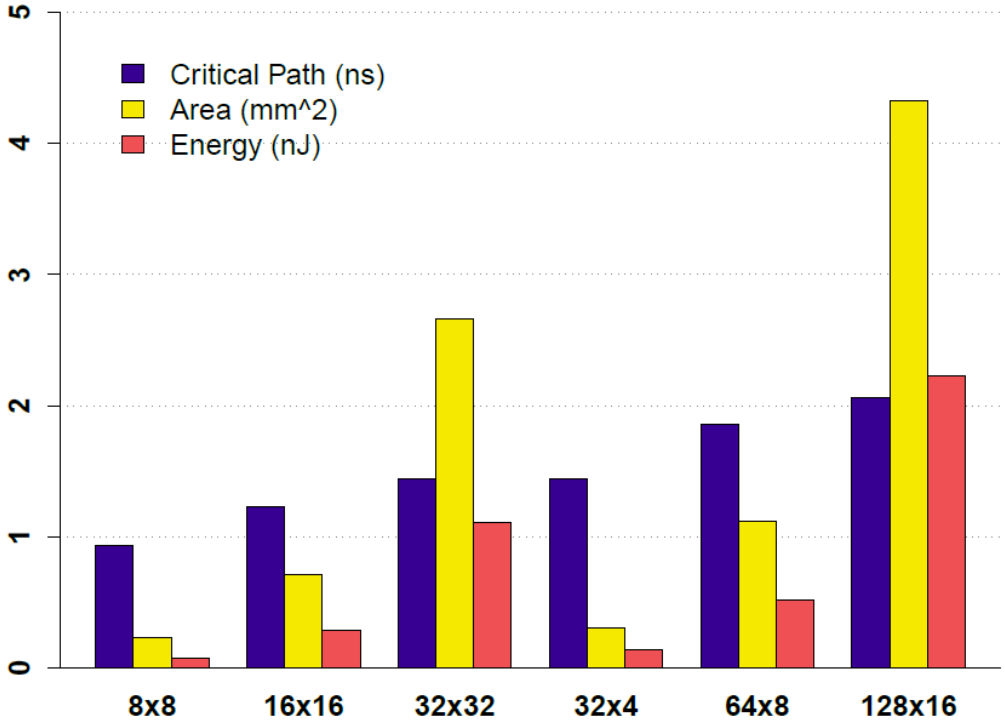
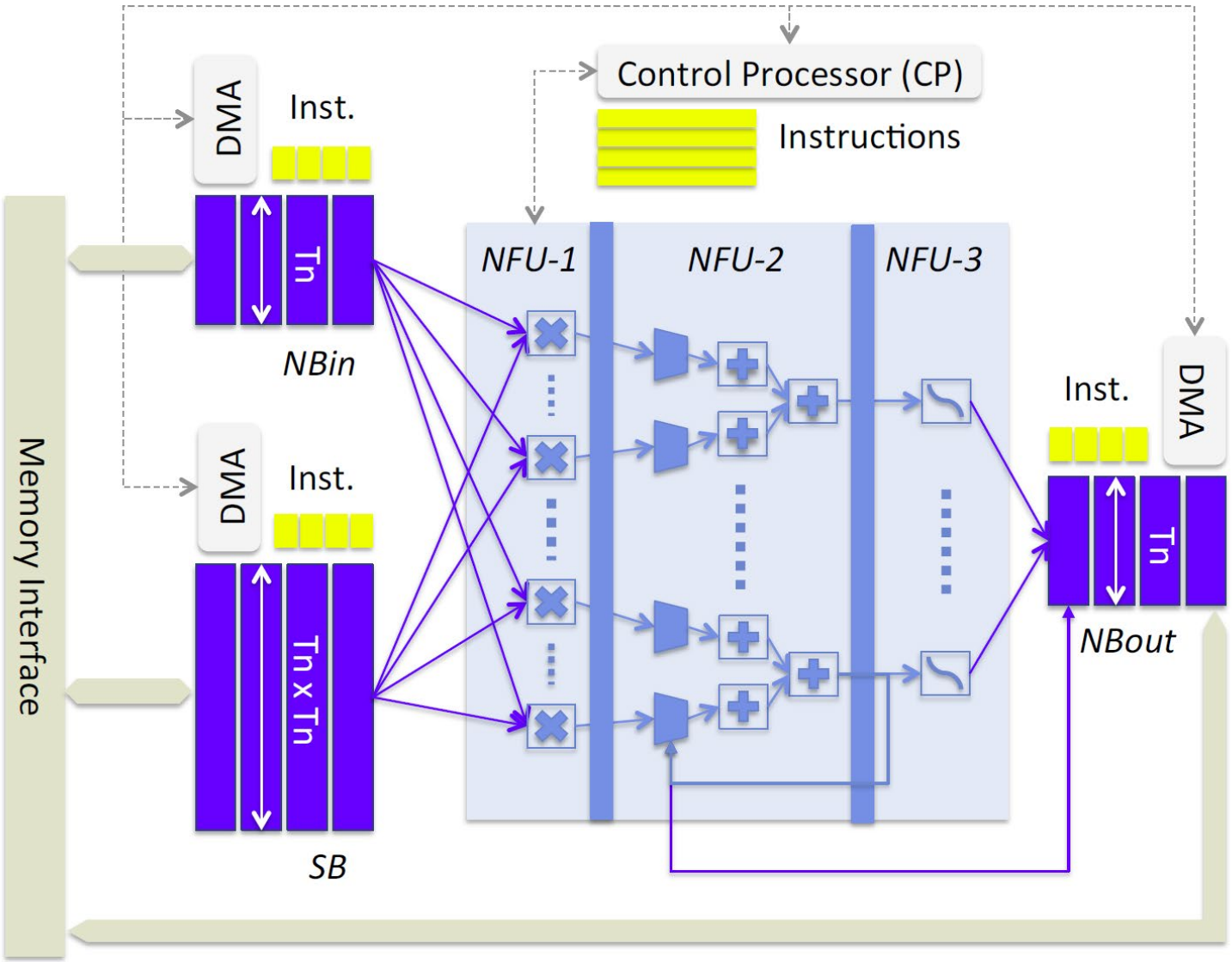


