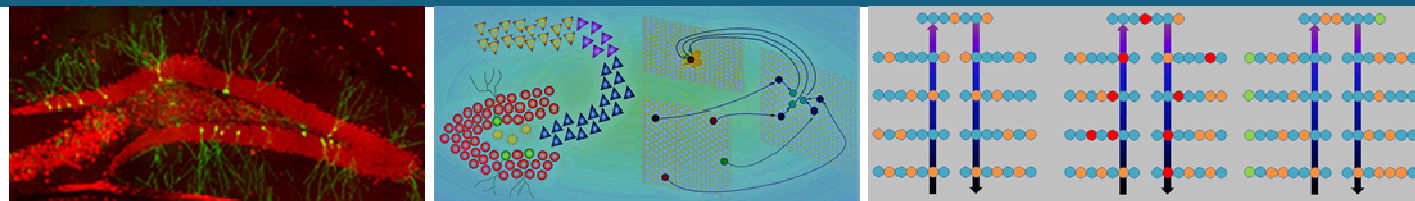


Whetstone

An Accessible, Platform-Independent Method for Training Spiking Deep Neural Networks for Neuromorphic Processors



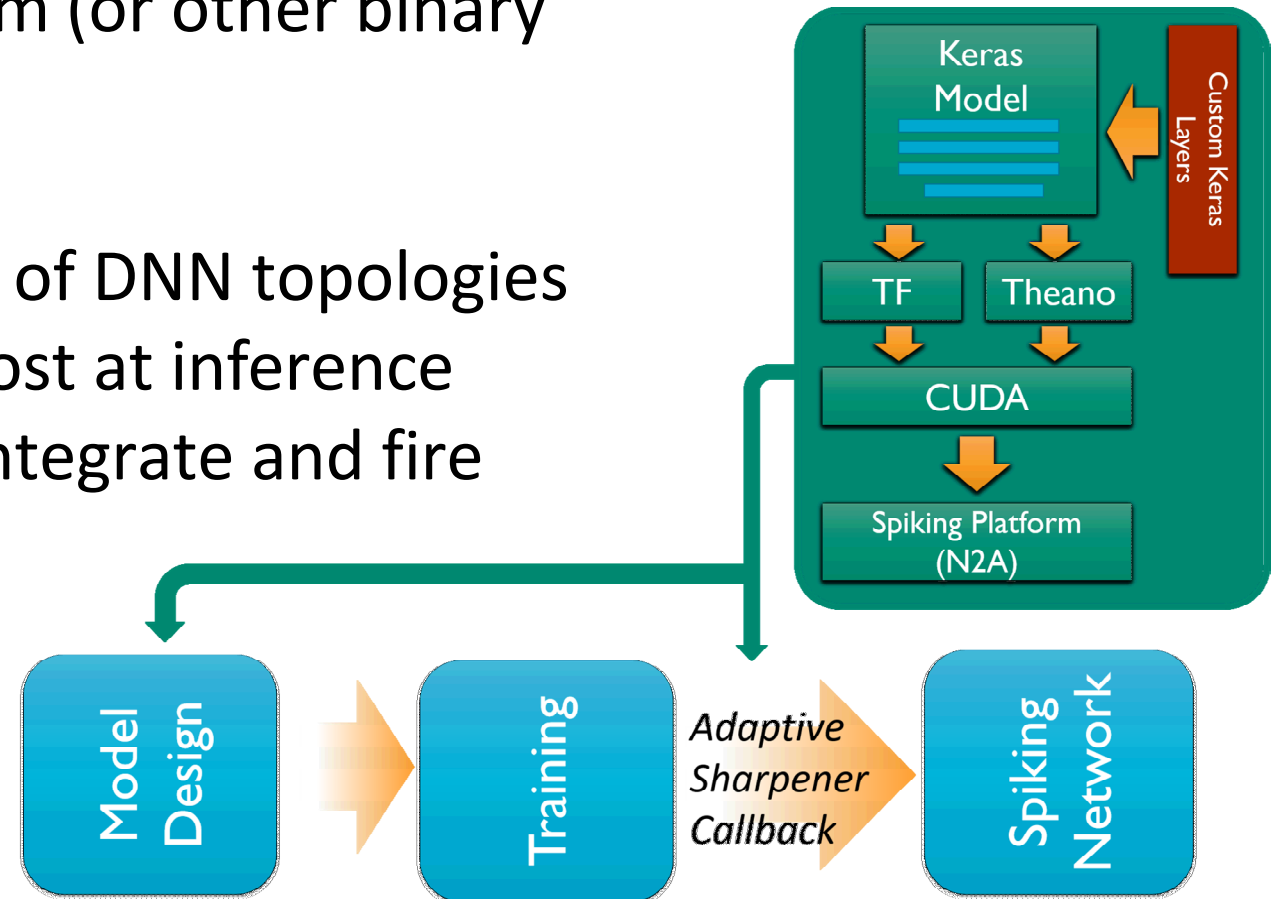
William M. Severa*, Craig M. Vineyard, Ryan Dellana
and James B. Aimone



Sandia National Laboratories is a
multimission laboratory managed and
operated by National Technology and
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.

