慶應義塾
Keio University

JST MIRAI 未来社会創造事業

FAWS: Fault-Aware Weight Scheduler for DNN Computations in Heterogeneous and Faulty Hardware

Shaswot Shresthamali, Yuan He, Masaaki Kondo

Kondo Laboratory

Keio University

ISPA 2022

Melbourne (Virtual)

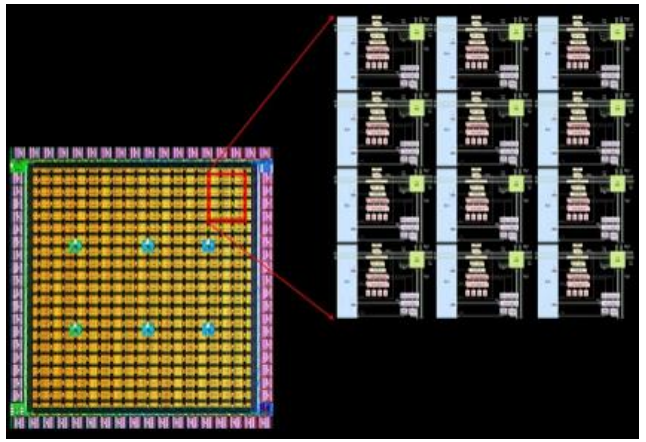
17-19 Dec 2022

(Faulty) DNN Accelerators

- Accelerators consist of many (possible faulty) Processing Elements (PEs)

- Faulty operation may be due to
  - Marginal Operation
  - Manufacturing Defects
  - Degradation with age

https://morethanmoore.substack.com/p/teslas-dojo-supercomputer-deep-dive

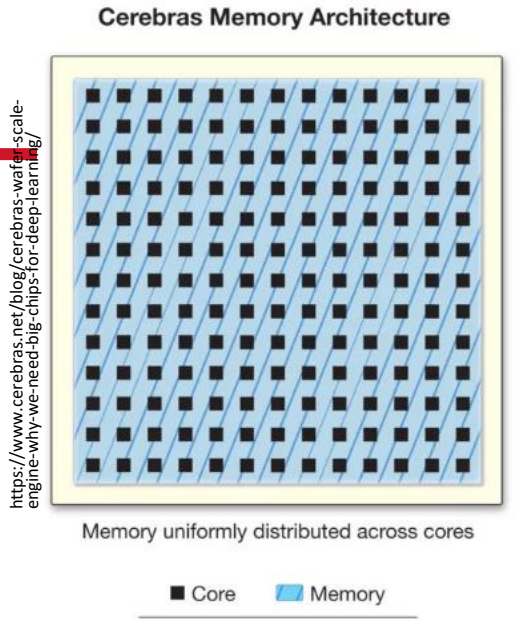


Tesla Dojo D1 has **354** PEs per chip

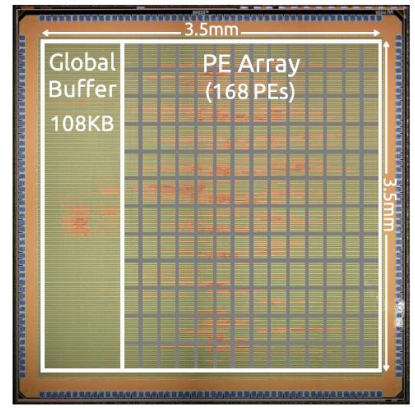
https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

NVIDIA GA100 with **128** SMs. Each SM has multiple CUDA cores with INT32, FP32 an FP64 capabilities

https://www.cerebras.net/blog/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/



The Cerebras Wafer Scale Engine has **400,000** AI-optimized compute cores in one large chip

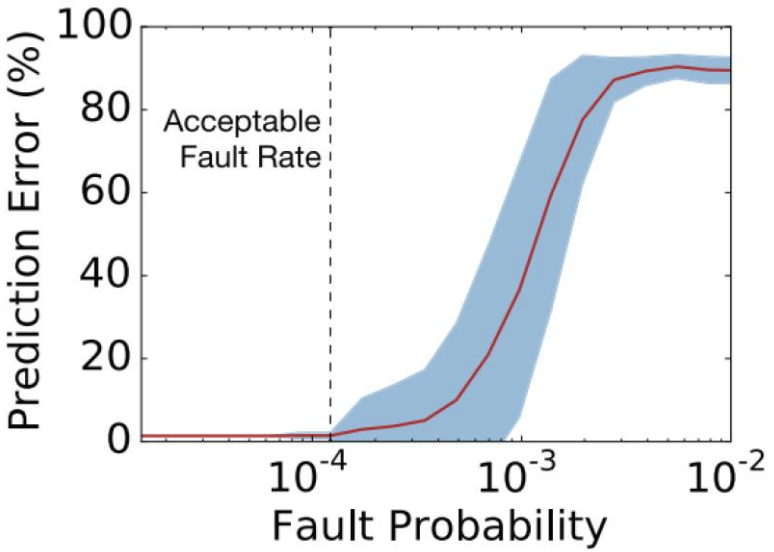


Eyeriss – **168** PEs per chip

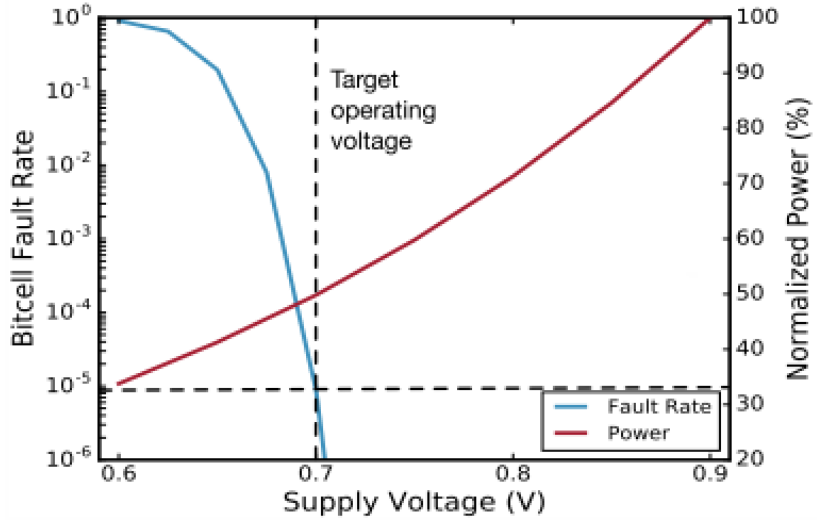
Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE journal of solid-state circuits 52.1 (2016): 127-138.

