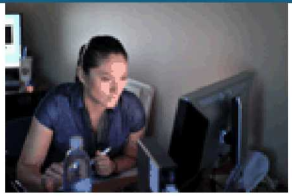
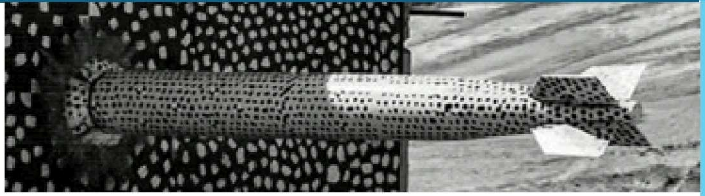
SAND2019-3295C

# Low-Power Deep Learning Inference using the SpiNNaker Neuromorphic Platform





Craig M. Vineyard, Ryan Dellana, James B. Aimone, William M. Severa

PRESENTED BY

Ryan Dellana

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

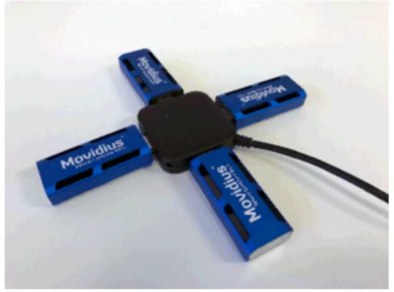
# Some Problem Scenarios Requiring Efficient/Local Inference

| Application | Relevant Constraints |
|---|---|
| Smartphones | power, latency, bandwidth |
| Autonomous Vehicles | latency |
| Supercomputing | thermal |
| Security Systems on Backup Power | power, interference |
| Imaging Satellites | power, bandwidth |
| Micro-Drones | power, latency, bandwidth, interference |
| Autonomous Hypersonics | latency, interference |
| Nuclear Reactor Emergency Maintenance | power, thermal, interference |
| Advanced Neuroprosthetics | power, latency, thermal, privacy |

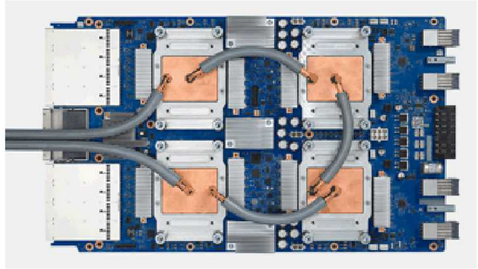
# Platforms Offering Efficient Low-Power Inference

## Continuous

NVIDIA Jetson TX1-2

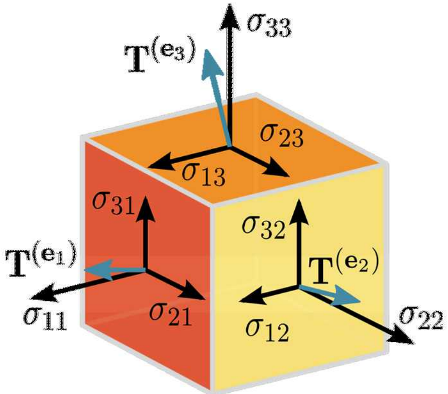



Intel Movidius



Google TPU

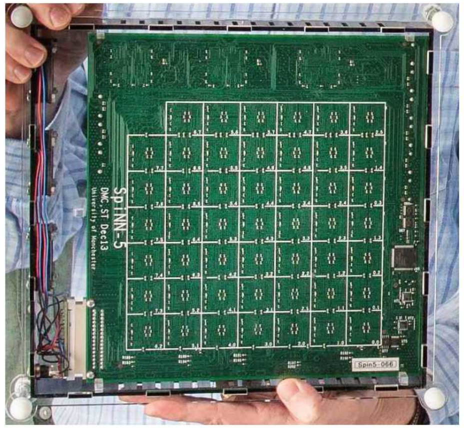
?