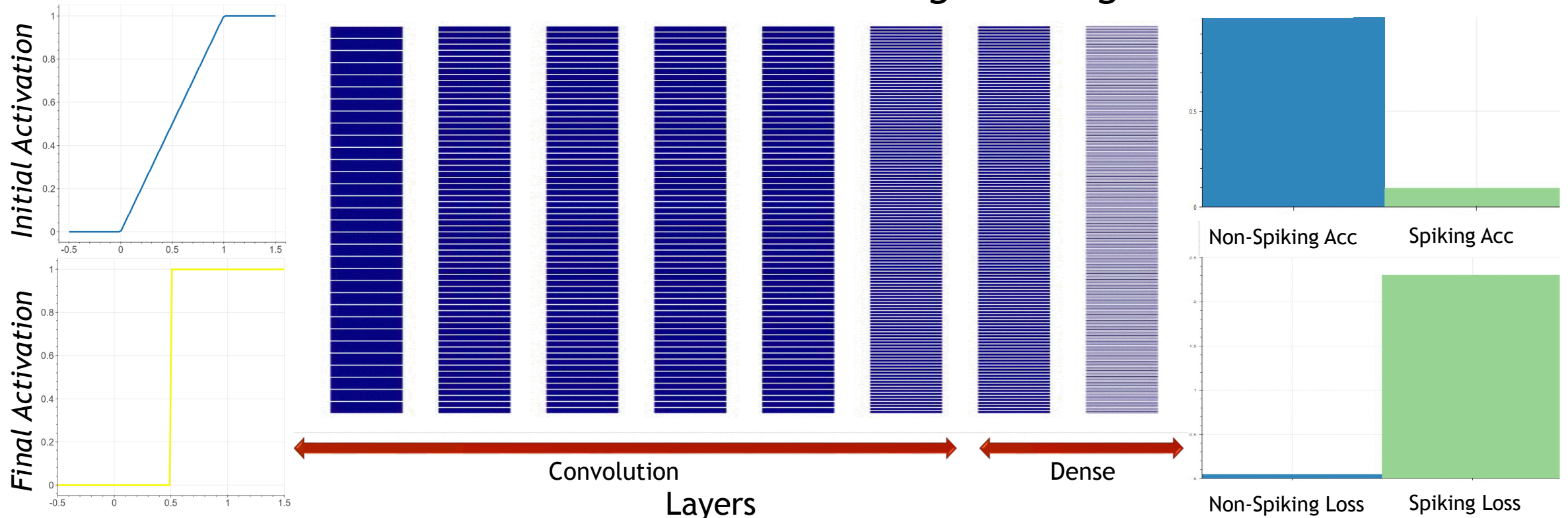
Whetstone provides a drop-in mechanism for tailoring a DNN to a spiking hardware platform (or other binary threshold activation platforms)

- Hardware platform agnostic
- Compatible with a wide variety of DNN topologies
- No added time or complexity cost at inference
- Simple neuron requirements: Integrate and fire



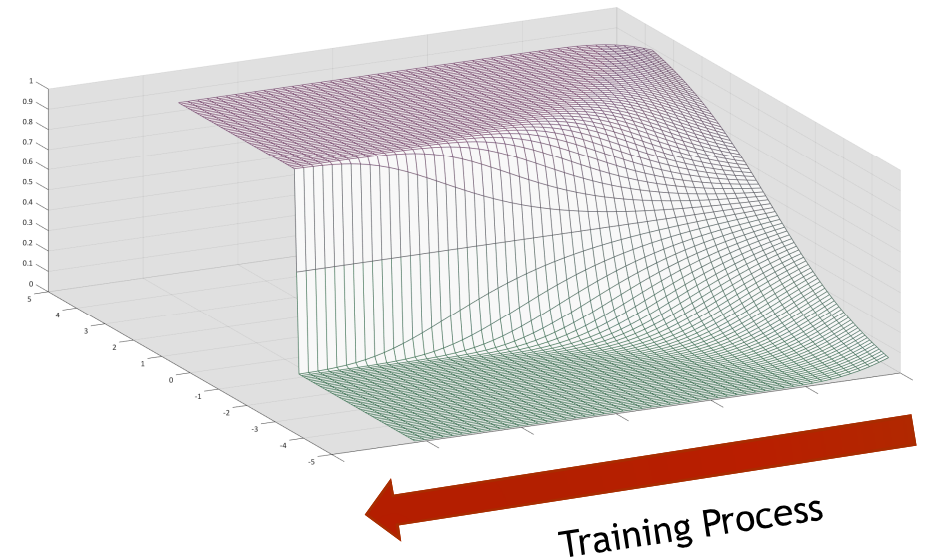
The real challenge for deep learning on spiking is the threshold activation function.

Using Whetstone, activation functions converge to a threshold activation *during training*.



Whetstone Overview

- Generally, gradient descent generates a sequence of weights A_i with the goal of minimizing the error of $f(A_i x)$ in predicting the ground truth y .
- We generalize this by replacing the activation function f with a sequence f_k such that $f_k \rightarrow_{L_1} f$, where f is now the threshold activation function.
- Now, the optimizer must minimize the error of $f_k(A_i x)$ in predicting y .
- Since the convergence in **neither i nor k is uniform**, this is a mathematically dangerous idea
- However, with a little care and a few tricks, the method reliably converges in many cases.





When/Where do we decide to 'sharpen' the activations?