DNNs are resilient to computational errors

- DNNs have overprovisioned parameters
- Computations within a layer are
  - independent
  - distributed
  - parallel
- DNNs degrade *gracefully* with increasing error probabilities
- Some degradation of model accuracy is tolerable in order to extract
  - higher energy efficiency (lower Vdd)
  - lower latency (reduced precision)
  - E.g., Approximate computing
    - Approximate arithmetic – e.g., linear approximations
    - Approximate hardware – e.g., reduced precision



Reagen et al. - 2016 - Minerva Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators



Related Works

- Approach: Leverage DNN error-resilience for
  - Power Savings
  - Lower Latency/ Higher Throughput

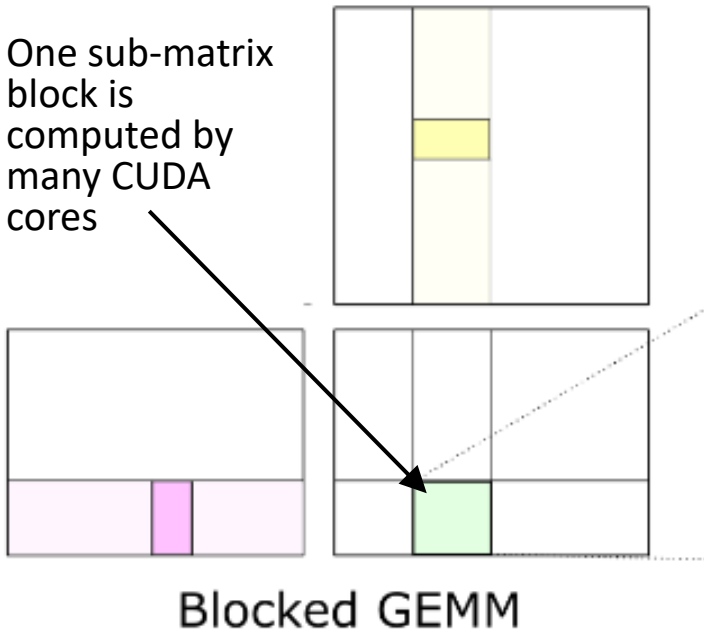
- Voltage Scaling *(MINERVA, ARES)*
  - Reduce SRAM voltage and increase Bit Error Rate (BER)
  - Detect and correct errors in PEs (Processing Elements)

- Approximate Computing (*AxNN, AxTrain, ApproxANN*)
  - Use inexact arithmetic for low priority neurons
  - Requires neuron sensitivity analysis and retraining

- Reduced/mixed precision computation (*HAQ, RAPiD*)
  - Use hardware-in-loop optimization method
  - Specialized accelerator architecture for mixed precision

Limitations of previous approaches

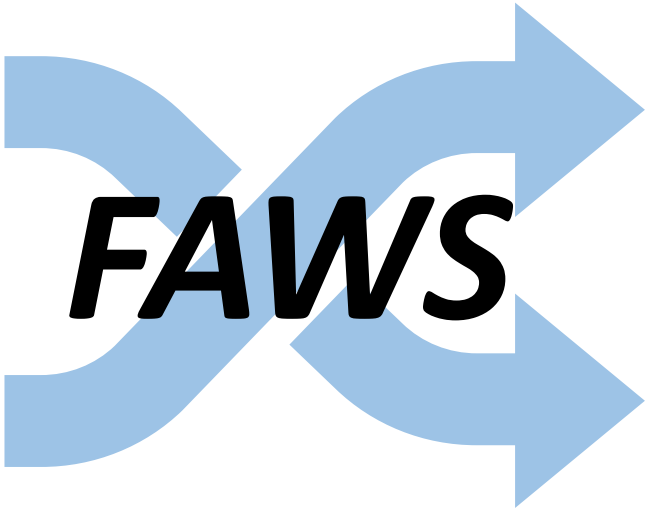
- Previous works focus on analyzing *neuron sensitivity*
  - Allocate non-critical neurons to compromised hardware [AxNN, AxTrain, ApproxANN]
  - Use model-specific/hardware-specific optimizations [HAQ, RAPiD]
- A neuron is a *computational concept.*
- In reality, neurons (computations) are spread out among many PEs during parallelization
  - One neuron is computed using multiple PEs
  - A PE maybe reused for computation of multiple neurons
  - One-to-one mapping between computations and PEs does not exist.
- Unimportant **computations** need to be scheduled to compromised hardware
  - Without interfering with accelerator optimizations – cache misses/shared memory/banking conflicts etc..



https://developer.nvidia.com/blog/cutlass-linear-algebra-cuda/

Motivation

- Trained DNN models are deployed in various hardware accelerator platforms
  - Edge devices in IoT systems
  - Different accelerator types – GPUs, TPUs etc.
- Hardware platforms have different ***fault profiles*** that degrade performance
  - Fault profile – where do faults occur in the hardware with what probability
- Recovering performance by retraining, reoptimizing is not practical
  - Lack of access to training data, computation resources
  - Too many variations
- We need a general method to reschedule computations (for performance recovery) that
  - treats DNN as a black box (model agnostic)
  - is independent of accelerator hardware (hardware agnostic)



Fault-Aware Weight Scheduler (*FAWS*)

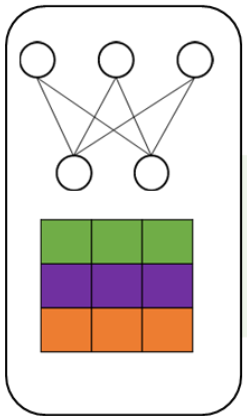
Proposed Solution: *FAWS*

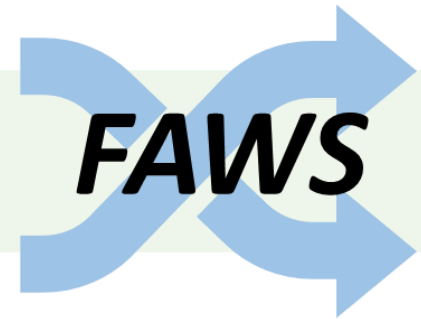
- Focus is only on DNN inference (for now)
- Schedule important ***computations*** in reliable hardware by shuffling the rows of the matrix during multiplication
- n rows → $n!$ permutations [search space is too large]
  - Use GA (Genetic Algorithm) to find a good shuffling order

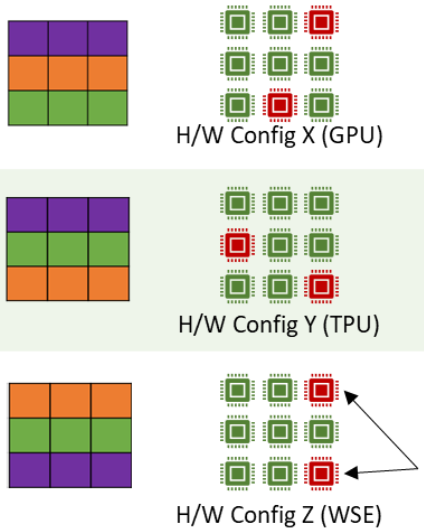
FAWS shuffles the matrix rows.
Shuffling order is found using *GA*.

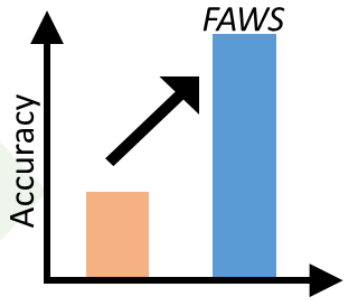
Naïve Scheduling in faulty hardware causes severe degradation.