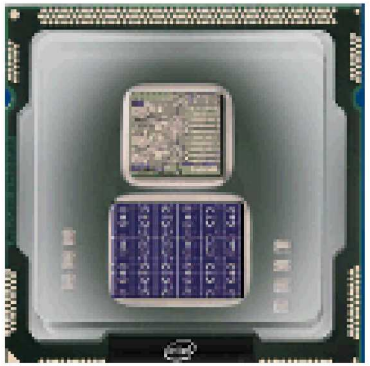

Deep ANNs

Native

## Spiking



Manchester SpiNNaker



Intel Loihi

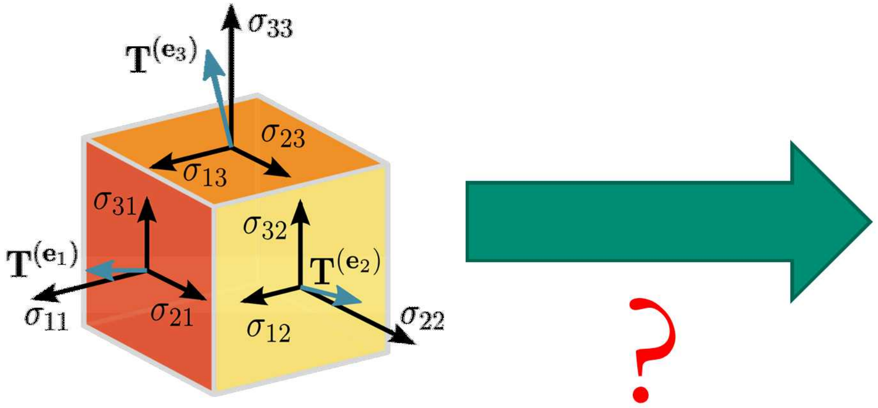

IBM TrueNorth

SNL STPU

# Some Challenges Porting "Traditional" Deep Nets to Spiking Hardware

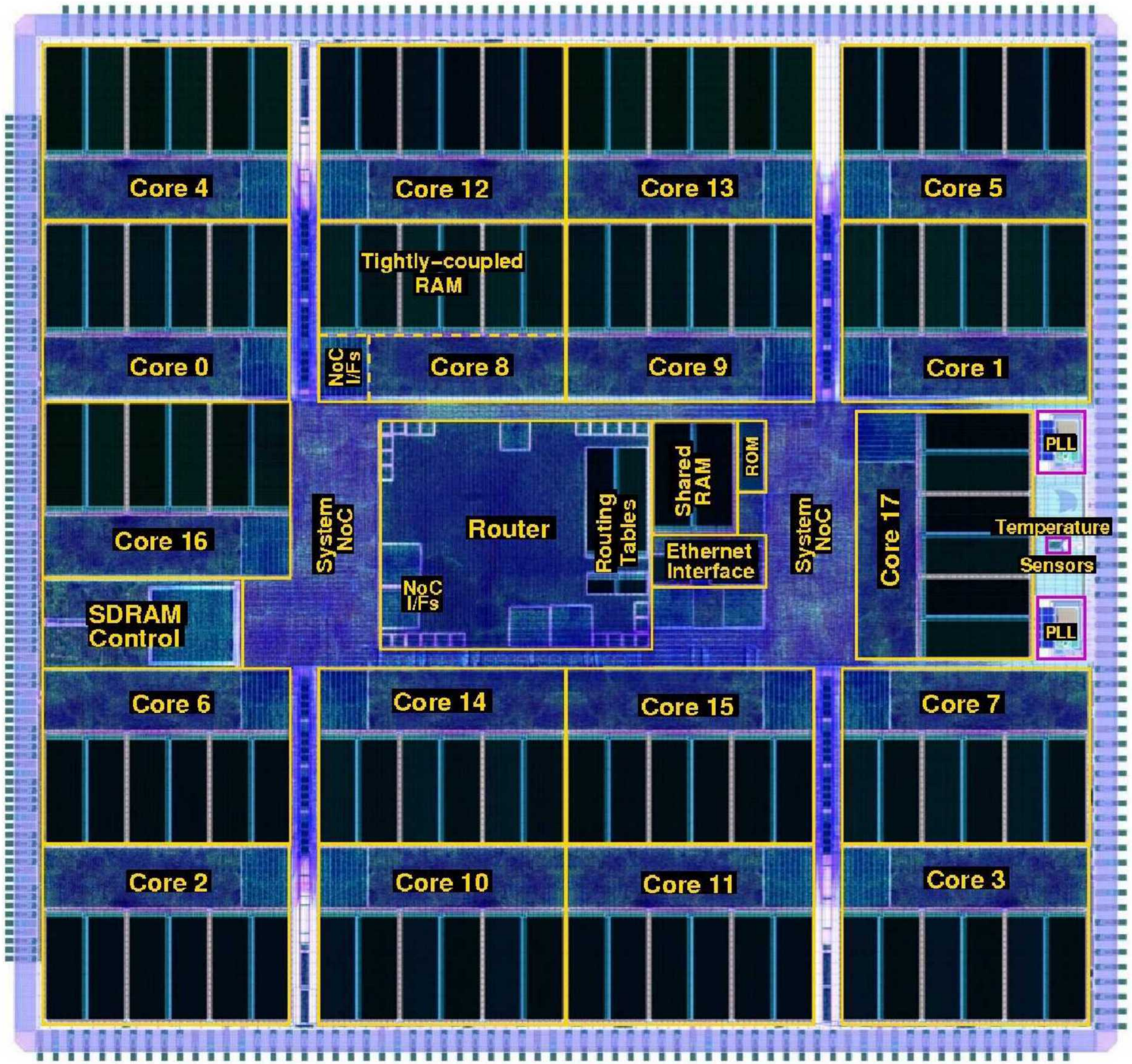
- Spiking Hardware is Typically Designed for Neuro-Simulation.

| Traditional ANNs | Spiking Nets |
|---|---|
| Continuous Activations | Discrete/Binary "Spikes" |
| Global Clock-Driven Synchrony | Local Event-Driven Asynchrony |
| Dense Matrix/Tensor Representations | Sparse Synapse/Neuron Representations |
| Floating-Point Weight Precision | Weights Often Fixed-Point |
| Individual "Neuron" Biases | (Often Shared) Firing Thresholds |
| Batch Normalization Layers | ??? |




# SpiNNaker (Case Study)

- Optimized for biological real-time execution ("time models itself").
- Run spiking networks with biological timing and topological constraints.
- Well suited to robotics applications.
- Event-driven.
- Locally synchronous, globally asynchronous.
- Multicast packets with fixed routing tables.
- "SpiNNaker comes into its own when a problem can be cast into a form that requires many, many tiny asynchronous messages..." [6].



# High-Throughput Binary-MNIST on SpiNNaker

[input(785) -> dense(100) -> dense(100) -> dense(100)] x 190 Tiles

Between-Sample Delay: 2 miliseconds.

With Instant Decay Neuron ^^^

time_scale_factor: 5.0

Cores per tile: 4

(1 per pop/layer including spike-source-array)

Core Usage: 760/760

Chips Usage: 48

Total Neurons: 57,000

Total Synapses: 97,617*190 = 18,547,230

Total runtime: 7:09 (includes setup, routing, and I/O)

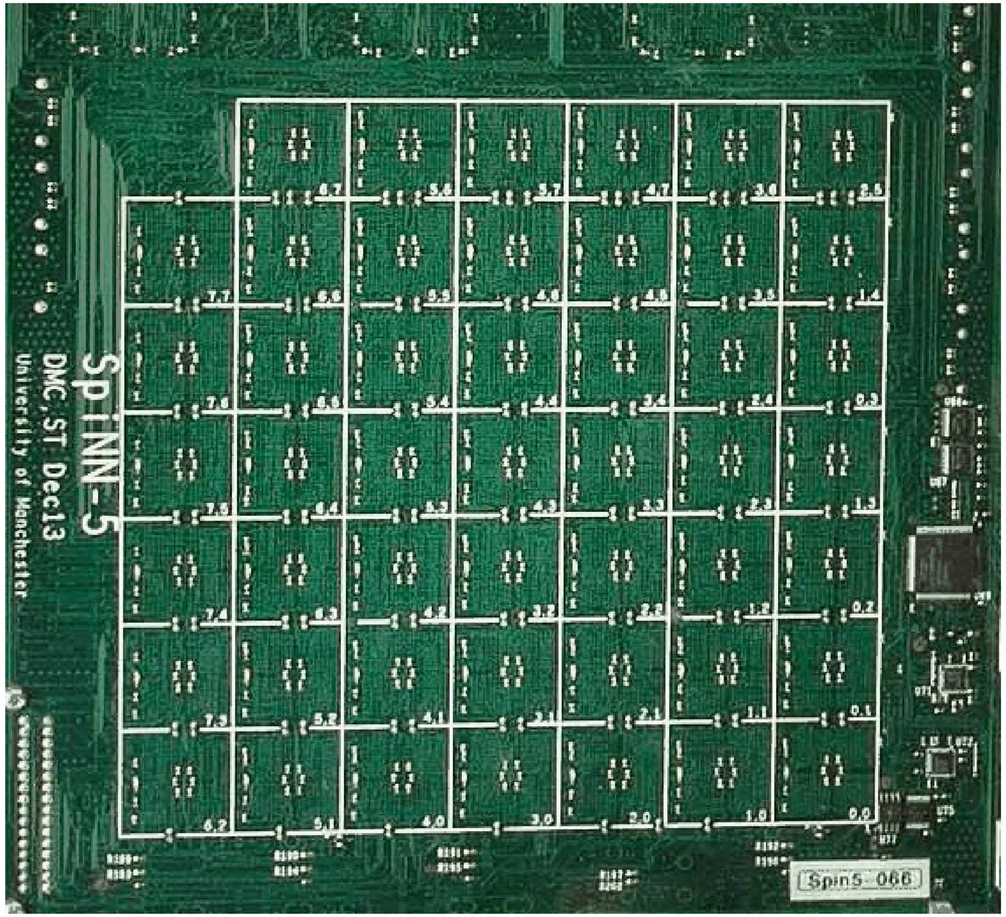
Samples processed: 10,000

Samples per core: 52 -> 53

Inference time: 0.657 seconds.