- 1) Bottom-up Sharpening (The 'toothpaste tube' method)
 - Begin sharpening at the bottom layer
 - Wait until previous layer is fully sharpened
 - Increases stability of convergence
- 2) Adaptive Sharpening Callback
 - Hand-tuning sharpening rates is hard
 - Instead, use loss as a guide for an *adaptive sharpener*
 - Adaptive sharpener implemented as a callback automatically adjusts sharpening based on loss thresholds

Original Model Example

```
⋮  
model.add(Dense(256))  
model.add(Activation('relu'))  
model.add(Dense(10))  
model.add(Activation('softmax'))  
⋮  
model.fit(x,y)  
⋮
```

Modified Model Example

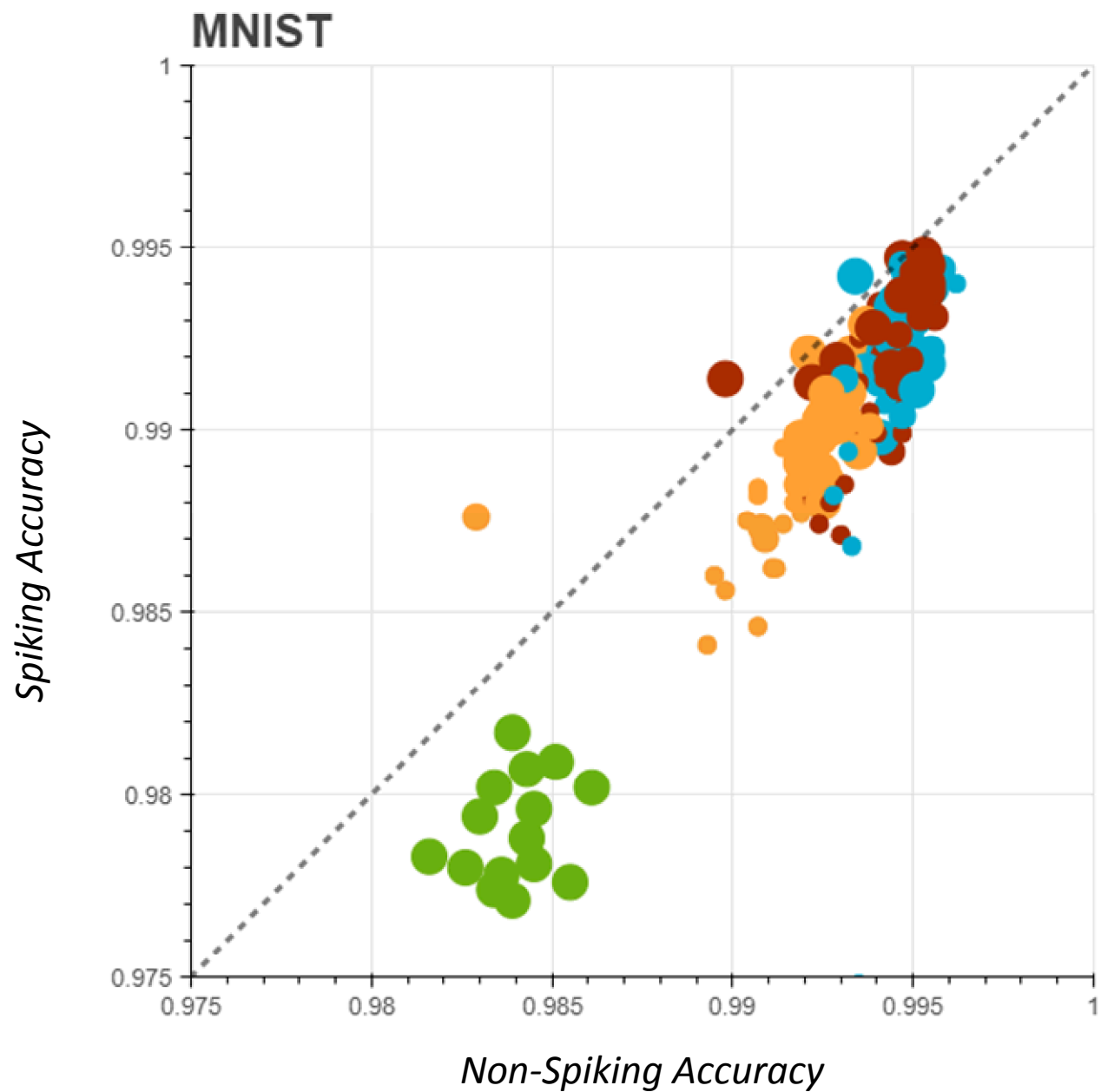
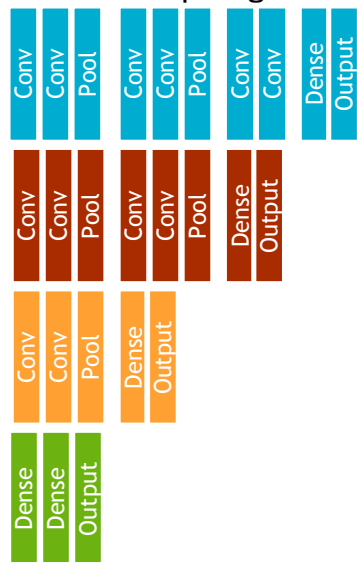
```
⋮  
model.add(Dense(256))  
model.add(Spiking_BRelu())  
model.add(Dense(10))  
model.add(Spiking_Brelu())  
Model.add(Softmax_Decode(key))  
⋮  
sharpener = AdaptiveSharpener()  
model.fit(x,y,callbacks=[sharpener])  
⋮
```

