Lecture: SNN vs. MLP

- Topics: comparing SNNs and MLPs in terms of accuracy and hardware efficiency, SpinalFlow

In this discussion, we will attempt an apples-to-apples comparison of an ANN and an SNN to see if one is more energy-efficient or more accurate than the other. The list of questions on the next slide is excellent – it is exactly the questions we're trying to answer in this course. But this paper falls well short of doing the perfect apples-to-apples comparison and it only evaluates simple networks. But studying this paper gives us a blueprint for what we should do and should not do in our own evaluation of this question. We'll end with a discussion of more recent work that provides a better comparison (in my biased view).

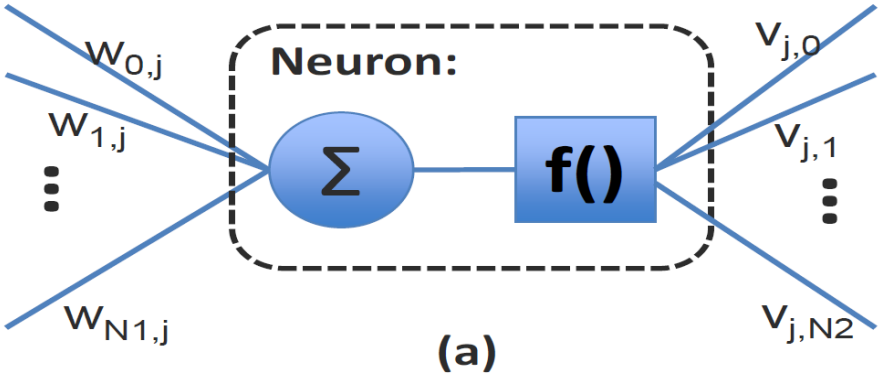
RISC vs. CISC, ML vs. Neuroscience

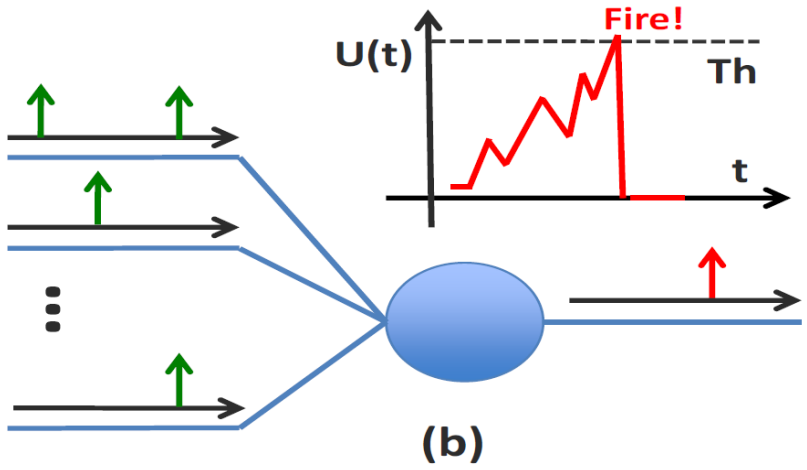
Questions answered in the MICRO'15 paper by Du et al.:

1. Can the neuroscience-inspired approach (SNN + STDP) achieve the same accuracy as the machine learning inspired approach (MLP + BP)?

2. Is the cost of hardware SNN lower than that of hardware MLP?

3. When should a designer use hardware SNN or MLP?





Workloads

- Most of the paper focuses on small networks for MNIST
- One small section on two other even smaller networks:
  MPEG-7 object recognition
  Spoken Arabic Digits for speech recognition

SNN Set-Up

- One layer with 300 neurons
- Each neuron receives inputs from all 784 pixels
- An image is shown; this results in spike trains for the next 500 cycles; 0-10 spikes in each train corresponding to pixel values 0-255; spike intervals are drawn from a Gaussian distribution
- Weights end up being 8b or 12b
- The winning neuron is the one with highest potential; could also use first-to-spike or most spikes
- Trained with STDP

They implement a simple SNN with a single layer of neurons. They experiment with different numbers of neurons and observe that the knee of the accuracy curve is at 300 neurons (see next slide). These neurons receive inputs from all 784 input pixels. With STDP training, some of the neurons end up being the winners for each of the 10 digits. They get decent results with the weights being either 8b or 12b. The input image is shown for 500 cycles. In those 500 cycles, each input sees a spike train corresponding to the input value. They use rate coding, but convert the 8-bit input value into just 0-10 spikes, i.e., the input is down-sampled into a less-than-4-bit input, i.e., their SNN gets lower-resolution inputs than the ANN. But this seems to be the norm in SNN literature.

MLP Set-Up

The ANN has 100 neurons in the first layer that receive inputs from all 784 pixels, followed by 10 neurons in the 2nd layer. They use 8b precision for all computations. See curve below for ANN/SNN accuracies.