Inference involves many matrix multiplications









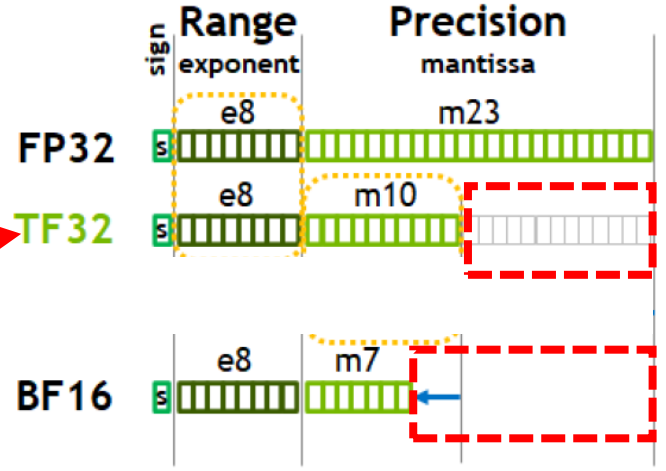
FAWS can recover significant accuracy even with faulty hardware

Modeling Hardware Faults

We simulate faults using bit-level error injection during inference computation

- **Flip-to-0/1:** a random exponent bit is stuck at either 0 or 1
  - Due to permanent damage e.g., shorts/opens
- **Bitflip:** the value of a random exponent bit is flipped.
  - intermittent faults due to timing delays, crosstalk, marginal operation
- **Reduced Precision**: some blocks of mantissa bit are zeroed out
  - TF32 datatype
  - BF16 datatype

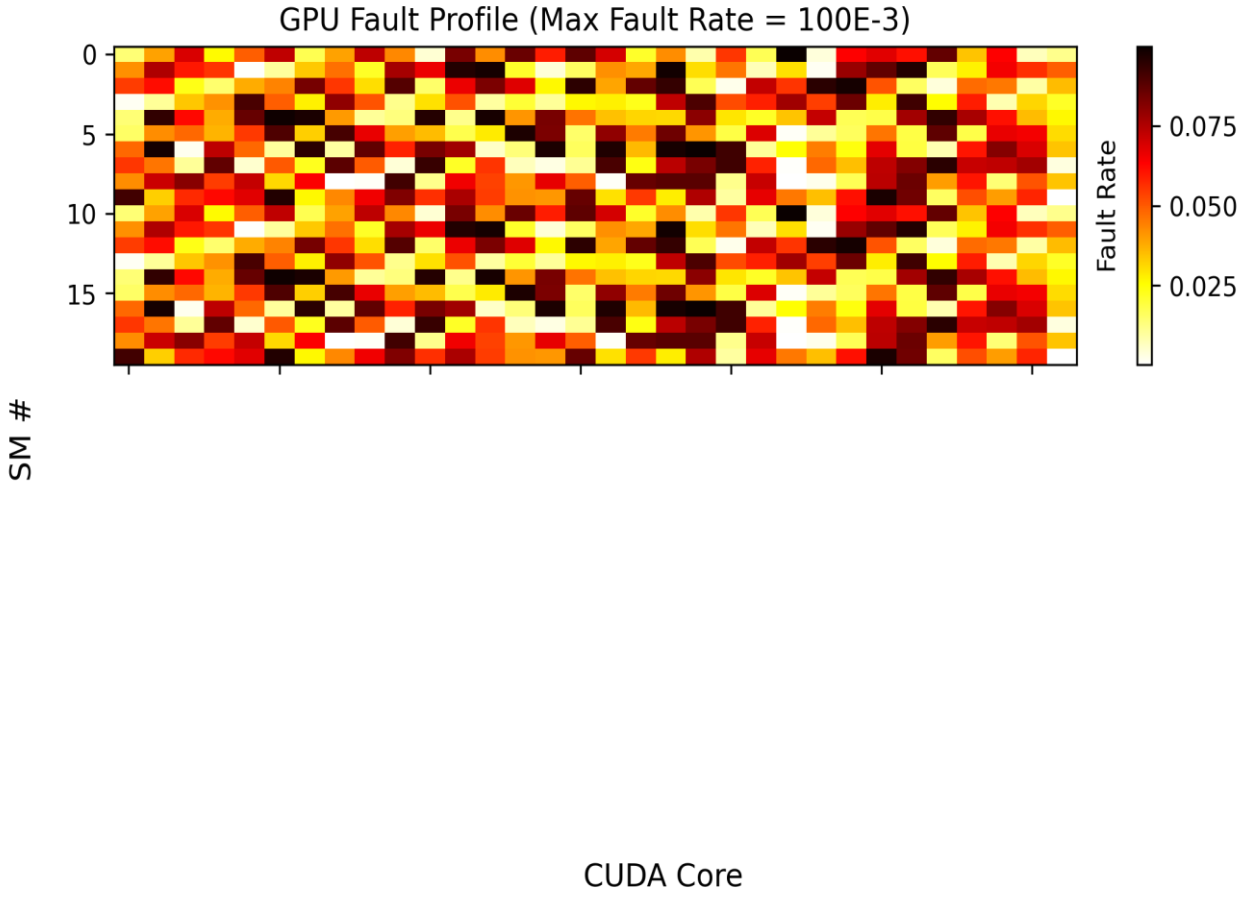
Assuming faults occur in SRAM of CUDA cores of an NVIDIA GPU



NVIDIA A100 Tensor Core GPU Architecture v1.0

Fault Simulation Methodology

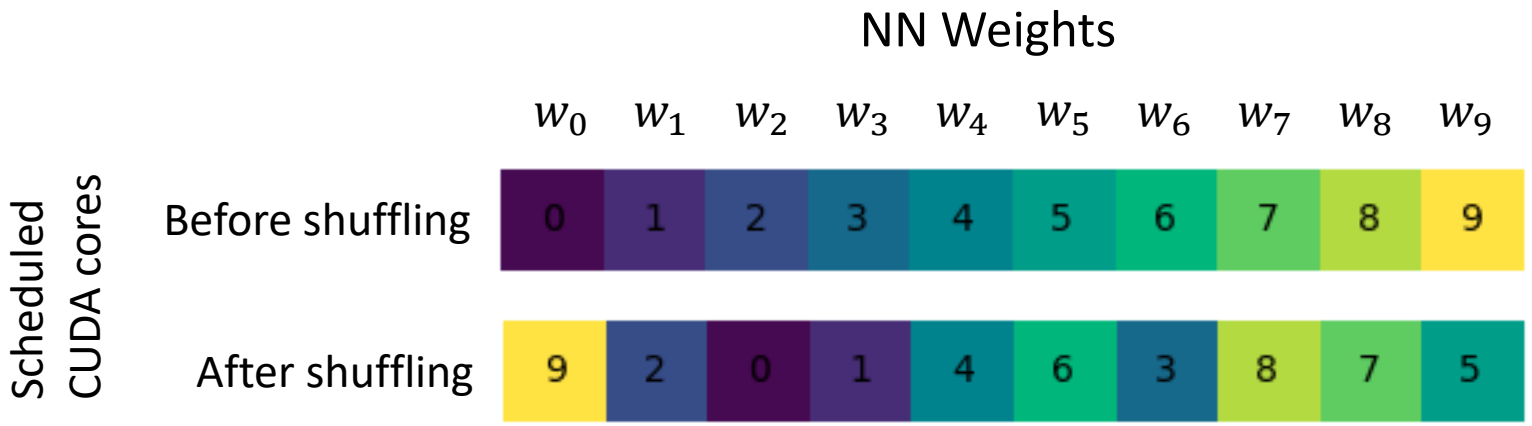
- Each CUDA core has a fixed **fault probability/rate (FR)**.
- A **fault profile** of a GPU determines the fault rate of each of its CUDA cores.
- At each instant in time, a CUDA core is sampled as either faulty or not-faulty by binomial sampling from the fault profile probability distribution