Throughput: ~15,317 samples/second

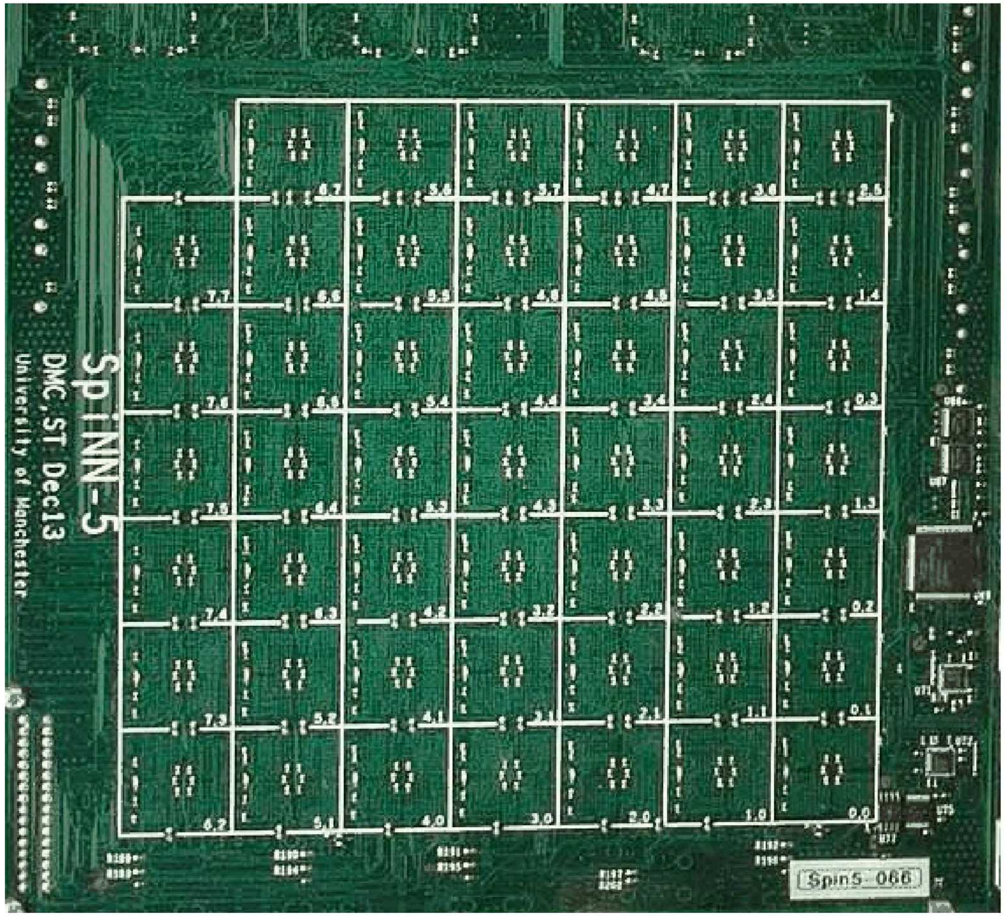
Accuracy: ~94%



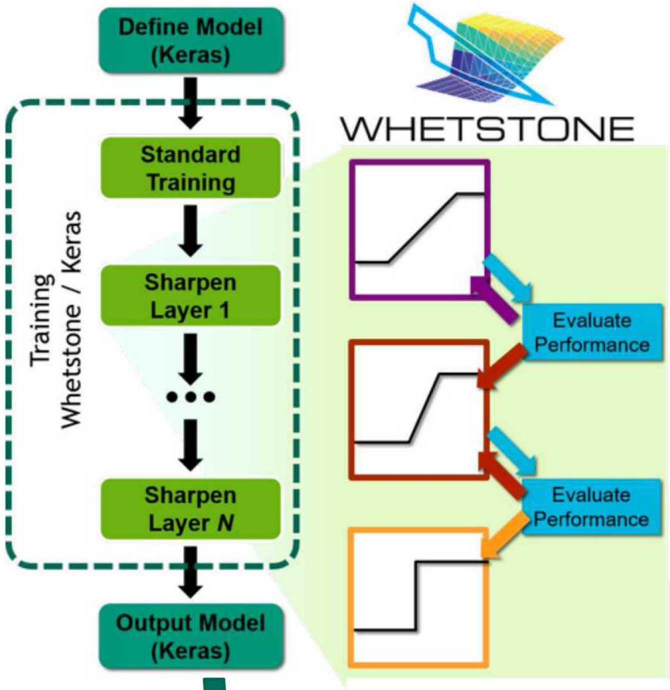
# Convolutional Binary-MNIST on SpiNNaker

[Input(785) -> conv2d((5x5), 32) -> maxpool2d((2x2)) -> conv2d((5x5), 64) -> maxpool2d((2x2)) -> dense(500) -> dense(100)] x 1 Tile

time_scale_factor: 14.0
Temporal Groups: 20
Max-Neurons/Core: 255
Between-Sample Delay: 2ms * tsf = 28ms real-time.
Cores per tile: 371
Cores Used: 371/760 ~49%
Chips Used: 28/48 ~58%
Total Neurons: 47,640
Total Synapses: ~2,596,432
Samples processed: 10,000
Samples per tile: 10,000
Total runtime: 1:22:39 hours:minutes:seconds
(includes setup, routing, and I/O)
Inference time: 51:21 minutes:seconds
Throughput: ~3.25 samples/second
Latency: 1.694 seconds.
Accuracy: 98.10%