- 8b fixed-point weights, inputs, operators
- 784 input neurons, 100 neuron hidden layer, 10 neurons in output layer
- Larger networks had marginally higher accuracies
- Sigmoid activation function
- Training with back-propagation

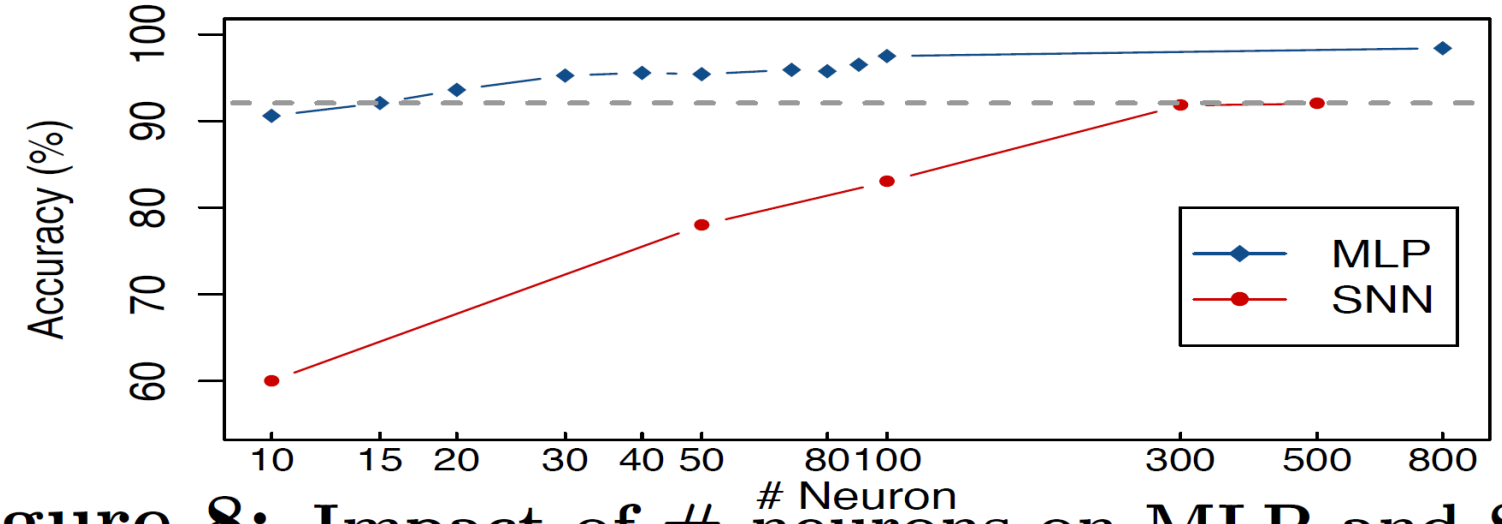


Figure 8: Impact of # neurons on MLP and SNN.

Accuracies on MNIST

The SNNs are about 6% worse in accuracy. SNNwot is basically simplifying the SNN by taking away the timing info. So instead of sending 10 spikes slowly over a 500-cycle window, they just send the number 10.

- CNNs/DNNs have achieved accuracies of 99.21% and 99.77% (60M+ synapses)
- MLP with 800 hidden neurons: 98.4% (related work)
- In this work, they focus on an MLP with 100 hidden neurons: 97.65%

- SNN+STDP: prior works have achieved 93.5% (300 neurons) and 95% (6400 neurons)
- In this work, they achieve 91.82% with 300 neurons
- They also introduce a simplified SNNwot model with accuracy 90.85%

SNNwot: SNN-without-time: send the number of spikes in a 500-cycle window, ignore leak, much better in terms of speed (500x better), 1% worse in terms of accuracy

Bridging the Accuracy Gap

- Starting with SNN+STDP, 91.82%
- … they instead use back-propagation to train (using SNN computations, but computing the error function and applying gradient descent) 95.40%

Most of the 6% gap can be bridged by using back-prop instead of STDP. The rest can be attributed to the ANN's better activation function and its use of higher precision math.

- MLP+BP with a step activation function 97.2%
- MLP+BP with a sigmoid activation function 97.65%

  (going from 32b fl-pt to 8b fx-pt takes accuracy to 96.65%)

Note that a TrueNorth paper achieved 99.42% accuracy on MNIST by using BP and an ensemble of networks