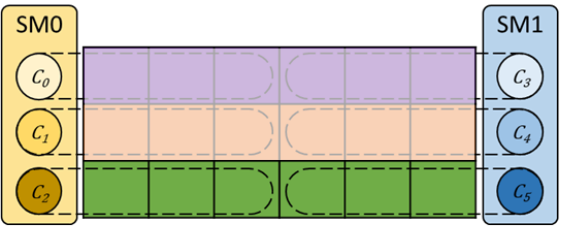
GA Problem Statement

- Given a hardware fault-profile,
  - what is the best shuffling order for the rows of the weight matrix so that the performance degradation is minimized

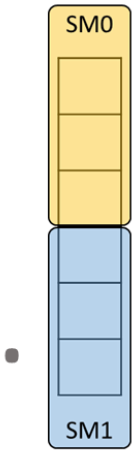
- GA method
  - Chromosome: represents the row shuffle order
  - Mutation/Crossover: different row shuffle orders
  - Fitness Function: top-1 accuracy over the test dataset



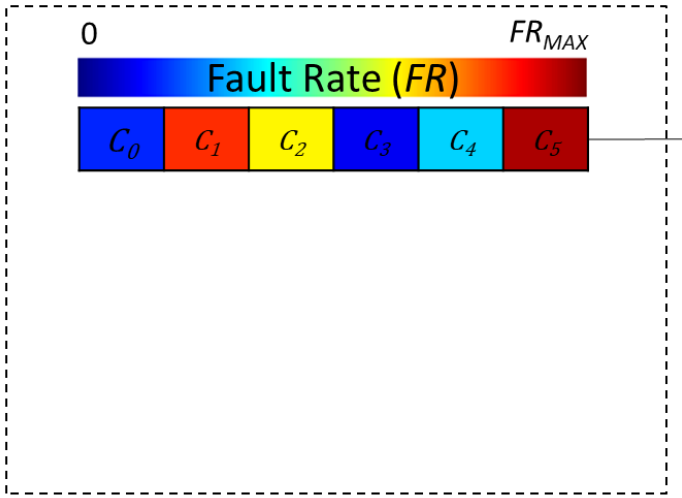
Putting it all together: *FAWS* + GA + Fault Injection



Original Weight Matrix
(Each row contains the weights for one neuron)

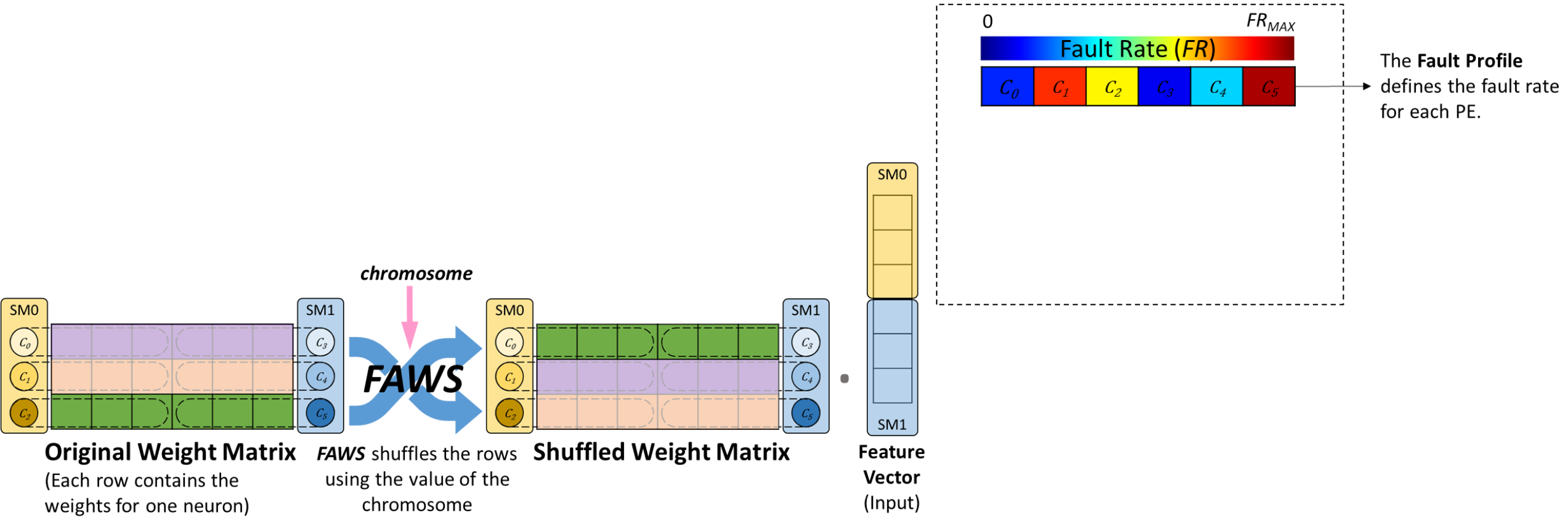


Feature Vector
(Input)