Preliminary Results

● ● ● Filter Size

7x7 5x5 3x3

Network Topologies

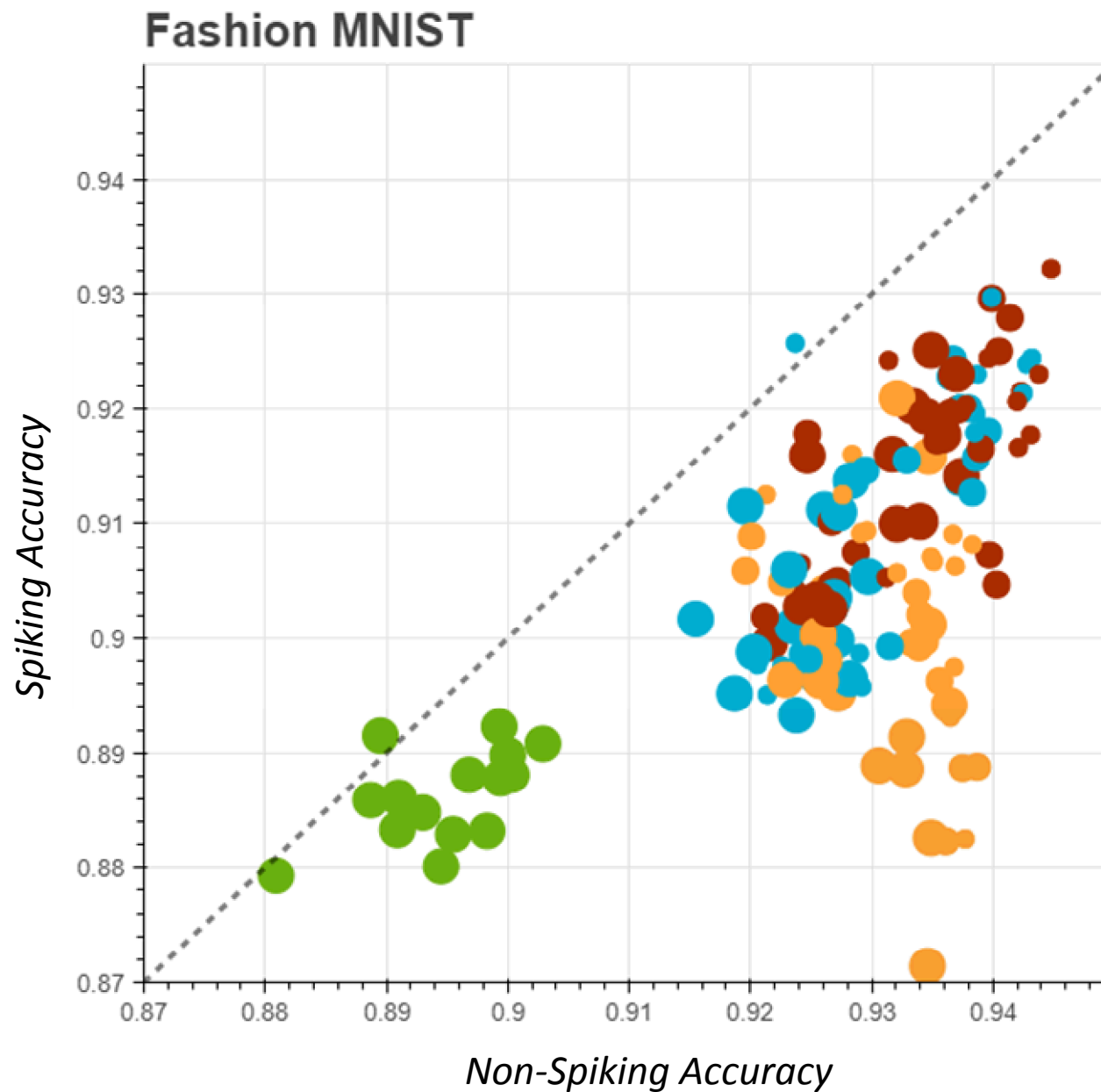
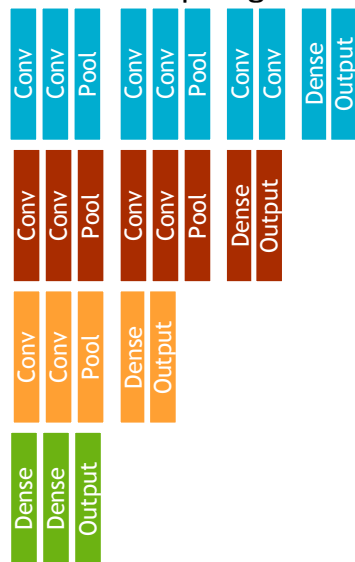