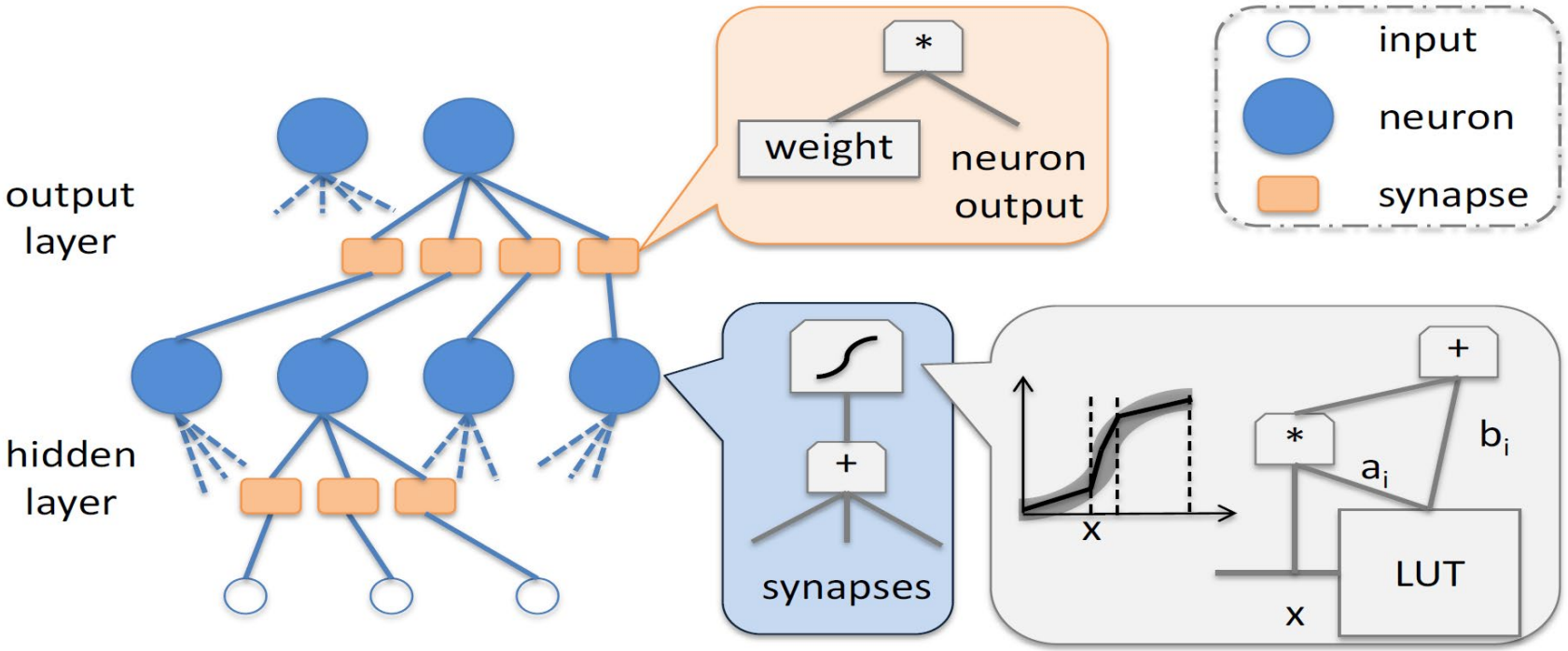
Potential Extensions

- Are we handicapped by only using 10 spikes to encode the input pixel?
- By introducing stochasticity, can augment SNN so it has a sigmoid activation function
- Even better, can augment SNN to use ReLU
- Can also make the network deeper/smaller

MLP Implementation

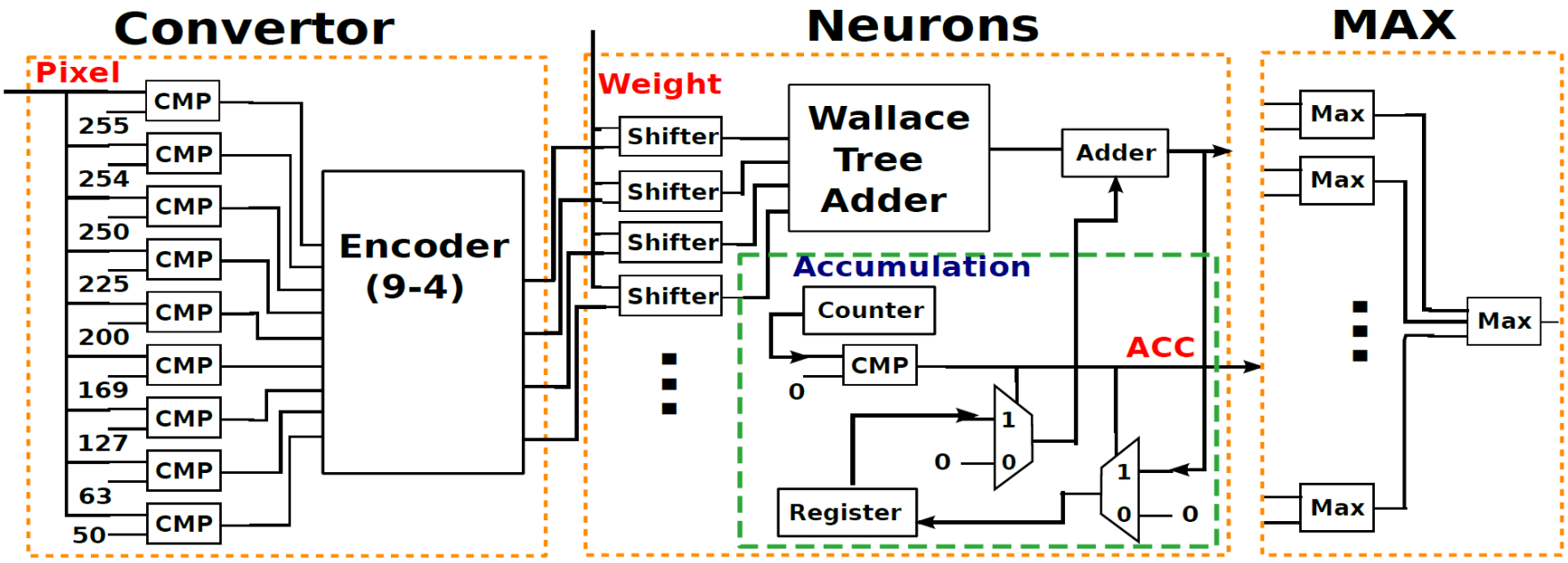
After the accuracy study, now let's focus on hardware complexity. We'll first consider a spatially expanded design, like the one shown below for an ANN. Every neuron has its own dedicated hardware. This will only work for small networks. You've seen this design before – there's a multiplier and register for every synapse, a multi-input adder, and a look-up table for activation.