# Porting Deep Nets to SpiNNaker using Whetstone (Overview)




ANN with Binary Quantized Activations and Batch Normalization Layers

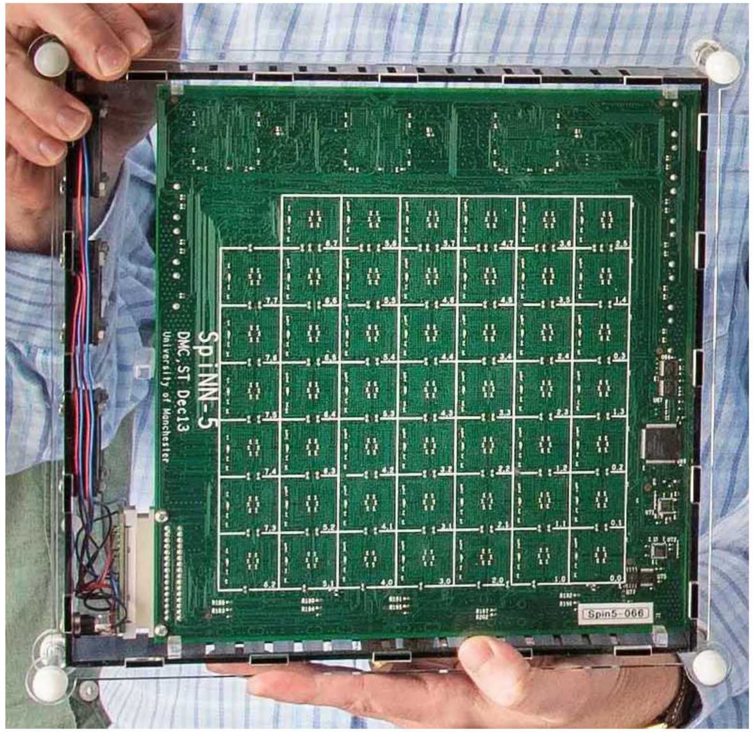
Remove Batch Normalization Layers

Convert from Tensor to Sparse Representation

Replace Biases with Bias Neurons

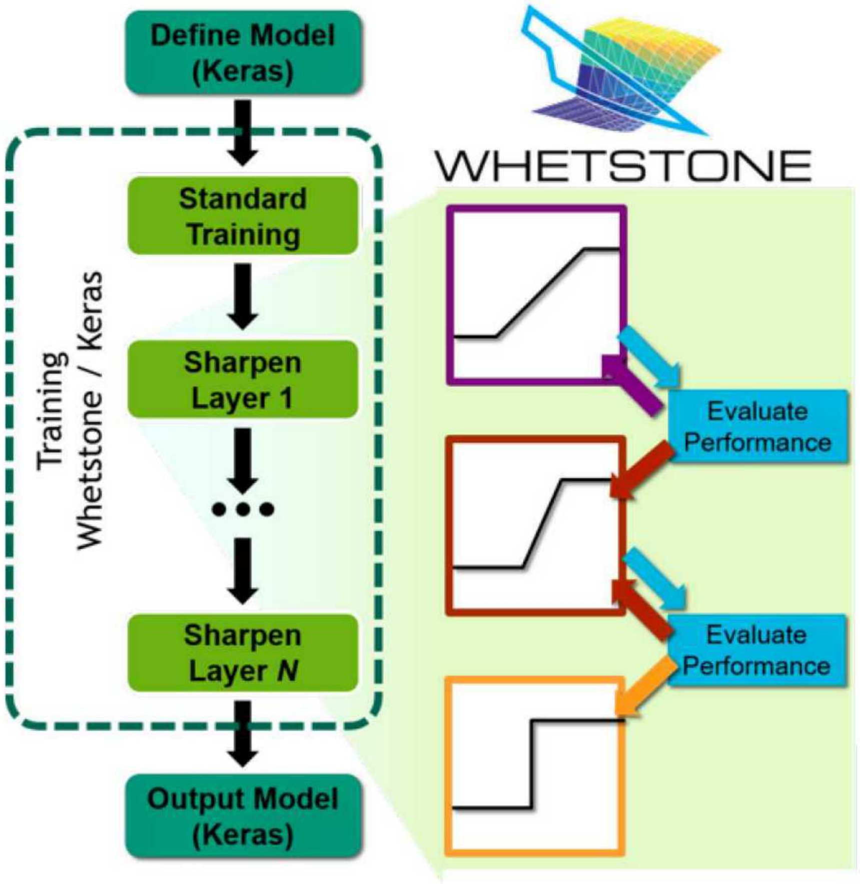
(Optional) Use Propagation Delays to Define Temporal Groups for Time-division Multiplexing

Convert to PyNN Representation and Load to SpiNNaker using SpyNNaker Library



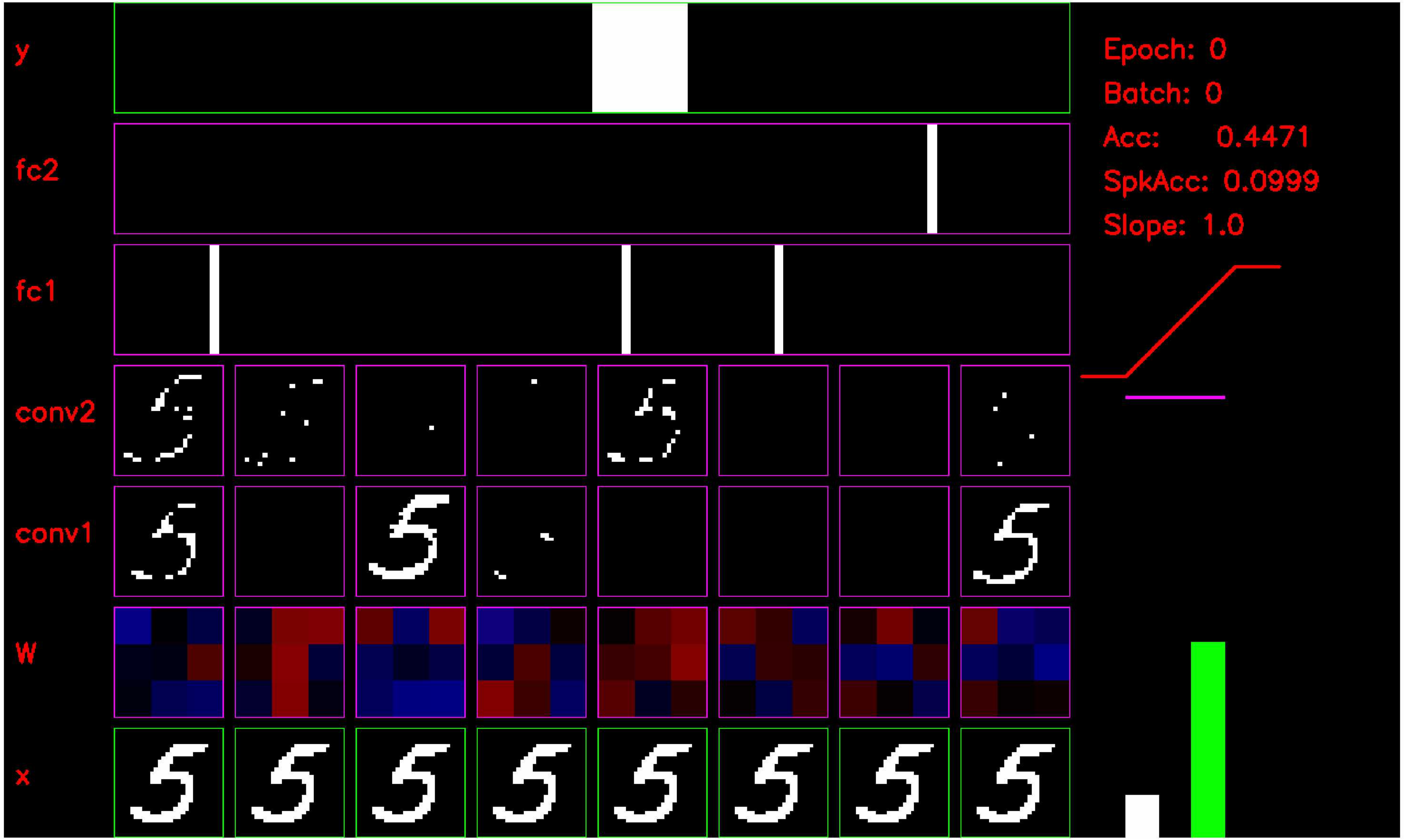
# From Continuous Activations to Binary

| Traditional ANNs | Spiking Nets |
|---|---|
| Continuous Activations | Discrete/Binary "Spikes" |




- During training, activations are gradually sharpened one-layer-at-a-time, starting with the first.
- The rate of sharpening can be determined adaptively based on changes in the loss.
- Result is a binary-activation quantized version of the original network, which can be directly run in many kinds of neuromorphic hardware.
- These networks allow "single-pass inference", where multiple "wave-fronts" can pass through the net simultaneously. This provides advantages in throughput, latency, and possibly energy usage, relative to rate-coded methods which must accumulate spikes for some period of time.