Figure 10. *Energy, critical path and area of full-hardware layers.*

DianNao Approach

- Place data in memory and prefetch “tiles” into buffers
- Each data type has a different requirement, so implement multiple buffers
- The ALUs must be time-multiplexed across several neurons

DianNao


Control Processor (CP)
Instructions
DMA
Inst.
Tn
NBin
DMA
Inst.
Tn x Tn
SB
NFU-1
NFU-2
NFU-3
Inst.
DMA
Tn
NBout
Memory Interface

Design Summary

- Different bit widths for each buffer (Tn = 16, 64 entries)
- Separate “scratchpads” avoid tags/unpredictability/conflicts, and improve latency, parallelism
- 256 parallel multiplies (16 neurons, each with 16 inputs)
- 16 parallel accumulates (each is a tree with 15 adders)
- Partial sums if necessary
- Sigmoid (or any function) is implemented with piecewise linear interpolation – 16 hard-wired segments and coefficients can be programmed into a small table
- Control processor has a number of instructions that specify how data is loaded/accessed in buffers
- Memory bandwidth/energy bottleneck and Amdahl’s Law

Reasoning about Performance, Power

Fixed Point Arithmetic

- DianNao uses 16b fixed-point arithmetic; much more efficient than 32b floating-point arithmetic, and little impact on accuracy

Type	Area (μm^2)	Power (μW)
16-bit truncated fixed-point multiplier	1309.32	576.90
32-bit floating-point multiplier	7997.76	4229.60

Table 2. *Characteristics of multipliers.*

Fixed Point Error Rates

Type	Error Rate
32-bit floating-point	0.0311
16-bit fixed-point	0.0337

Table 1. *32-bit floating-point vs. 16-bit fixed-point accuracy for MNIST (metric: error rate).*

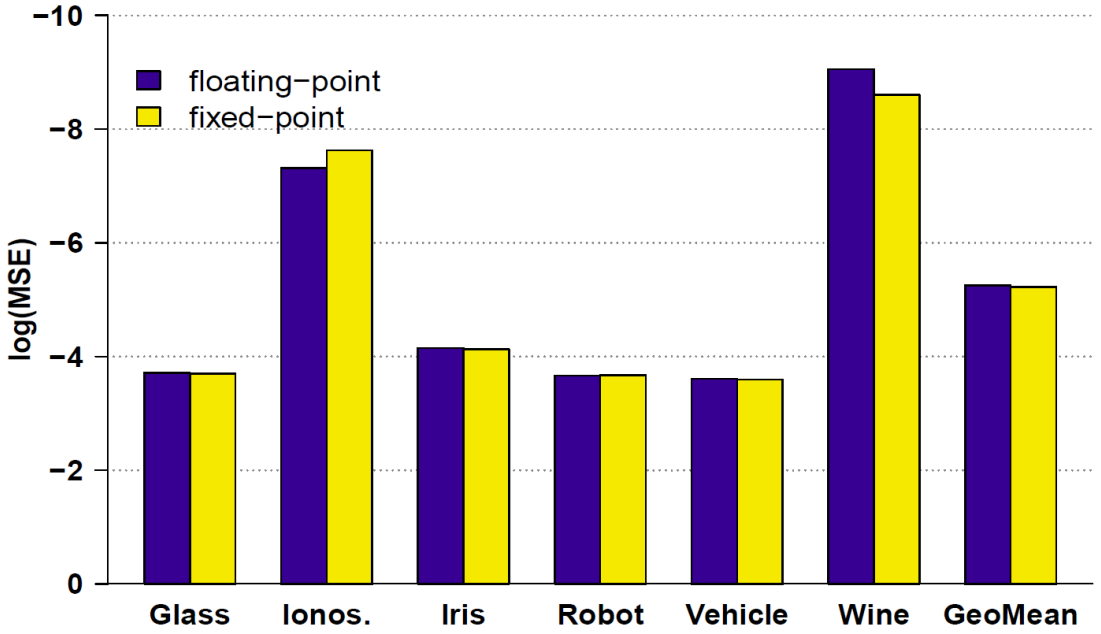
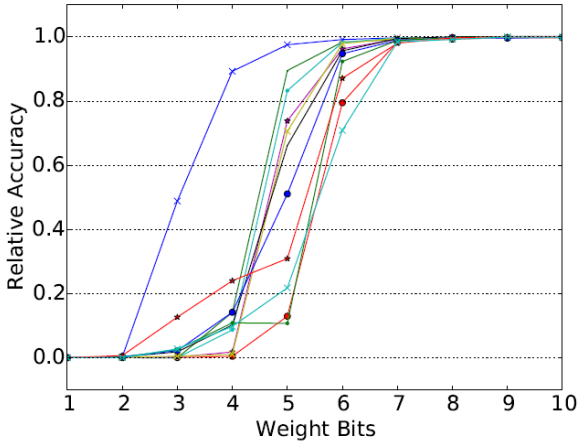