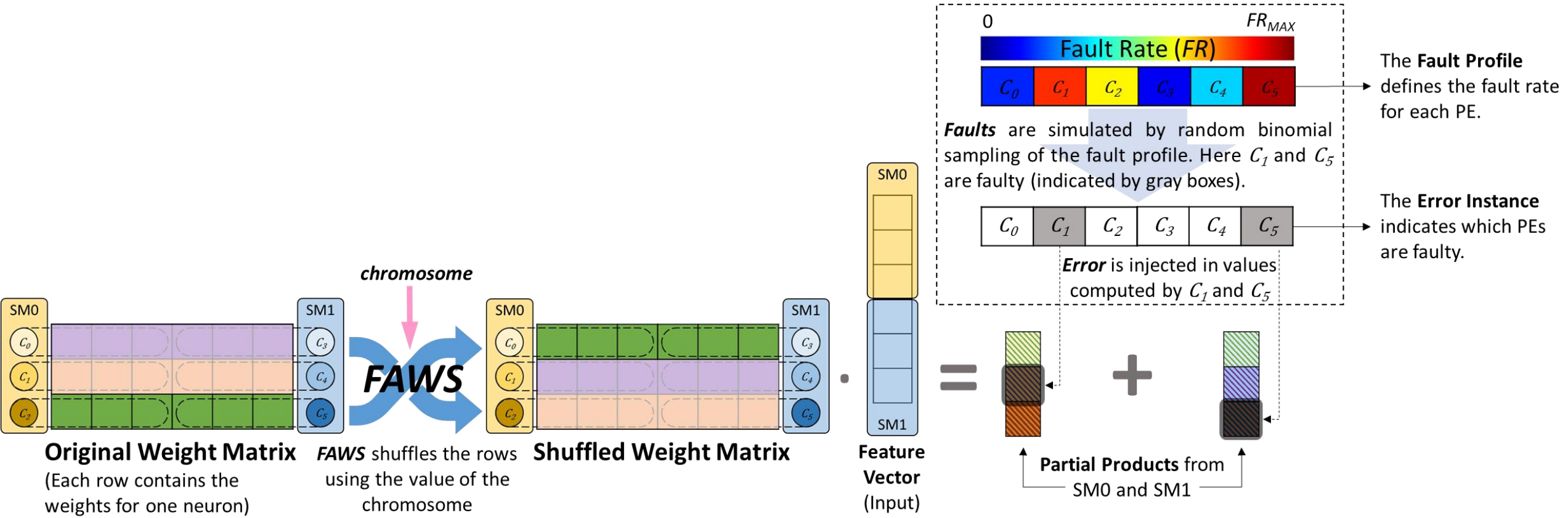


The **Fault Profile** defines the fault rate for each PE.

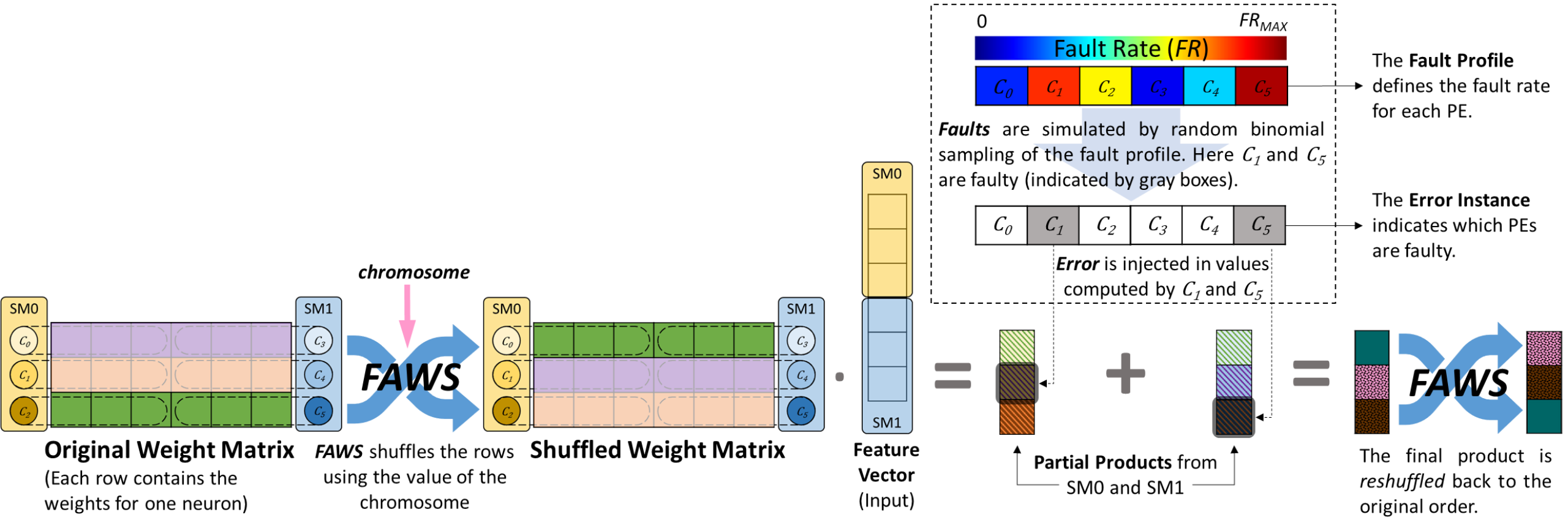
Putting it all together: *FAWS* + GA + Fault Injection



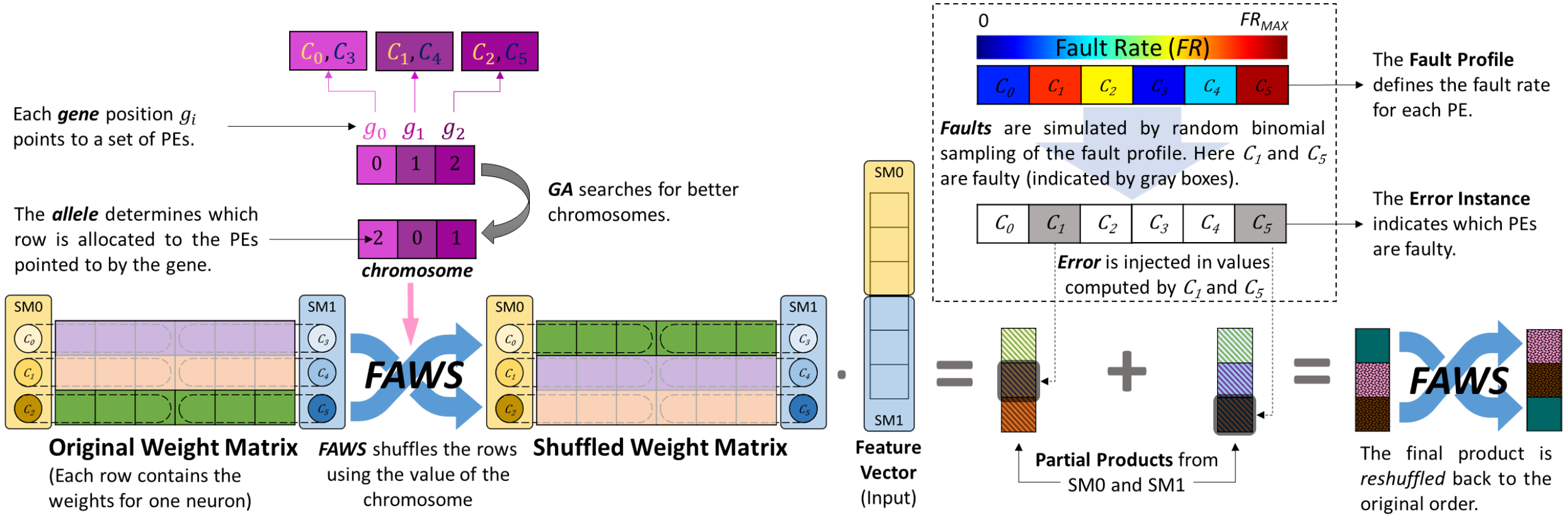
Putting it all together: *FAWS* + GA + Fault Injection