# How Whetstone Works (Animation GIF Format)




Legend for bars:

White: Spiking Accuracy

Green: Non-Spiking Accuracy

# Removing Batch Normalization Layers

| Traditional ANNs | Spiking Nets |
| --- | --- |
| Batch Normalization Layers | ??? |

$$BN(x_i) = \gamma\left(\frac{x_i - \mu_B}{\sigma_B + \varepsilon}\right) + \beta \tag{1}$$

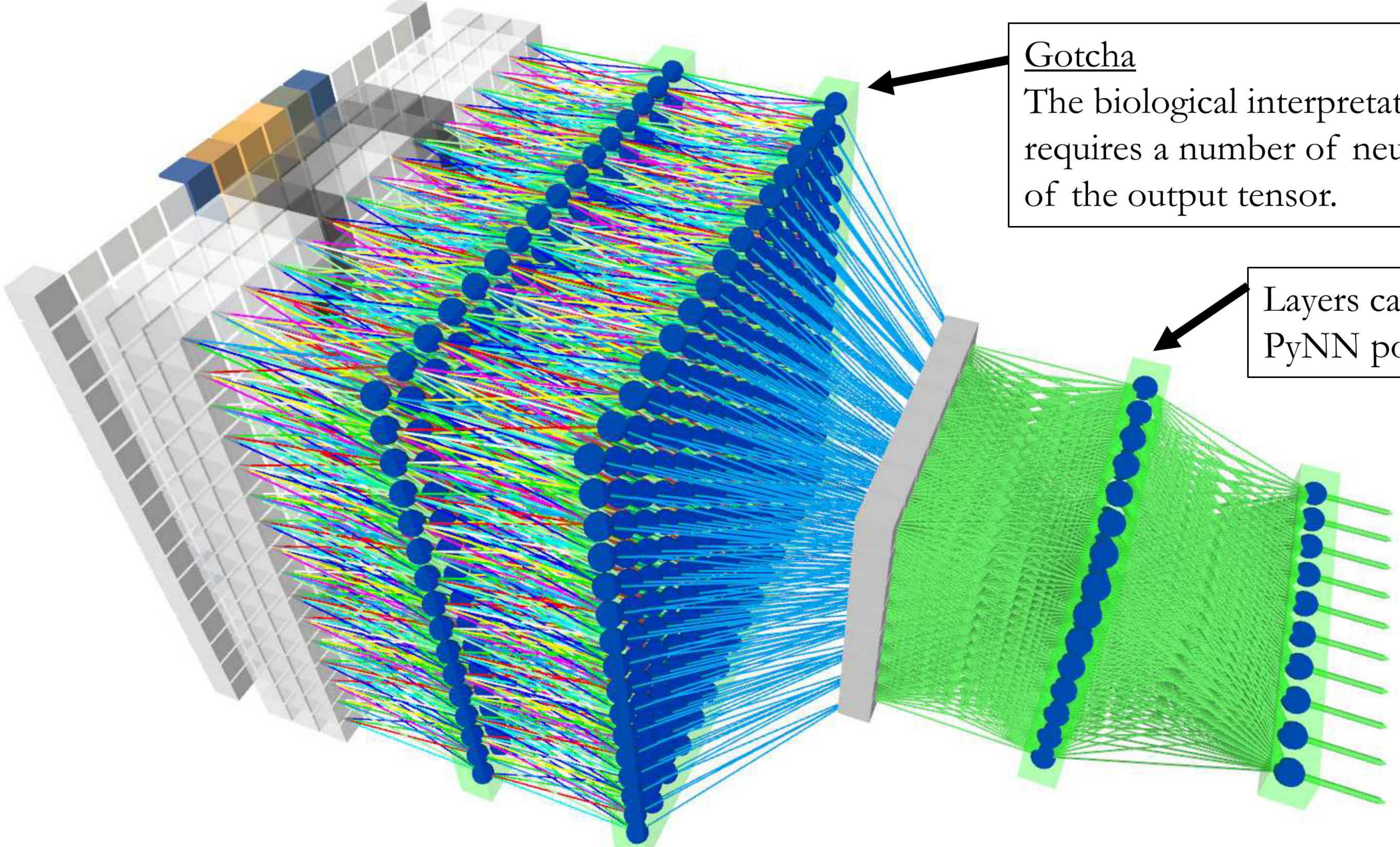
One problem with batch normalization is that the moving averages of the normalization parameters are left in the model after training is complete. This leaves us with four extra parameters for each neuron that are used in determining pre-activations. Before we can export the model parameters to spiking hardware, is it necessary to remove these extra parameters. To accomplish this, we merge them into the weights and biases of each neuron using (2) and (3).

$$\text{NewWeights}(\mathbf{w}_i) = \mathbf{w}_i\left(\frac{\gamma}{\sigma + \varepsilon}\right) \tag{2}$$

$$\text{NewBias}(b_i) = \left(\frac{\gamma}{\sigma + \varepsilon}\right)(b_i - \mu) + \beta \tag{3}$$

# From Tensors to Neurons

| Traditional ANNs | Spiking Nets |
| --- | --- |
| Dense Matrix/Tensor Representations | Sparse Synapse/Neuron Representations |



Gotcha
The biological interpretation of convolution requires a number of neurons equal to the size of the output tensor.