Preliminary Results

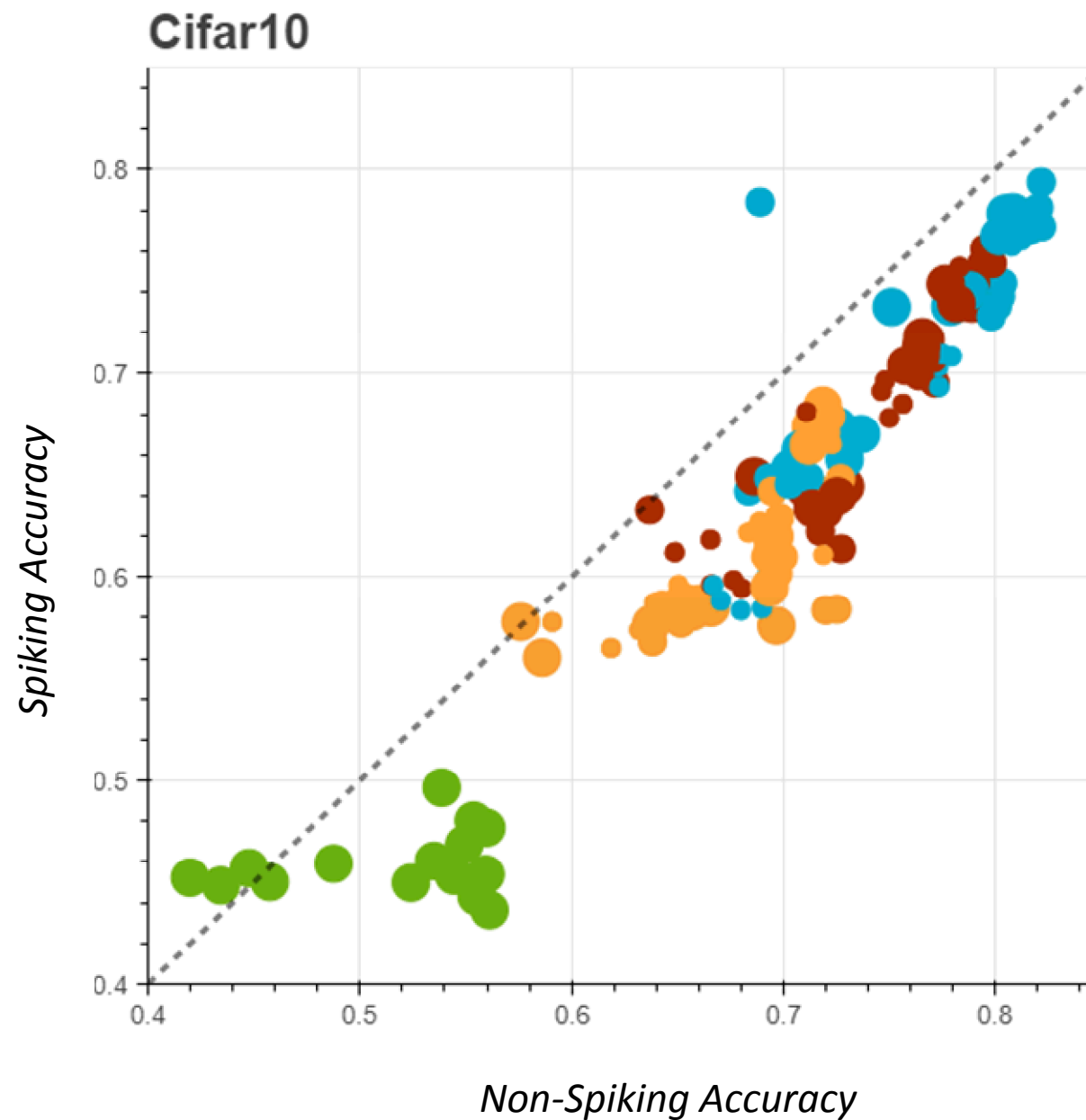
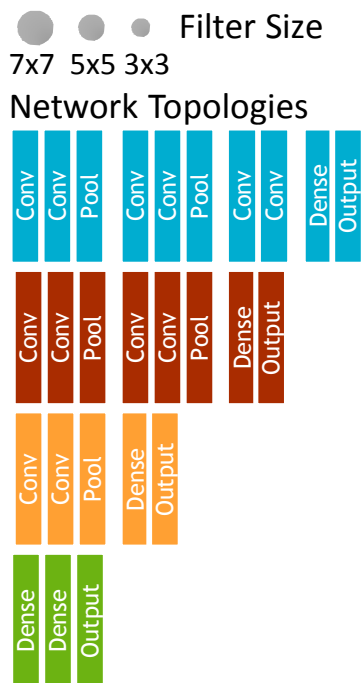
● ● ● Filter Size