Expanded design, i.e., there is a dedicated ALU/register for every neuron and synapse (since the MLP is small enough)



SNN Implementation

Expanded design. Input pixels are converted to 4-bit inputs. Using 4-bit multipliers in SNNwot. Using counters/accumulation in SNNwt. Using MAX circuit to decide output (max of potential or spikes).



The SNN design first has an encoder to convert the 8-bit pixel value into a number between 0-10 (spikes). There's a 4bx12b multiplier at the neuron, followed by an adder (in SNNwot). In SNNwt, there is no multiplier, but a counter and accumulator. Finally, there's a max circuit to figure out the winning neuron (whoever has the highest potential or whoever produced the most spikes). Recall that we're first considering expanded designs where every neuron has its own dedicated hardware.

Expanded Design Comparison

Because the SNNwot circuit has a simpler 4bx12b multiplier than the ANN's 8bx8b multiplier, it consumes about 1/3rd the area. This is because the multiplier is about 9x cheaper, but the SNN has 3x more neurons. The SNN also needs 3x more area for SRAM because it has 3x more neurons. More details on next slide.

Table 4: Spatially expanded SNN vs. MLP.

Network	Operator	Area /operator (μm^2)	# operators	Total cost /operator (mm^2)	Total cost /wo SRAM (mm^2)	SRAM (mm^2)	Total cost (mm^2)
SNNwot (28x28-300)	adder tree max	89006 6081	300 16	26.70 0.10	26.79	19.27	46.06
SNNwt (28x28-300)	adder tree rand	60820 1749	300 784	18.25 1.37	19.62	19.27	38.89
MLP (28x28-100-10)	adder tree adder tree multiplier	45436 5657 862	100 10 79510	4.54 0.06 68.54	73.14	6.49	79.63
MLP (28x28-15-10)	adder tree adder tree multiplier	45436 1131 862	15 10 11935	0.68 0.01 10.29	10.98	1.35	12.33

Expanded Design Results

- The 4x12 multiplier (SNNwot) is 8x cheaper than the 8x8 multiplier (MLP)
- The SNN needs 3x more ALUs and storage than MLP
- The SNN takes 3x less area for ALUs, but 3x more area for storage
- All put together, SNN takes ~2x less area than MLP
- SNNwt takes less area than SNNwot because it has simpler adders
- A 15-10-neuron MLP has the same accuracy as the SNN – it also consumes 3x less area than SNN

- For a small-scale design, they also evaluate delay and power

Table 5: Hardware Characteristics of SNN (4x4-20) and MLP (4x4-10-10).

Type	Area (mm^2)	Delay (ns)	Power (W)	Energy (nJ)
SNN	0.08	1.18	0.52	0.63
MLP	0.21	1.96	0.64	1.28

Spatially Folded Designs

- For a spatially folded MLP, each neuron is assumed to have only n_i inputs; must compute multiple partial sums; need an SRAM buffer to store all weights and feed ALUs
- They do not consider folding where multiple neurons share a set of ALUs
- Same approach for SNNs as well

In the spatially folded design, typically a single piece of hardware is shared by multiple neurons. Here, they look at a more limited form of folding where every neuron still has its own hardware, but that hardware has a limited number of inputs. So the many synapses for one neuron are sharing a few multipliers/adders. They assume that a neuron can only process n_i inputs at a time. They therefore need a buffer to store all weights and pass them to the multipliers at the right time.

See discussion of these numbers on next slide.

Spatially Folded Comparison

Type	# Inputs per operator	Area (no SRAM) (mm^2)	Total Area (mm^2)	Delay (ns)	Energy (uJ)	# Cycles (per image)
SNNwot (28x28-300)	$ni = 1$	1.11	3.17	1.24	1.03	791
	$ni = 4$	1.89	5.34	1.48	0.68	203
	$ni = 8$	2.79	8.91	1.76	0.67	105
	$ni = 16$	4.10	16.33	1.84	0.70	56
	expanded[3]	26.79	46.06	3.17	0.03	3
SNNwt (28x28-300)	$ni = 1$	0.48	2.56	1.15	471.58	791*500
	$ni = 4$	0.84	4.36	1.11	315.33	203*500
	$ni = 8$	1.19	7.45	1.18	307.09	105*500
	$ni = 16$	1.74	14.25	1.84	325.69	56*500
	expanded[3]	19.62	38.89	2.61	214.70	500
MLP (28x28-100 -10)	$ni = 1$	0.29	1.05	2.24	0.38	882
	$ni = 4$	0.62	1.91	2.24	0.29	223
	$ni = 8$	1.02	3.26	2.25	0.30	113
	$ni = 16$	1.88	6.36	2.25	0.29	57
	expanded[3]	73.14	79.63	3.79	0.06	4