Putting it all together: *FAWS* + GA + Fault Injection



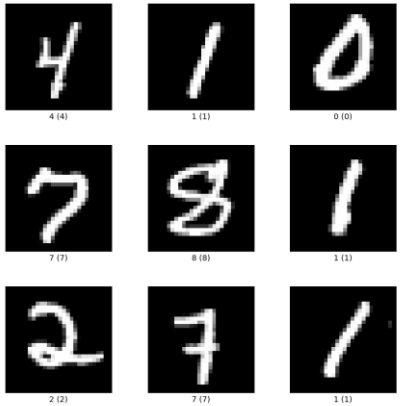
Putting it all together: *FAWS* + GA + Fault Injection



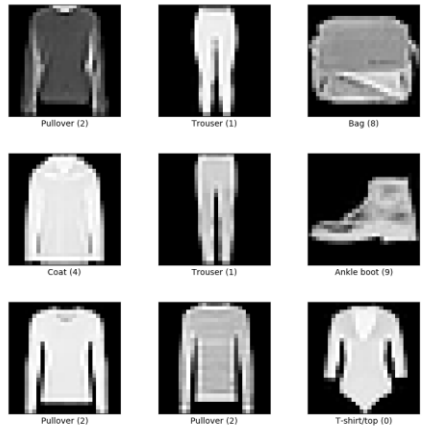
Experiments and Results

Experimental Setup

- DNN Models and Dataset
  - *mnist32-cnn* using MNIST Dataset
    - One conv layer followed by three fully-connected hidden layers
  - *fashion-cnn* using Fashion-MNIST Dataset
    - Two conv layers followed by one fully-connected hidden layer
- Assuming GPU has 20 SMs
  - Each SM has 32 CUDA Cores
- Fault profiles are randomly generated
- Max Fault Rates
  - 1E-3, 2E-3, 5E-3
  - 100E-3, 200E-3, 500E-3



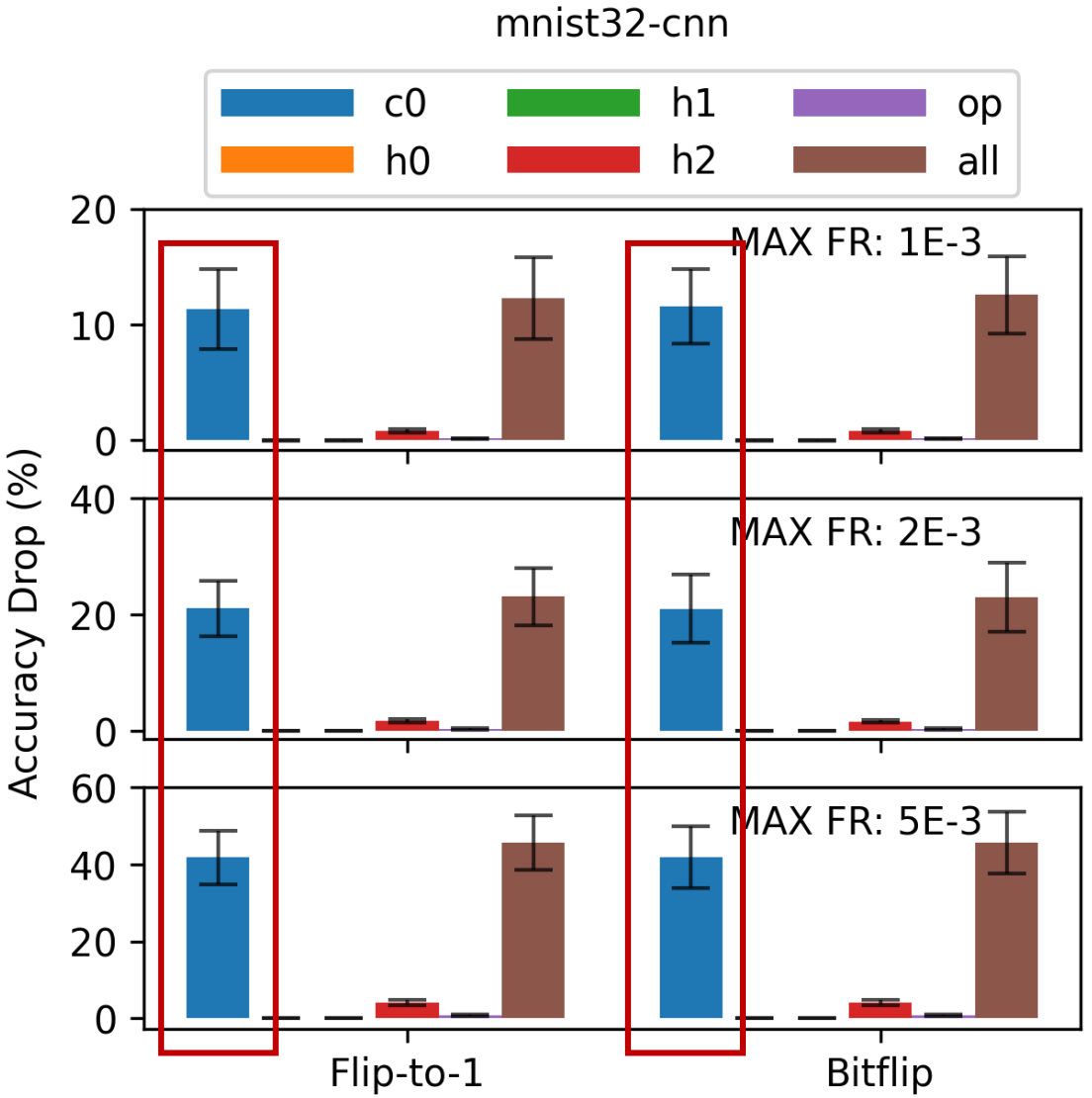
https://www.tensorflow.org/datasets/catalog/mnist