7x7 5x5 3x3

Network Topologies



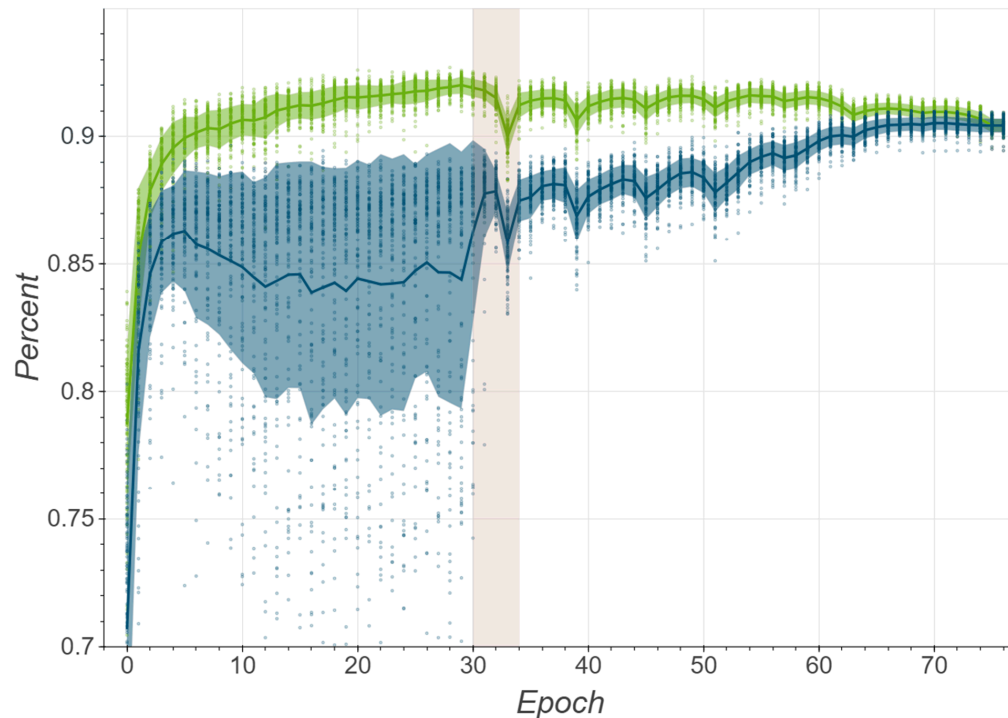
Preliminary Results



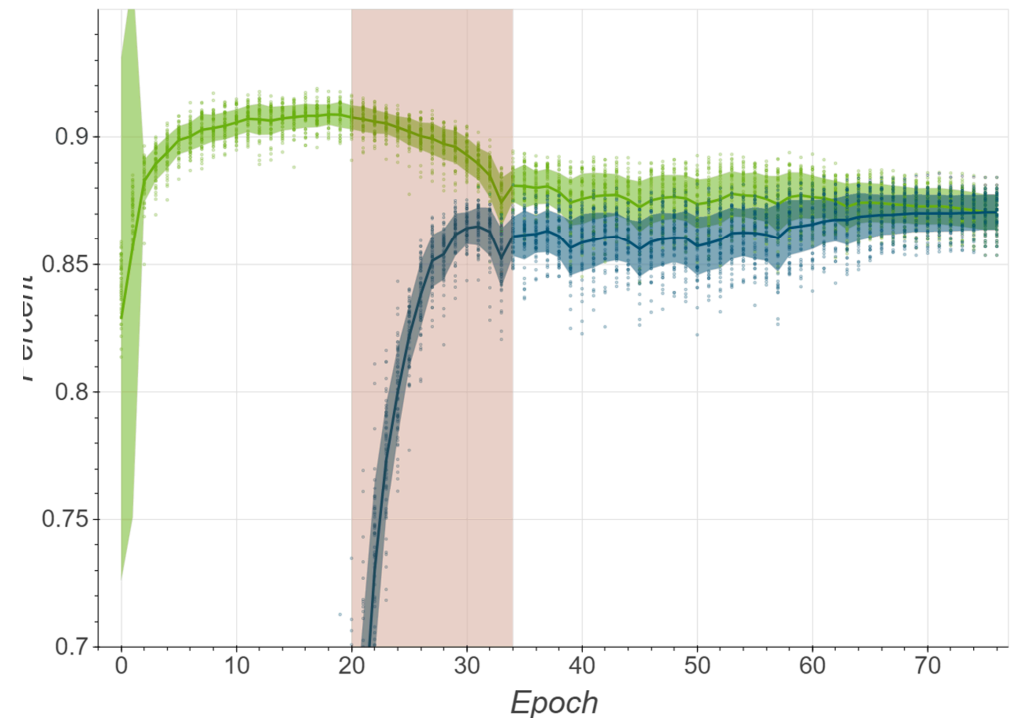
Established Deep Learning Techniques

- Batch Normalization helps training stability and network performance
- Improvements across network sizes
- Sharpening loss, particularly on first sharpening layer, is significantly less
- At inference time, bias (threshold) and weights are modulated according to stats collected during training

Fashion MNIST with Batch Normalization



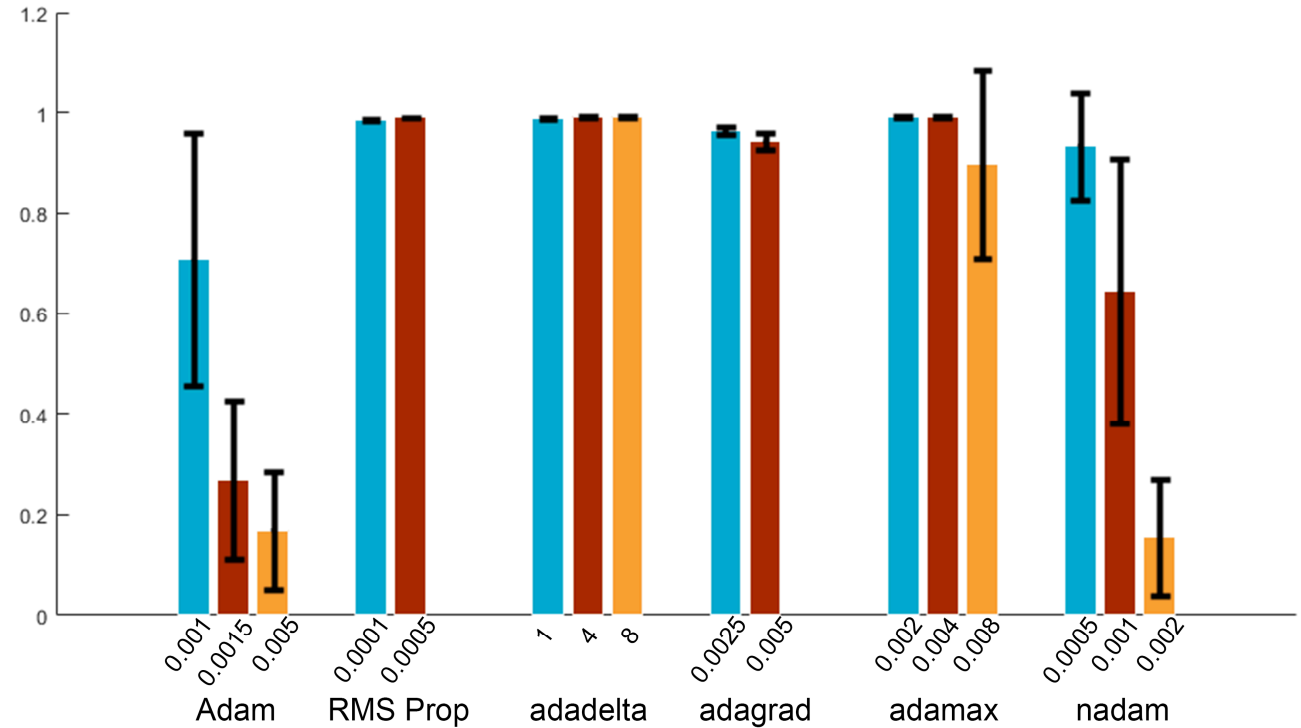
Fashion MNIST without Batch Normalization