Spatially Folded Results

- Need more cycles to process all inputs, but the clock is faster
- Small n_i is very area-efficient and has high clock speeds
- High n_i is very energy-efficient because the task finishes sooner
- SNNwt is not competitive in terms of time because we have to sequence through 500 cycles (at least); but it consumes less area because it uses adders instead of multipliers
- Each SNNwt neuron has roughly the same area as an MLP neuron (because of the SRAM, the multiplier complexities are less pronounced); therefore, SNNwt consumes 3x more area because it uses 3x more neurons

- Bottomline: not much hw difference between an SNN neuron and an MLP neuron; differences are because of #neurons needed for high accuracy
- Their conclusion: SNNs may be better when expanded (small networks) or when on-line learning (STDP) is required

Adding Features for On-Line Training

- STDP: more counters, comparators, SRAM updates
- Counters for refractory and inhibitory periods
- Homeostasis: change the threshold so every neuron has a roughly equal firing rate
- Adding these features increases area by 1.3-1.9x, and energy by <1.5x

They also show that STDP features can be easily added to SNNs at modest cost.

Other Benchmarks

- MPEG-7 object recognition
  MLP 15-10 99.7% accuracy
  SNN 90 92% accuracy

- Spoken Arabic Digits speech recognition
  MLP 60-10 91.35% accuracy
  SNN 90 74.7% accuracy

Note that these benchmarks are even smaller than the MNIST benchmark. What they really need to do is examine deeper/larger networks.

Main Conclusions

- MLP has an advantage in accuracy
- Some of the gap can be bridged with BP, sigmoid, better input encoding, etc.
- For time-multiplexed hardware units, the neuron model may not matter much
- SNN has an advantage in on-line learning and for spatially expanded designs

These conclusions should be taken with a large grain of salt since we've already pointed out a few drawbacks of this study (see next slide).

Limitations of this Study

1. Small-scale workloads
2. 4-bit rate coding (and no thorough evaluation of other codes)
3. Modeling other activation functions in SNNs (ReLU, sigmoid)(not MAX)
4. Considering the impact of stochasticity on SNN accuracy/cost
5. Considering the impact of the many TrueNorth modes on acc/cost
6. Finding the right balance of ALUs/storage/SIMD-ness
7. Need a much better analysis/breakdown for tiled large-scale designs
8. What about the SRAM overheads for neuron potential (Conv and FC)?

They should consider larger networks and larger scalable (tiled) architecture. It would be good to get a detailed breakdown of a DaDianNao or TrueNorth tile to see if one is indeed lower cost or higher throughput than the other and why. From this study, for a limited type of spatial folding, all I learn is that SNN/ANN neurons have similar area/energy, but I have little insight for why that's the case. The next few slides discuss our attempt at fixing the above limitations (paper still under review).

Eyeriss vs. Spiking Eyeriss

As a baseline, we adapt Eyeriss to handle spikes. Eyeriss' partial-sum registers are used to store neuron potential. Because we are using the Eyeriss dataflow and have to process spikes for each cycle of the input interval, performance is much lower. Energy is much higher. But energy is lower for smaller input intervals (the resolution) and higher levels of sparsity in the input spike train.

- Slowdown of 16x for a 16-tick input interval
- Rate-coded energy results:

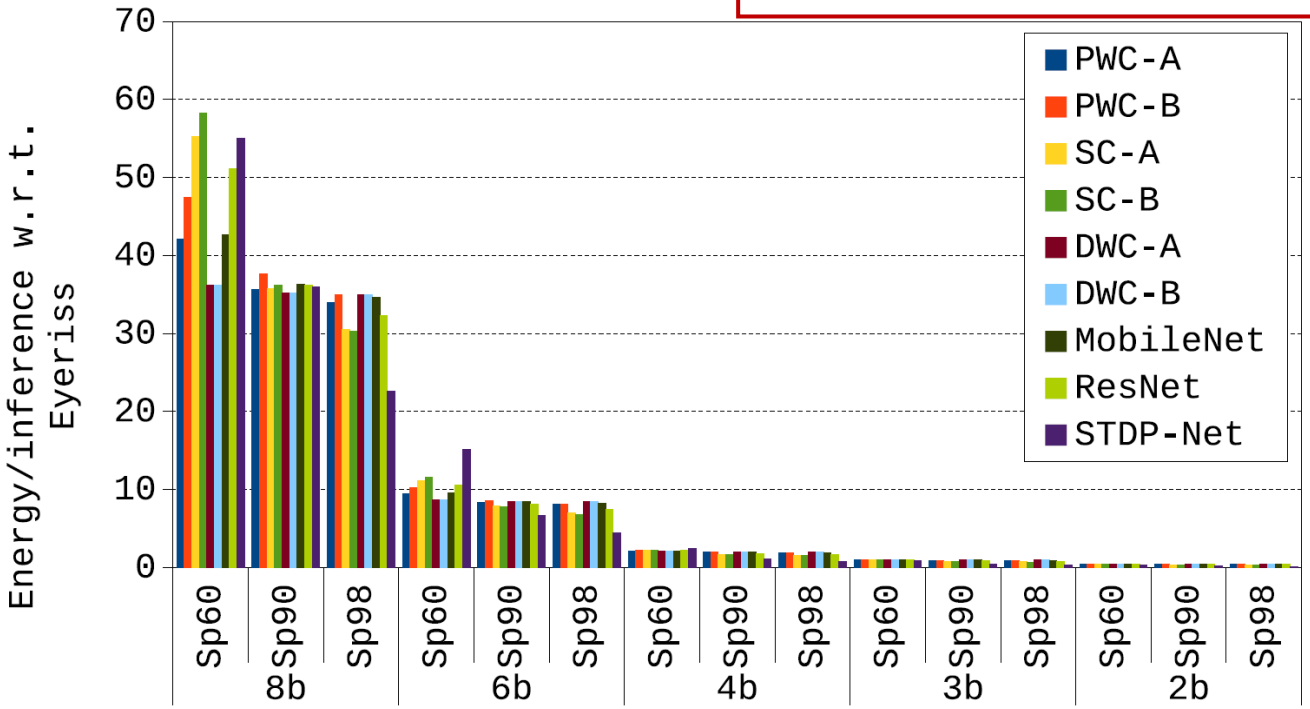


Figure 4: *Energy consumed per inference by r-SNN on Spiking-Eyeriss normalized to Eyeriss. Sp60, Sp90, and Sp98 refers to 60%, 90%, and 98% sparsity respectively.*

Eyeriss vs. Spiking Eyeriss

Temporal codes are significantly more energy-efficient than rate codes. At about 16-cycle input interval and sparsity of 90%, Spiking Eyeriss is competitive with Eyeriss.

- Crossover points: 3bSp90 (rate) and 3bSp60 (temp)
- Temporal coding energy results:

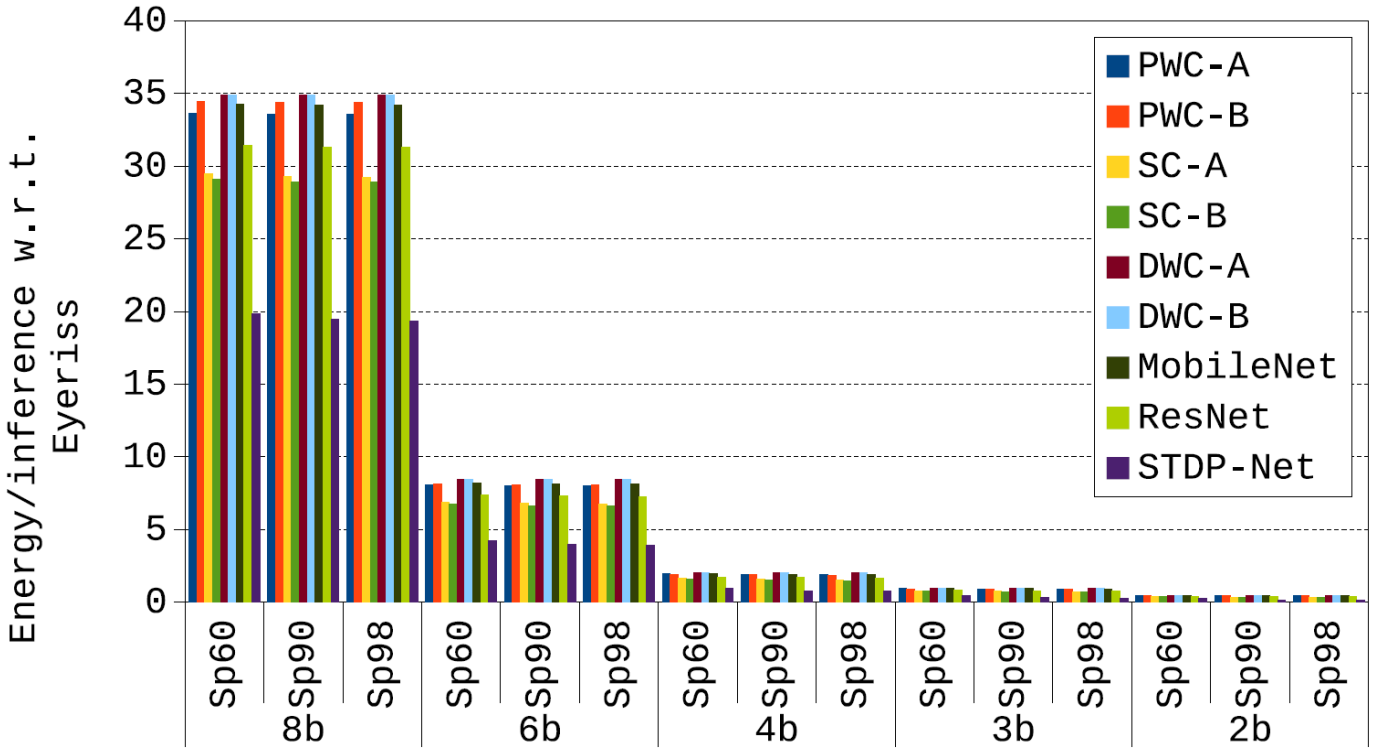
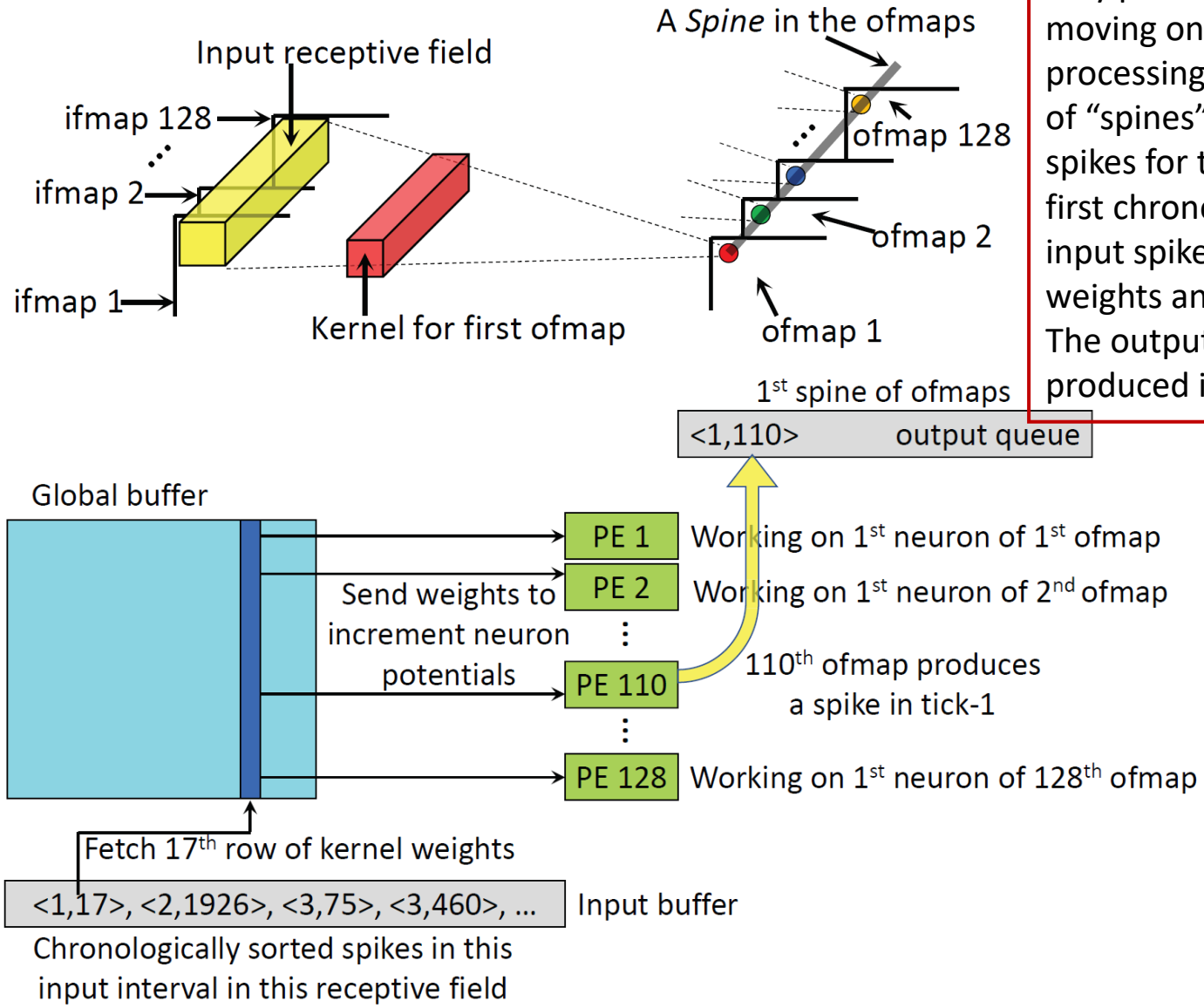


Figure 5: *Energy consumed per inference by t-SNN on Spiking-Eyeriss normalized to Eyeriss.*

SpinalFlow

To do better, we introduce a new dataflow. The key novelty is to reduce global buffer fetches of weights and to avoid storage for neuron potentials by fully processing a neuron before moving on to the next. The processing is performed in the order of "spines", defined here. The input spikes for the input receptive field are first chronologically sorted. For each input spike, we read a spine of kernel weights and send it to an array of PEs. The output spikes for a spine are produced in chronologic order.



SpinalFlow Energy

The better dataflow gives energy that is competitive with Eyeriss and scales well as sparsity increases. For the typical operating range, it is 5x less energy than Eyeriss.

