# A Controlled Comparison of FINN and Vitis AI for Small Neural Networks

**Samantha Ehrle**
Arizona State University
sehrle@asu.edu

**Ruthwik Reddy Sunketa**
Arizona State University
rsunketa@asu.edu

**Aman Arora**
Arizona State University
aman.kbm@asu.edu

## Abstract

Small neural networks are a common FPGA workload in low-latency applications such as high-energy-physics triggering, network-intrusion detection, and keyword spotting. Two widely used toolchains for deploying such models on AMD FPGAs are FINN and Vitis AI. FINN compiles each network into a custom spatial dataflow accelerator with weights held on-chip. Vitis AI runs networks on the DPU, a fixed overlay that streams weights from off-chip DRAM. Prior comparisons do not isolate the framework's own effect because they compare at mismatched precision or evaluate the dataflow design only in simulation. We compare FINN and Vitis AI on a resource-constrained Zynq UltraScale+ device (ZU3EG), holding precision (INT8) and operating point (batch-1) fixed, across a family of small CNN and MLP classifiers, and with both frameworks measured on the same hardware. Neither framework dominates. The overlay delivers fixed, model-independent resource use and throughput that varies only weakly with model size. The dataflow accelerator is sensitive to network structure, reaching higher throughput and lower energy per image on the smallest models of both families, but unable to fit larger networks at INT8 that the overlay deploys directly. We report performance, resource utilization, and programming effort, separating differences that are intrinsic to the two paradigms from those specific to a toolchain, device, or model.

## 1 Introduction

Many scientific and edge inference tasks run small neural networks under tight latency budgets, where a few microseconds matter. FPGAs are a natural target for this regime, and two toolchains from AMD dominate the deployment workflow. How they compare at the same precision, on the same hardware, has not been measured previously. We aim to bridge this gap through this work.

The two toolchains take opposite approaches. FINN compiles a network into a custom layer-pipelined dataflow accelerator [5], with weights held on-chip and per-layer parallelism set by a folding schedule. Network compilation and bitstream generation must be repeated for each network. Vitis AI, on the other hand, generates a fixed overlay (called the DPU) that time-multiplexes one engine across layers and streams weights from off-chip DRAM, so its resource footprint is set by the chosen engine size and is independent of the model.

Choosing between FINN and Vitis AI involves tradeoffs in performance, programming time, and deployment flexibility. In this work, we perform a qualitative and quantitative comparison of both deployment options for small neural networks, keeping all else (precision, batch size, model, FPGA device, etc.) constant, isolating the effect of overlay vs. dataflow. Precision stays at INT8 and the operating point at batch-1, both frameworks run on the same ZU3EG device, and we sweep a family of small CNN and MLP classifiers. We then attribute each difference to the paradigm, the toolchain, the device, or the model.

## 2 Related work

Three prior studies compare FINN and Vitis AI directly, and each confounds the framework with another variable. Hamanaka et al. [1] compare on a larger device using a customized FINN flow (FINN-S) that is not publicly available, at FINN INT4 against DPU INT8, and report a large FINN advantage. Machura et al. [2] compare on the same ZU3EG device we use, with both frameworks

on hardware, but again at FINN INT4 against DPU INT8; they report that FINN at INT8 did not fit their model on this device (over four times the available logic). Jaiswal et al. [3] compare on a third device at FINN 1W/5A against DPU INT8, with FINN evaluated only in simulation, and report a DPU advantage. Across all three, FINN runs below INT8 and, in one case, only in simulation, so the effect of the framework at matched INT8 on real hardware is unmeasured. Our choice of small networks makes the matched comparison possible, because the models fit FINN at INT8. While lower precisions are an attractive deployment choice that makes FINN favorable, especially because it may be difficult to find overlay generators that work with custom low precisions, it is not an option in many cases. Boutros et al. [4] survey the dataflow and overlay paradigms we compare.