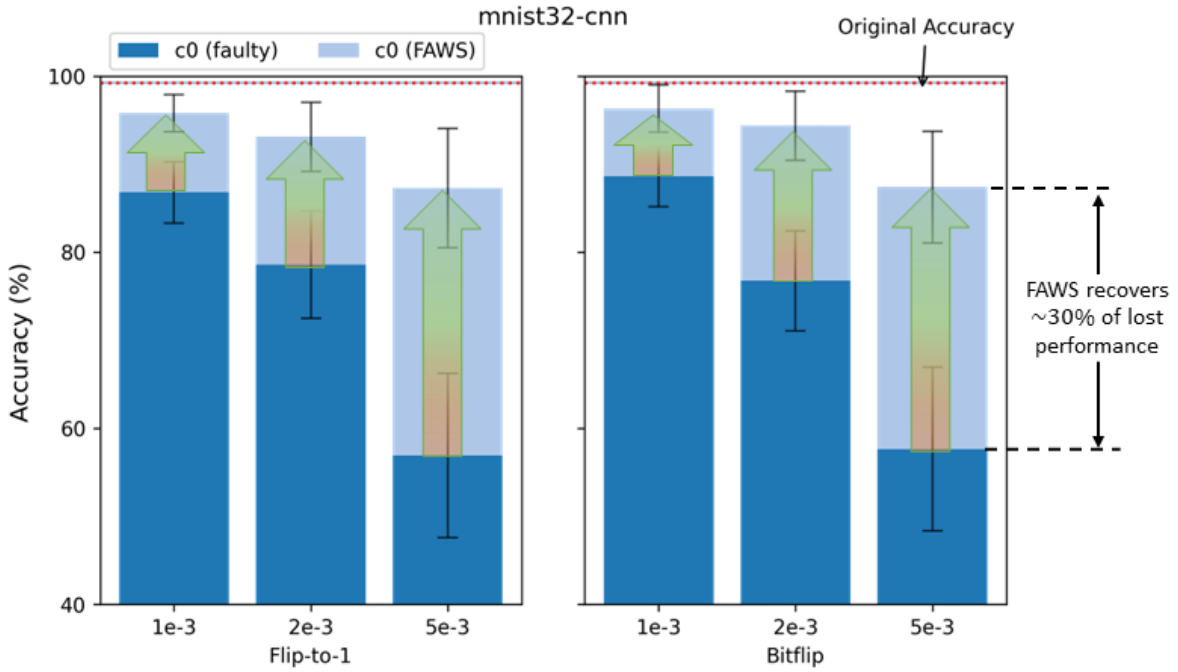
https://www.tensorflow.org/datasets/catalog/fashion_mnist

mnist32-cnn: Fault sensitivity (Bitwise Errors)

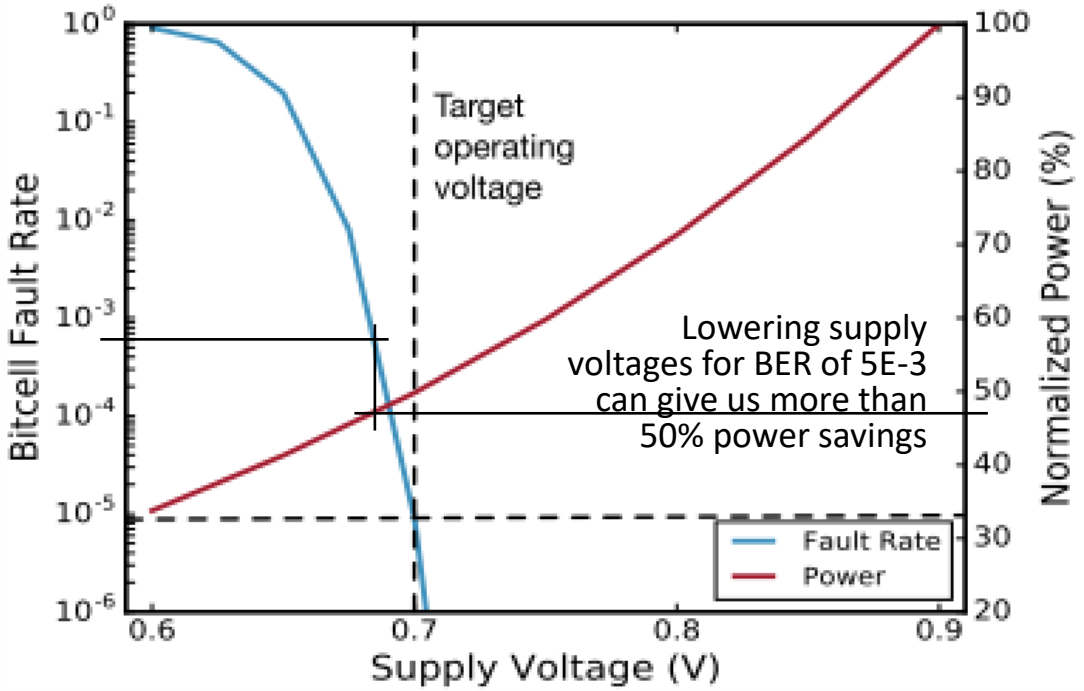
- Graceful degradation with increasing bit error rate
- conv layer *c0* is the most sensitive
  - Due to high reuse
- Can we recover performance using *HAS* on *c0*?



mnist32-cnn: Recovery with *FAWS* (Bitwise Errors)



Yes *FAWS* can!

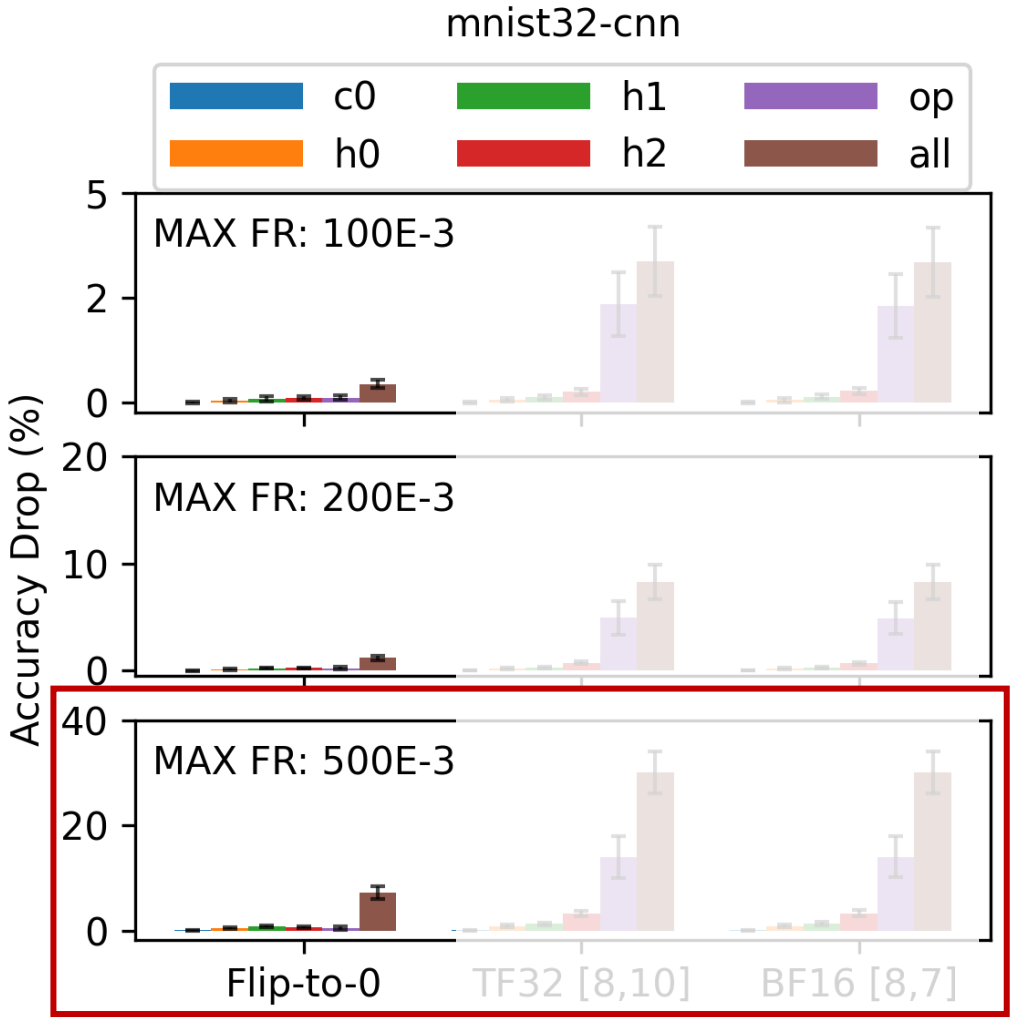


Reagen et al. - 2016 - Minerva Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators