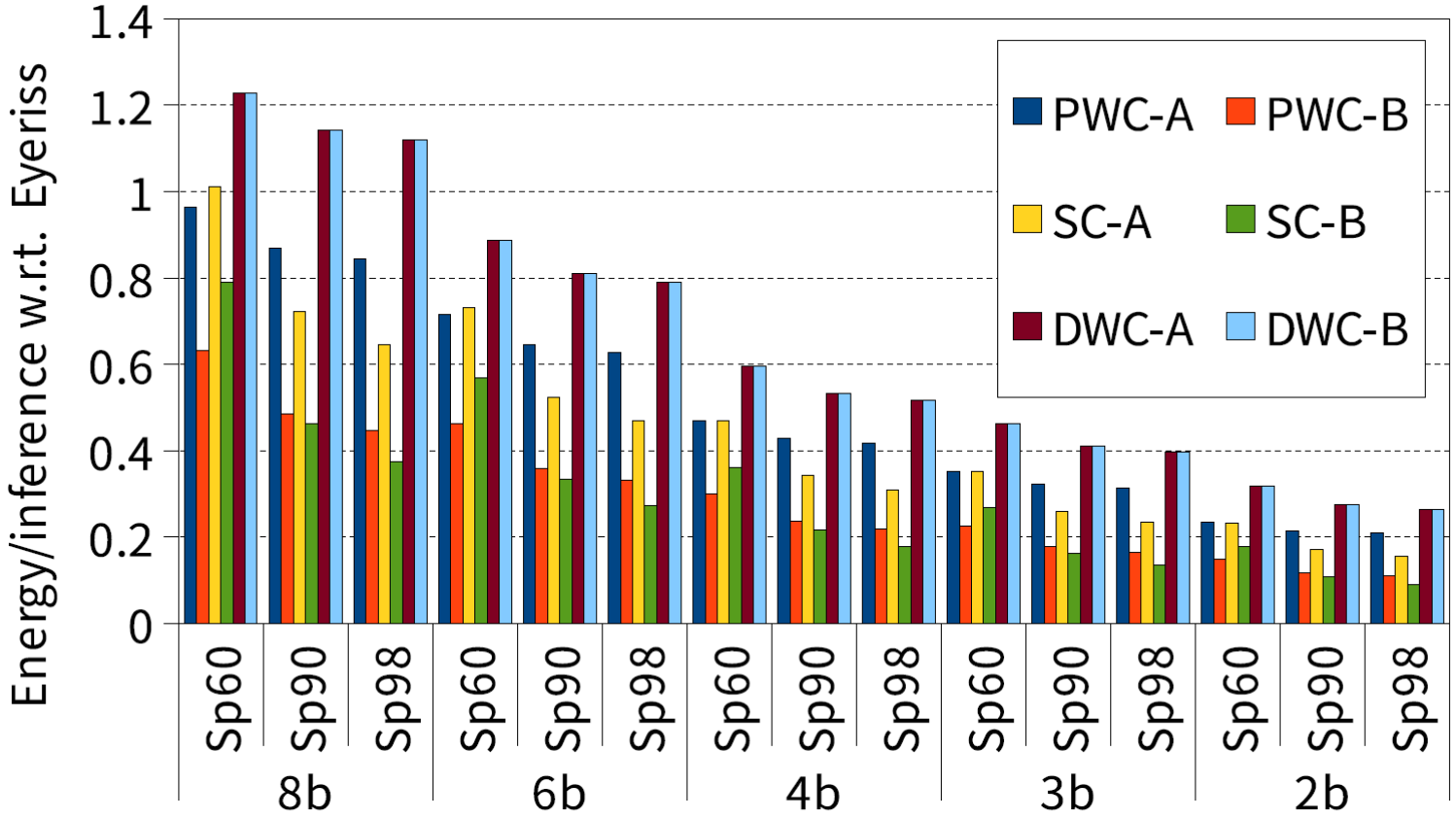


Figure 11: *Energy/inference of SpinalFlow normalized to an 8-bit Eyeriss with 60% sparsity. Sp60, Sp90, and Sp98 refers to 60%, 90% and 98% sparsity for the SNN.*

Accuracy Comparison

This table is a more recent snapshot of SNN vs. ANN accuracy for a range of workloads, precisions, coding, and training approaches.

Workload	ANN Accuracy	SNN Accuracy
MNIST	99.8% [57] 1-bit res: 99.04% [11]	r-SNN(SGD): 99.59% [33] r-SNN(STDP+SGD): 99.28% [32] t-SNN (SGD): 97.55% [40] t-SNN (STDP): 98.4% [29]
CIFAR10	92.38% [49] 1-bit res: 88.6% [11]	r-SNN: 90.53% [59]
AlexNet on ImageNet	55.9% [61, 24] 1-bit res: 44.2% [47] 2-bit res: 49.8% [61] 4-bit res: 53.0% [61]	r-SNN: 51.8% [24]
VGG on ImageNet	70.52% [50]	r-SNN: 69.96% [50]
ResNet on ImageNet	70.69% [50]	r-SNN: 65.47% [50]

Table 4: *Accuracy comparison with supervised training on labeled datasets.*

References

- “Neuromorphic Accelerators: A Comparison Between Neuroscience and Machine Learning Approaches”, Z. Du et al., MICRO 2015

- “SpinalFlow”, under review.