Established Deep Learning Techniques

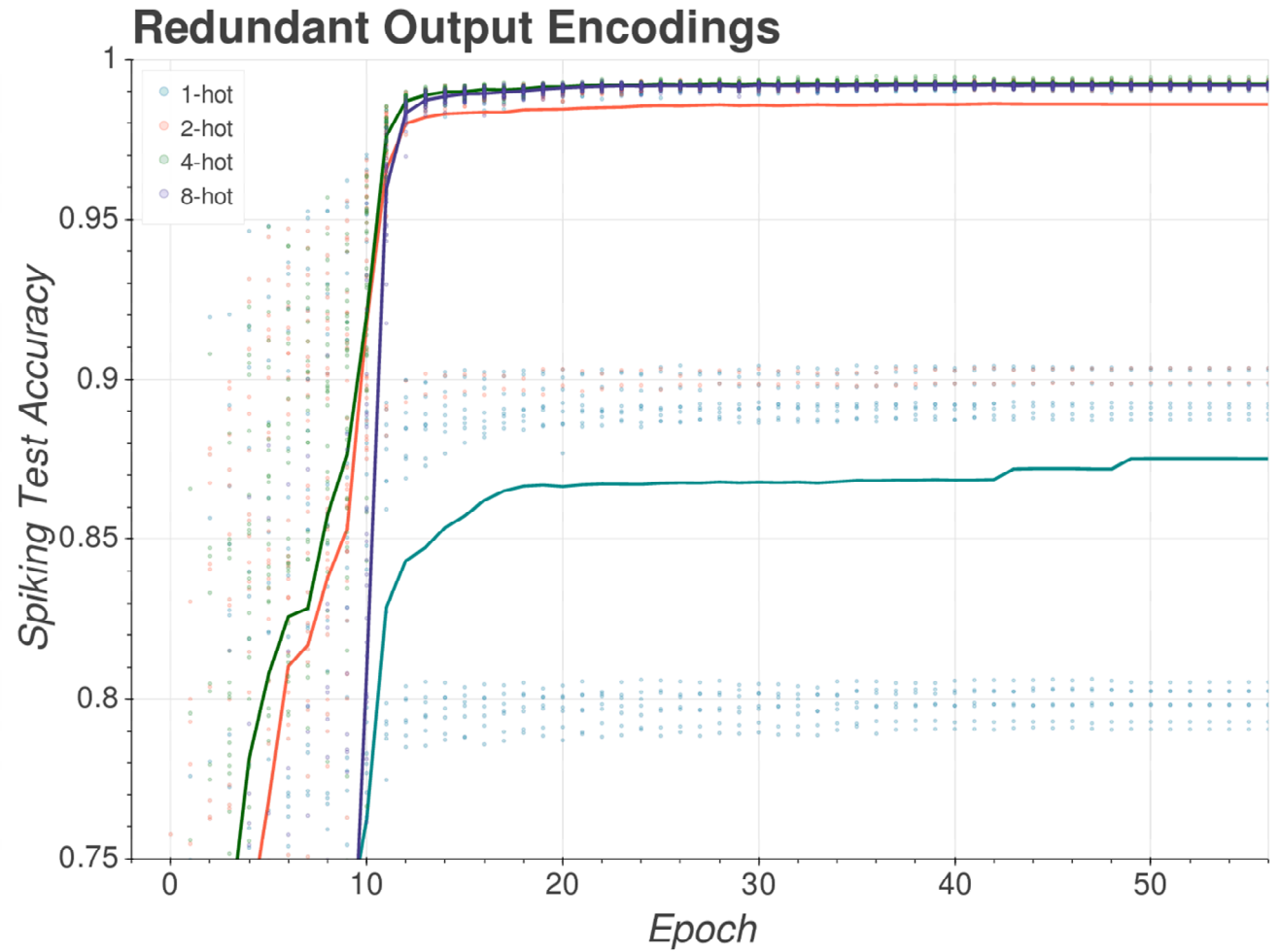
- Sharpening process is sensitive to optimizer selection
- Adaptive optimizers often work better
- Learning rate modulation by moving average seems to help stability
- A custom Whetstone-aware optimizer is in early stages