- Upto 50% power savings!
- Recovery of up to 30% points!
- With almost no computation overhead!

mnist32-cnn: Fault sensitivity (Mantissa Errors)

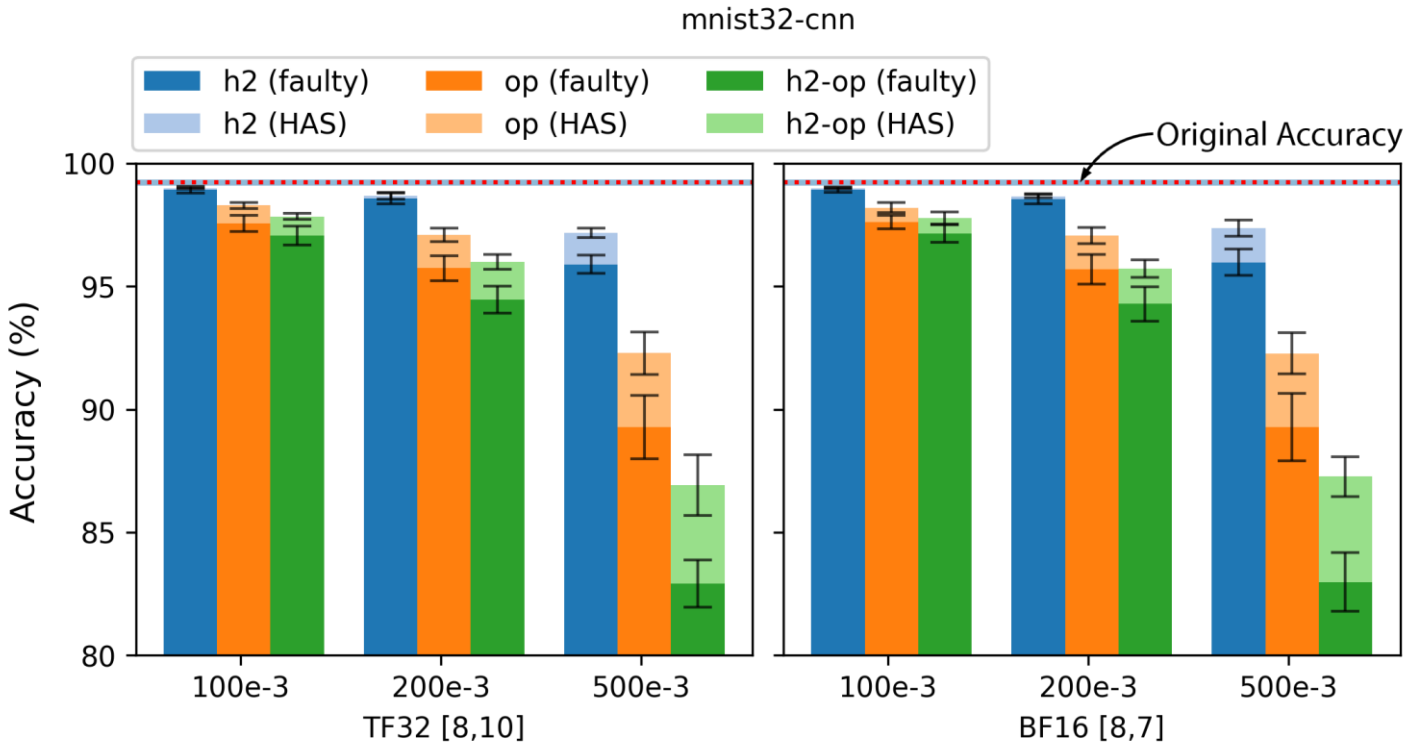
- Flip-to-zero is not that critical
  - Suppressed activations are mostly benign
- Precision can be reduced aggressively!
  - HAQ, RAPiD
- Output layer *op* is most sensitive to reduced precision
  - Softmax function requires high precision
- Degradation in each layer has a cumulative effect!
- Use *FAWS* on *h2* and *op* layers



mnist32-cnn: Recovery with *FAWS* (Mantissa Errors)

- Recovery of ~5% points
- Greater recovery in op layer

- In our simulation precision is reduced randomly
  - Harder optimization problem
- In reality
  - More deterministic and controlled mixed precision schemes
  - May result in better recovery



Conclusions and Future Directions

- Accelerators are faulty
- DNN can tolerate some computation inexactness
- We propose *FAWS* so that unimportant computations are scheduled to compromised hardware
  - Achieved by shuffling rows
  - Best shuffling order is found using GA
- *FAWS* can recover up to 30% points of performance
  - Corresponds to large power savings
- *FAWS* is hardware-agnostic black-box optimization process – general, simple and cheap to implement

- GA hyperparameters can be tuned for faster convergence
  - Different mutation rates/ no. of generations for different hardware/models
- Can we use faulty hardware for training?
- Can we optimize *FAWS* dynamically depending upon the input to the model?

REFERENCES

- Reagen, Brandon, et al. "**Minerva**: Enabling low-power, highly-accurate deep neural network accelerators." *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016.
- Reagen, Brandon, et al. "**Ares**: A framework for quantifying the resilience of deep neural networks." *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018.

- Venkataramani, Swagath, et al. "**AxNN**: Energy-efficient neuromorphic systems using approximate computing." *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2014.
- He, Xin, et al. "**AxTrain**: Hardware-oriented neural network training for approximate inference." *Proceedings of the International Symposium on Low Power Electronics and Design*. 2018.
- Zhang, Qian, et al. "**ApproxANN**: An approximate computing framework for artificial neural network." *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015.

- Wang, Kuan, et al. "**Haq**: Hardware-aware automated quantization with mixed precision." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
- Venkataramani, Swagath, et al. "**RaPiD**: AI accelerator for ultra-low precision training and inference." *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021.

Shaswot SHRESTHAMALI
https://www.acsl.ics.keio.ac.jp/
www.shaswot.com

Yuan HE
https://www.acsl.ics.keio.ac.jp/

Masaaki KONDO
https://www.acsl.ics.keio.ac.jp/

慶應義塾
Keio University

JST MIRAI 未来社会創造事業

Your questions/comments and feedback are most welcome