Layers can be converted to PyNN populations

# Impact of Reduced Precision Weights

| Traditional ANNs | Spiking Nets |
|---|---|
| Floating-Point Weight Precision | Weights Often Fixed-Point |

**Table 1.** Accuracy of sharpened and non-sharpened networks at reduced precision. Presented are the mean and range of accuracies for MNIST across ten sample networks each of two types. Dense networks had two hidden layers (512 neurons each) and a 10-hot output encoding. A small convolution network was chosen to give realistic, but conservative estimates of degradation. The topology consists of two Convolution-MaxPool blocks and three dense layers before a 10-hot output layer.

| | | Spiking | | Non-Spiking | |
|---|---|---|---|---|---|
| | Precision | Mean | Range | Mean | Range |
| Dense | float32 | 0.9794 | [0.9784, 0.9820] | 0.9854 | [0.9837, 0.9865] |
| | Q4.16 | 0.9794 | [0.9777, 0.9821] | 0.9854 | [0.9838, 0.9865] |
| | Q4.8 | 0.9786 | [0.9772, 0.9803] | 0.9849 | [0.9836, 0.9866] |
| | Q4.7 | 0.9773 | [0.9757, 0.9800] | 0.9842 | [0.9834, 0.9855] |
| | Q4.6 | 0.9712 | [0.9673, 0.9742] | 0.9798 | [0.9774, 0.9827] |
| | Q4.5 | 0.8679 | [0.7732, 0.9207] | 0.8922 | [0.8385, 0.9447] |
| Convolution | float32 | 0.9815 | [0.9791, 0.9836] | 0.9905 | [0.9896, 0.9914] |
| | Q4.16 | 0.9815 | [0.9789, 0.9835] | 0.9905 | [0.9896, 0.9914] |
| | Q4.8 | 0.9815 | [0.9797, 0.9838] | 0.9905 | [0.9897, 0.9915] |
| | Q4.7 | 0.9802 | [0.9782, 0.9817] | 0.9902 | [0.9894, 0.9916] |
| | Q4.6 | 0.9754 | [0.9714, 0.9795] | 0.9884 | [0.9871, 0.9899] |
| | Q4.5 | 0.9306 | [0.8867, 0.9482] | 0.9752 | [0.9639, 0.9813] |

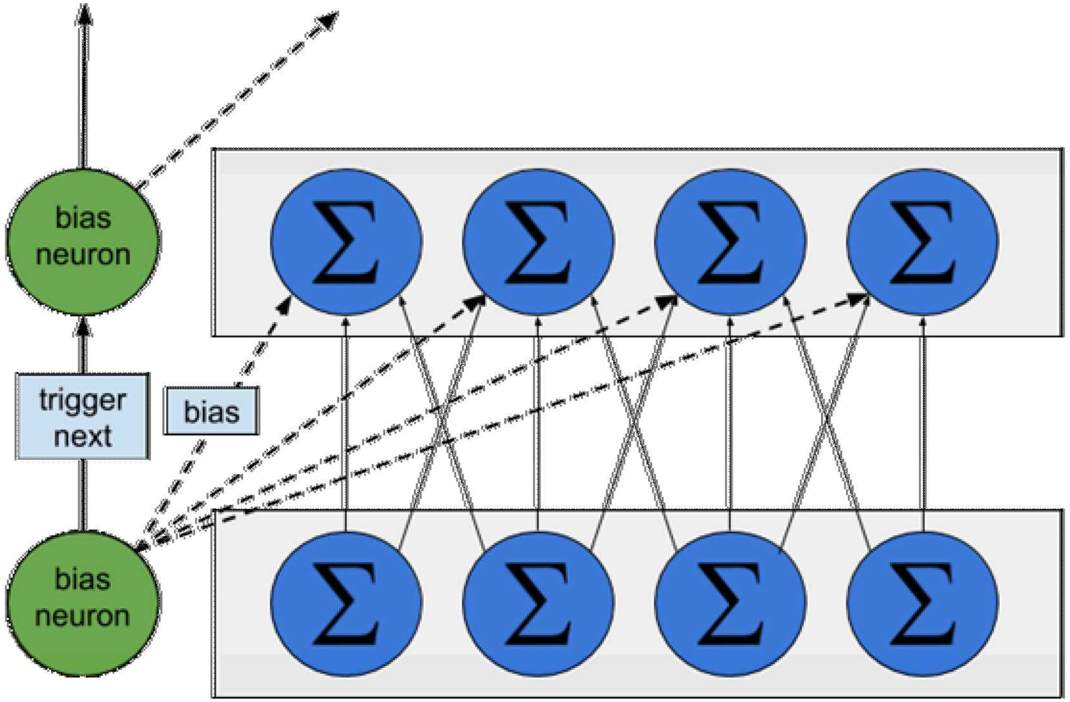
# Translating Biases

| Traditional ANNs | Spiking Nets |
| --- | --- |
| Individual "Neuron" Biases | (Often Shared) Firing Thresholds |

Gotcha

Though one can interpret biases as firing thresholds, PyNN and SpiNNaker make this approach impractical since thresholds are typically shared across all neurons of a given population.

An alternative is to create a network of explicit bias neurons. Bias neurons are daisy-chained from layer-to-layer, with the first layer requiring an additional input to start it off.