Figure 12. *32-bit floating-point vs. 16-bit fixed-point accuracy for UCI data sets (metric: log(*Mean Squared Error*)).*

Precision Analysis for DNNs

Network	Data (Per Layer)	Weights (Uniform)
LeNet[12]	2,4,3,3	7
Convnet[13]	8,7,7,5,5	9
AlexNet[14]	10,8,8,8,8,8,6,4	10
NiN[15]	10,10,9,12,12,11,11,11,10,10,10,9	10
GoogLeNet[4]	14,10,12,12,12,12,11,11,11,10,9	9

TABLE I: Minimum precision, in bits, for data and weights for a set of neural networks.



(m) GoogLeNet: Weights

Min bits required per layer for accuracy within 1%
Source: Proteus, Judd et al., WAPCO'16

Implementation Details/Results

- Cycle time of 1.02ns, area of 3mm2, 485mW power
- The NFU is composed of 8 pipeline stages
- Peak activity is nearly 500 GOP/s
- 44KB of RAM capacity
- Buffers are about 60% of area/power, while NFU is ~30%
- Energy is 21x better than a SIMD baseline; this is limited because of the high cost of memory accesses
- Big performance boosts as well: higher computational density, tiling, prefetching

Benchmarks

Layer	N_x	N_y	K_x	K_y	N_i	N_o	Description
CONV1	500	375	9	9	32	48	Street scene parsing
POOL1	492	367	2	2	12	-	(CNN) [13], (e.g.,
CLASS1	-	-	-	-	960	20	identifying "building", "vehicle", etc)
CONV2*	200	200	18	18	8	8	Detection of faces in YouTube videos (DNN) [26], largest NN to date (Google)
CONV3	32	32	4	4	108	200	Traffic sign
POOL3	32	32	4	4	100	-	identification for car
CLASS3	-	-	-	-	200	100	navigation (CNN) [36]
CONV4	32	32	7	7	16	512	Google Street View house numbers (CNN) [35]
CONV5*	256	256	11	11	256	384	Multi-Object
POOL5	256	256	2	2	256	-	recognition in natural images (DNN) [16], winner 2012 ImageNet competition

Table 5. *Benchmark layers (CONV=convolutional, POOL=pooling, CLASS=classifier; CONVx* indicates private kernels).*

DianNao Conclusions

- Tiling to reduce memory traffic
- Efficient NFU and buffers to reduce energy/op and prefetch
- Even with these innovations, memory is the bottleneck, especially in a small accelerator with few pins
- For example, each classifier layer step needs 256 new synaptic weights (512 bytes), while 4 memory channels can only bring in 64 bytes per cycle
- Energy consumed by the DianNao pipeline per cycle = 500pJ
- Energy per cycle for fetching 64 bytes from memory = 35nJ

  (70 pJ/b for DDR3 at 100% utilization, Malladi et al., ISCA'12)

It's all about the memory bandwidth/energy !!!

The GPU Option

- Modest (but adequate) memory capacities (a few giga-bytes)
- High memory bandwidth, e.g., 208 GB/s (NVIDIA K20M)
- But, high compute-to-cache ratio on the chip
- Therefore, average GPU power of ~75 W, plus expensive memory accesses ➔ GPU card TDP of ~225W
- A GPU out-performs DianNao by ~2X

References

- “DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine Learning”, T. Chen et al., Proceedings of ASPLOS, 2014