

The AI Hardware Project

Simulating Analog Weights for Low-Power AI (AIHWKIT)

Matthew Anderson, John Schroter

<https://github.com/Mircea-s-classes/ai-hardware-project-proposal-the-ai-hardware-group-project.git>

Digital vs. Analog

Digital AI (normal computers):

- Memory and compute are separate
- The computer moves data back and forth a lot
- That costs time and energy

Analog AI (in-memory idea)

- Memory and compute happen in the same place (in the memory array)
- Less data movement
- That can save a lot of energy
 - Attractive for low-power edge devices because it can reduce data movement, which is a big energy cost

Roles

- John Schroter - Analog Sim and Repo
 - AIHWKIT port, sweeps, repo/Colab
- Matt Anderson
 - Train CNN baseline, metrics, plots

What AIHWKIT Provides

AIHWKIT = a simulator for analog hardware effects

- Works with PyTorch
- Lets us run same model idea in digital and analog
- Lets us add realistic analog issues like noise and limited update precision

What AIHWKIT gives us:

- Analog layers (AnalogLinear)
- Analog training (AnalogSDG)
- Adjustable hardware knobs like noise

We didn't use real hardware. Instead, we simulated how analog hardware might behave

Our Experiment Setup

Dataset: MNIST digits (loaded from IDX files in GitHub data section)

The loaders parse the IDX headers and normalize images to [0,1]

Model:

- This is currently a simple 2-layer network (MLP), not a CNN → run tests quicker
- Flatten $28 \times 28 \rightarrow 784$
- Layer 1: $784 \rightarrow 256$
- ReLU
- Layer 2: $256 \rightarrow 10$

Training setup:

- 3 epochs, batch size 64, learning rate 0.01

What We Test - The Main Experiment

We tested analog hardware behavior by changing noise settings.

The 3 parameters we swept:

- Forward.out_noise = noise during forward pass (affects activations)
- Backward.out_noise = noise during backward pass (affects gradients)
- Desired_bl = how fine weight updates are (update precision)

Sweep Size:

- Forward noise: 0.0 → 1.0 in steps of 0.2 (6 values)
- Backward noise: 0.0 → 1.0 in steps of 0.2 (6 values)
- Desired_bl: {1,3,5,7} (4 values)

Total = $6 \times 6 \times 4 = 144$ runs, each trained with 3 epochs and logged to CSV

What we are finding: how much noise can we tolerate before accuracy drops too much?

Results so Far

What we saw:

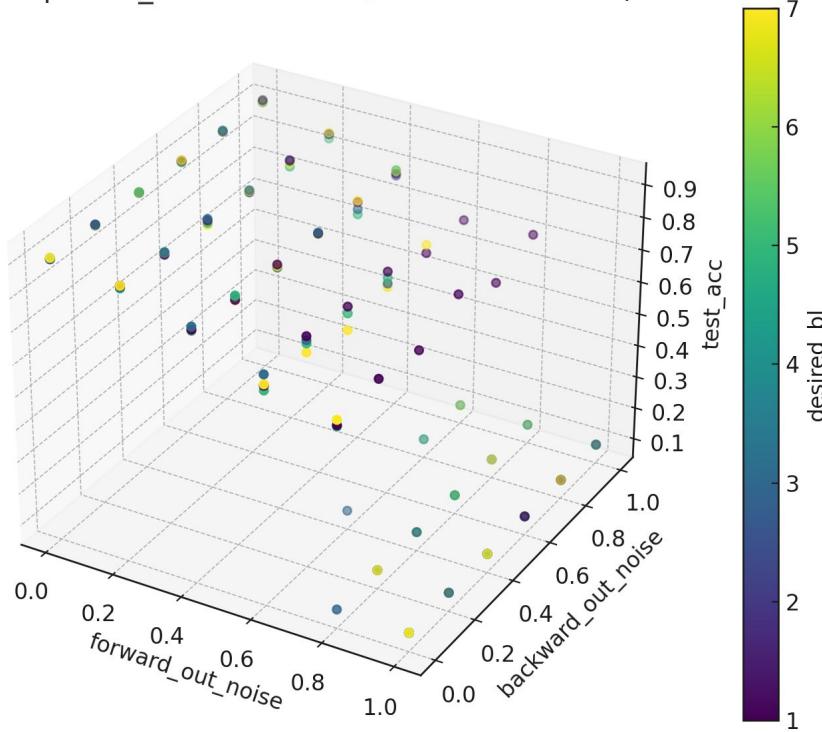
- Low noise = best accuracy; Higher noise = accuracy drops
- Some extreme noise settings cause training to break (NaN losses) and accuracy can fall near random guessing (around 10%)
- Forward noise is the strongest driver of accuracy drop
 - When forward noise rises, accuracy drops sharply

Best analog result in our sweep CSV

- Best test accuracy = 90.63% at
 - Forward noise = 0.0
 - Backward noise = 0.0
 - Desired_bl = 7

Results so Far

MNIST analog sweep: test_acc vs forward/backward noise (color=desired_bl)



What This Means

Digital vs. Analog

- Digital should look better on accuracy and speed in our experiment
 - Accuracy: because digital has no injected analog noise
 - Speed: because analog runs include simulator overhead
- Analog can be worth it in real hardware because the whole point is energy efficiency from less data movement
 - A very important aspect of analog

Key Message

- If analog hardware saves a lot of energy, then:
 - We might accept a small accuracy drop
 - As long as performance stays good enough for edge tasks