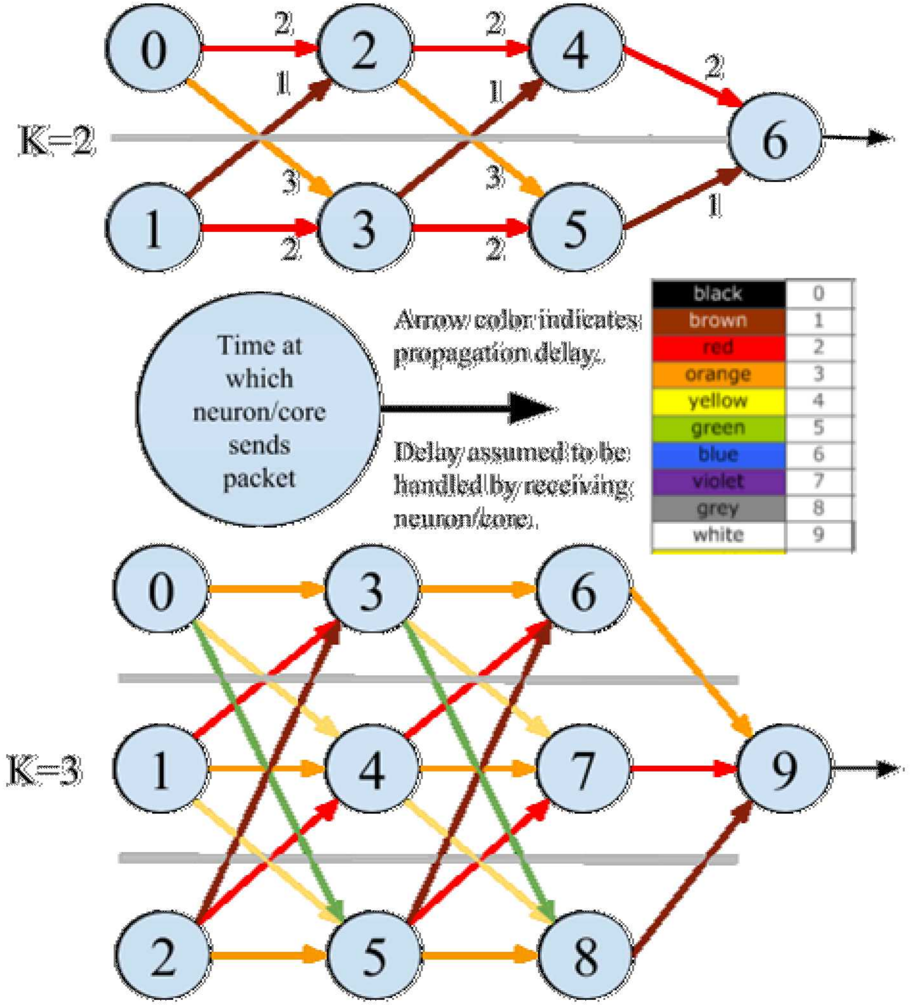


# Time-division Multiplexing (i.e. Temporal Groups)

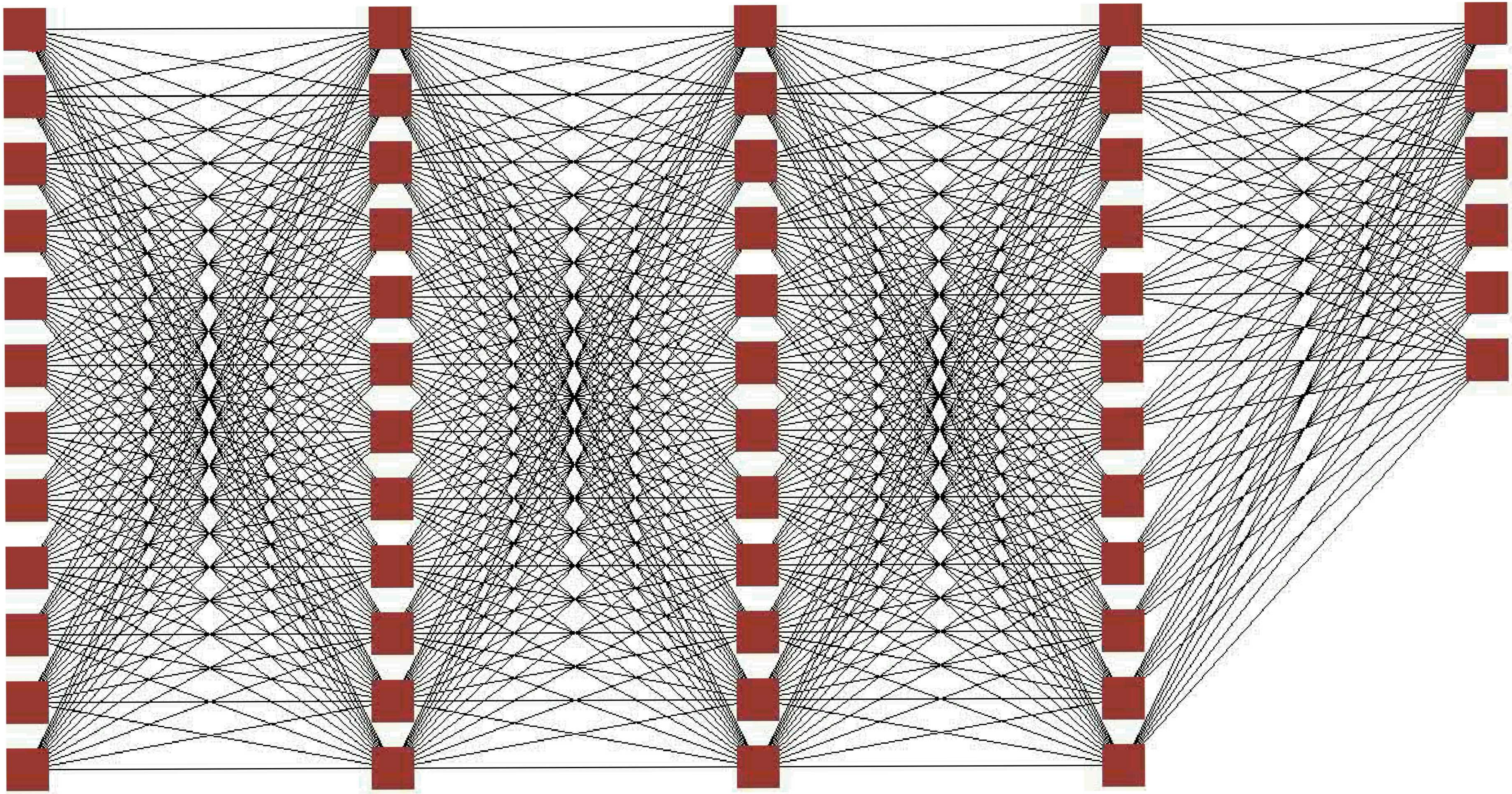
| Traditional ANNs | Spiking Nets |
|---|---|
| Global Clock-Driven Synchrony | Local Event-Driven Asynchrony |

Gotcha
Binary-activation ANNs require global synchrony to produce correct one-shot output. However, their real-time simulation produces synchronous bursts of activity which can overload the communications fabric, breaking global synchrony. Time-division multiplexing using propagation delays is one way to mitigate this problem, but has performance tradeoffs.

Gotcha
Sustained global synchrony is not guaranteed: "Relative drift between boards is possible due to slight variations in clock speed (from clock crystal manufacturing variability), however, this effect is small relative to simulation times…"