Accuracy Across Optimizers and Learning Rates



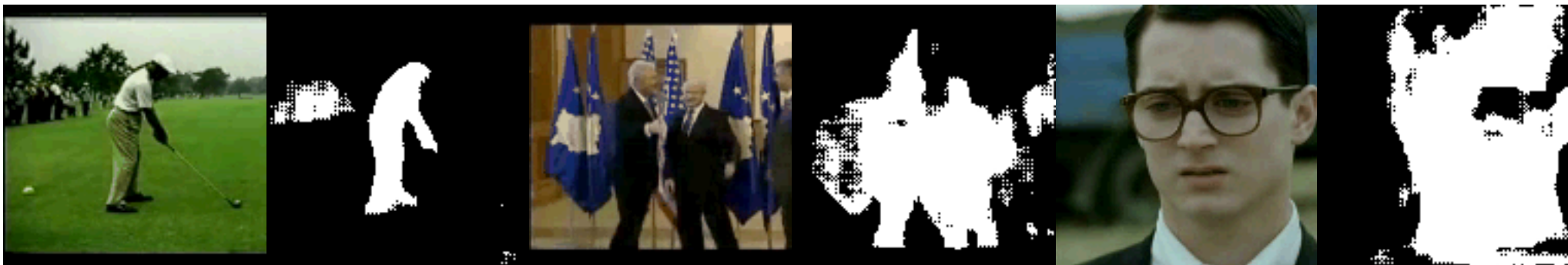
Established Deep Learning Techniques

- The trained neurons can be unreliable
- Redundant output encodings help mitigate this problem
- Similar to ensemble methods
- Reactive neurons feed into softmax during training (for classification)
- During inference, 'best-matched' group is used
- On simple datasets, 4-way redundancy is sufficient

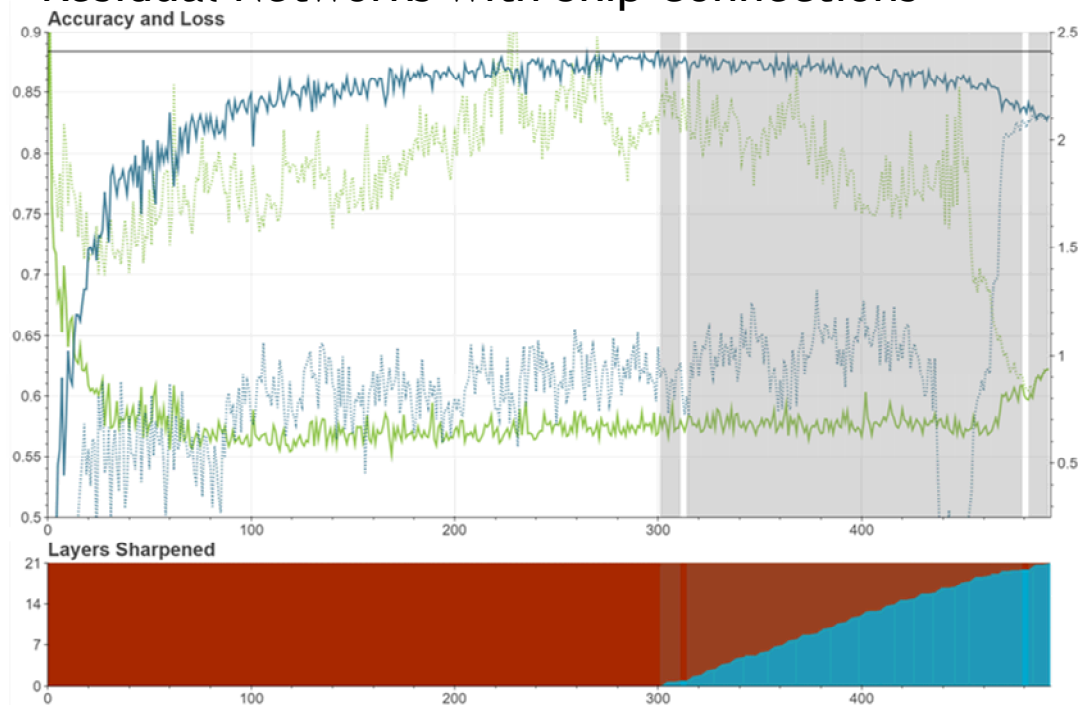


Effective Across Various Topologies, Datasets, and Tasks

Semantic Segmentation (Trained on COCO Dataset; Videos from HMDB51 Dataset)



Residual Networks with Skip Connections



Autoencoders



DQN



Neuromorphic hardware platforms are appealing for a wide variety of low-power, embedded applications

Sophistication and expertise required to make use of these platforms creates a high barrier of entry

Whetstone enables deep learning experts to easily incorporate spiking hardware architectures

Enabling Wide and Easy-to-Implement Adoption

Networks are portable and hardware-agnostic

Low barrier of entry; built on standard libraries (Keras, Tensorflow, CUDA, etc.)

No post-hoc analysis; no added time complexity

Only simple integrate-and-fire neurons are required

Compatible with standard techniques like dropout and batch normalization