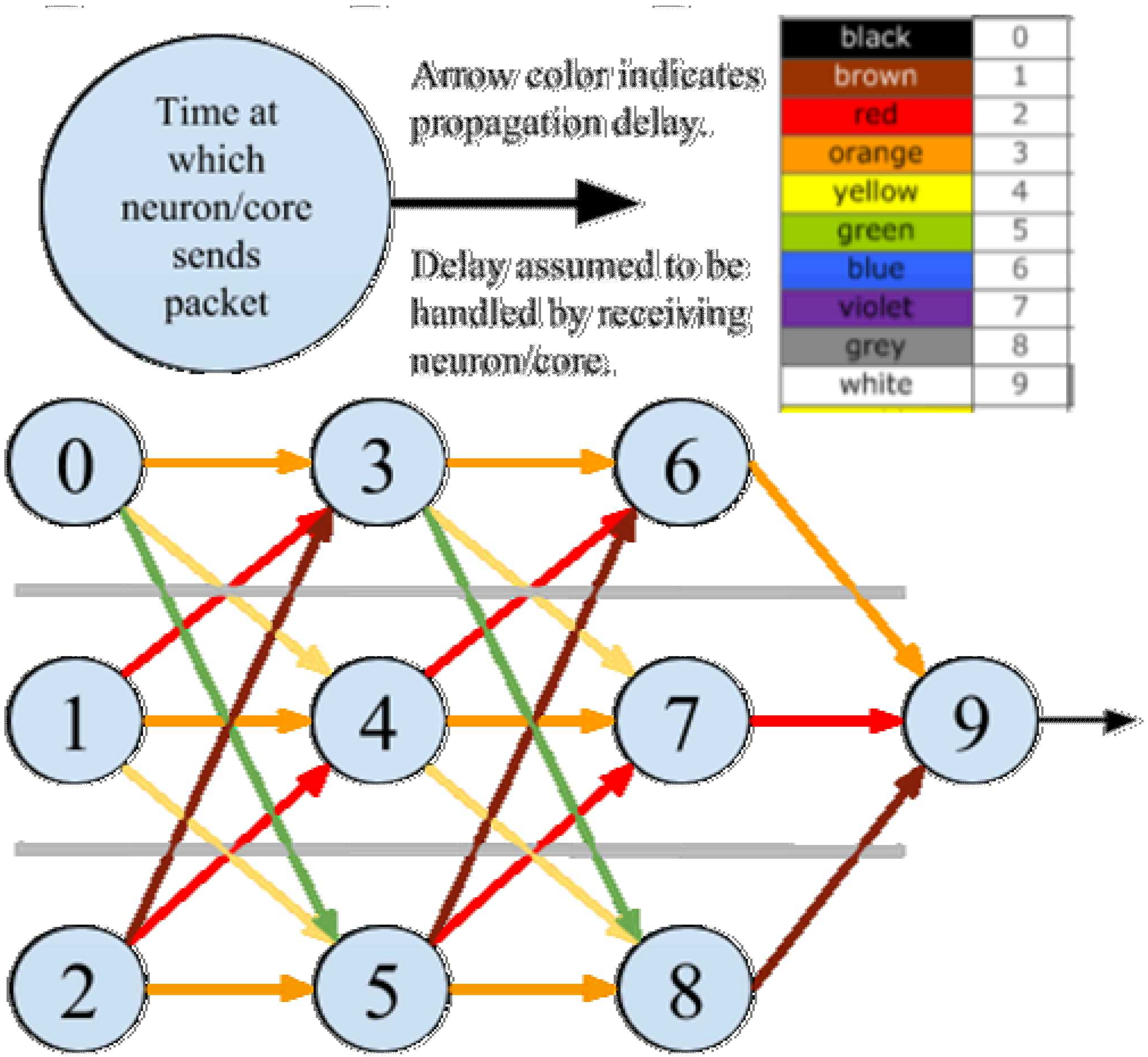


# Time-division Multiplexing (Animation)



# Time-division Multiplexing (Why it Works)

- While SpiNNaker processes synaptic events asynchronously, neuron state updates are local clock-driven synchronous, which we take advantage of to approximate global synchrony.
- When a firing event is generated by a neuron, SpiNNaker immediately transmits the spike packet to the destination cores where it waits in a ring buffer for a time determined by the propagation delay.
- Rather than having all neurons of a presynaptic population fire concurrently, we stagger their firing and use delays to ensure all synaptic events from the source population induce a membrane potential at the correct time-increment. Thus, while firings of source population neurons are not synchronous, their effects downstream are.
- Basically, the ring buffers are repurposed to reduce packet congestion.



# Time-division Multiplexing (Caveats)

- In SpiNNaker, delays greater than 10ms are too long for the ring buffers and so require the use of the "DelayExtentionVertex" application, which effectively doubles the required cores/neurons. This also causes the spike source array to be split over many cores like a normal population.
- The maximum delay induced by multiplexing is (K*2 - 1), where K is the number of temporal groups. Thus, a maximum of 5 temporal groups can be employed without invoking the above mechanism.
- The maximum number of temporal groups supported should be 72 based on the following: "While this application solves the problem of simulating extended delays, it cannot do so indefinitely and an effective new upper limit of 144 delta-t is enforced due to DTCM constraints."
- The need to increase the time-scale-factor from 5 to 14 may be due to the following: "An additional row must be included to identify spikes traveling directly from the presynaptic core, and also those sent from each individual delay stage of the delay extension. This increased master population table size can be costly to search, and detrimental for real-time performance (see section 4.2)."

# Future Work

- Real-time I/O: We'd like to characterize the latency and throughput when using alternatives to the SpikeSourceArray and potentially also play with the SpiNN-Link interface. Currently: "Each chip additionally has an Ethernet controller, although in practice only one chip is connected to the Ethernet connector on each board... Communication with other chips on a board from outside of the machine must therefore go via the Ethernet chip; system-level packets are used to effect this communication between chips."
- Looking into new input encoding methods. For example: reduced-precision binary coding of inputs. Input layer channels split into binary at desired precision and each connection weight is divided logarithmically between the resulting new connections (kudos to Mike Davies for suggesting the general concept). This has been tested in Tensorflow but not yet on SpiNNaker. Hopefully it'll be able to handle the increased demand on I/O.
- Laterally connected pseudo-recurrent tiles for image processing.
- Further experiments to better understand communication bottlenecks of the current version. Also, we've heard the SpiNNaker 2.0 prototype is clocked at 500MHz [3] which is 2.5 times that of the current version.

# SpiNNaker (References)


[1] Rhodes, Oliver, et al. "sPyNNaker: A Software Package for Running PyNN Simulations on SpiNNaker." Frontiers in neuroscience 12 (2018).

[2] Rowley, Andrew GD, et al. "SpiNNTools: the execution engine for the SpiNNaker platform." arXiv preprint arXiv:1810.06835 (2018).

[3] Liu, Chen, et al. "Memory-efficient Deep Learning on a SpiNNaker 2 prototype." Frontiers in neuroscience 12 (2018).

[4] Temple, Steve. "SARK - SpiNNaker Application Runtime Kernel" (http://spinnakermanchester.github.io/docs/sarkV200.pdf) (2016)

[5] Serrano-Gotarredona, Teresa, et al. "ConvNets experiments on SpiNNaker." Circuits and Systems (ISCAS), 2015 IEEE International Symposium on. IEEE, 2015.

[6] Brown, Andrew, et al. "SpiNNaker-programming model." IEEE Transactions on Computers 1 (2015): 1-1.

[7] Furber, Steve B., et al. "Overview of the spinnaker system architecture." IEEE Transactions on Computers 62.12 (2013): 2454-2467.

[8] "SpiNNaker datasheet version 2.02 6 January 2011" (http://spinnakermanchester.github.io/docs/SpiNN2DataShtV202.pdf) (2011)

# Thanks!

Questions?

# SpiNNaker (Hardware Overview)

Chip Hardware Specs:

- 18 ARM968 cores clocked at 200MHz (5ns/instruction)
- Current chips are implemented in UMC 130 nm silicon. [6]
- "Each chip uses up to 1W when all the processors are fully utilized, ..." [2]
- 32kB ITCM (Instruction Tightly Coupled Memory) per core.
- 64kB DTCM (Data Tightly Coupled Memory) per core.
- 128MB shared SDRAM per chip. (1Gbit)
- 5ns/word DTCM access speed (word = 32 bits) (entire read start-to-finish takes just 1 instruction).
- 100ns/word SDRAM access via bridge, subject to contention with other cores.
- 10ns/word SDRAM -> DTCM DMA transfer after fixed overhead (>= 15ns), independent of processor.
- 200ns packet routing time for on-chip router.

SpiNN-5 48-chip Board:

- "We budget for the nodes dissipating up to 1W, and with other components a board will dissipate up to 75W." [7]