# 3 Methodology

**Device and models.** We target the AUP-ZU3 board (Zynq UltraScale+ ZU3EG FPGA), with 70,560 LUT, 432 BRAM18K, and 360 DSP. The workload is five CNNs and five MLP MNIST classifiers, plus ResNet-8 on CIFAR-10. The five CNNs use two or three convolutional layers with channel widths from [8, 16] up to [32, 64, 128], spanning 1.4K to 94K parameters. The five MLPs are fully connected on the 784-dimensional input with one to two hidden layers, from 784-64-32-10 up to 784-512-256-10, spanning 53K to 536K parameters.

**Matched comparison protocol.** We compare at INT8 because the DPU supports only INT8, so it is the one precision at which both frameworks can run matched. Both frameworks run at batch-1 on the same ZU3EG board through compiled C orchestrators, with a 10-image warmup discarded. For Vitis AI, we use the DPUCZDX8G B512 configuration [6], built with Vivado 2024.1 and PetaLinux 2024.1 and run under VART 3.5.0. Only the five smallest DPU configurations (through B1600) fit the ZU3EG, but throughput is reportedly similar across these DPUs [1] for workloads of our size, so we pick the smallest one. Our FINN version is v0.10.1 and is used with Vivado 2022.2.

**Quantization schemes.** The two frameworks use their native, documented quantization flows. FINN uses Brevitas quantization-aware training with float per-tensor scales and unsigned activations. Vitis AI uses post-training quantization with power-of-two scales and signed fixed-point. Both frameworks are matched on bit-width (INT8), per-tensor granularity, and symmetry (integer zero maps to real zero, no zero-point offset). The accuracy gap is confined to the smallest 1.4K-parameter CNN, where post-training quantization trails quantization-aware training by about five points. On every other network the two frameworks reach nearly the same accuracy, within about a percentage point on the larger CNNs and a few tenths on the MLPs, so the gap is post-training-versus-trained sensitivity. Throughput, energy, and area are set primarily by the INT8 bit-width and the shared device rather than the scale calibration, so comparing them across the two flows stays controlled.

**Measurement.** We report latency as the time for a single inference, image to prediction. Power is measured with a FNIRSI FNB58 inline USB meter on the whole-board input rail at 100 Hz over the full duration of each run. To evaluate dynamic power, we subtract that design's bitstream-loaded idle power from the active power consumption while the inference is running. Because the meter reads the whole-board rail, the overlay's DRAM traffic is included by construction. The overlay fetches weights from DRAM on every inference, so an on-chip-only measurement would undercount its main energy cost. Resource numbers are taken from the post-route Vivado reports.

# 4 Results and Discussion

## 4.1 Throughput and Energy

Table 1 shows the results for the smallest and largest MLP and CNN models in our suite. On the smallest (1.4K-parameter) CNN at INT8 and batch-1, FINN reaches 10,738 FPS at 0.36 mJ/image against the DPU's 6,500 FPS at 0.70 mJ/image, a 1.65× throughput advantage and a 1.94× reduction in energy per image for the dataflow accelerator. The smallest (53K-parameter) MLP likewise favors FINN, at 10,338 against 6,547 FPS and 0.35 against 0.71 mJ/image, a 1.58× throughput advantage and 2.0× lower energy per image. The largest models reverse this. On the 94K-parameter CNN the DPU runs at 3,811 FPS while FINN does not fit at any folding tested, and on the 536K-parameter MLP the DPU's 3,337 FPS against FINN's best fitting 428 FPS is a 7.8× throughput advantage for the overlay, at 1.51 against 8.56 mJ/image.

Figure 1 shows the full folding sweep using FINN against the DPU across both families. FINN achieves higher throughput on the smaller models, but the DPU wins for larger models. The crossover sits at a different size in each family. FINN keeps its throughput lead up to much larger MLPs than CNNs, staying ahead through to the mid-size 235K-parameter MLP but only up to the smallest

Table 1: Performance of the smallest and largest CNN and MLP models. INT8, batch-1 (measured, compiled C runner on both sides). FINN rows are each model's highest-throughput fold that fits. Model names give approximate parameter counts (e.g. CNN (1.4K) is the 1,442-parameter CNN).

| Model (params) | Framework | Throughput (FPS) | Energy (mJ) | Accuracy (%) |
|---|---|---|---|---|
| CNN (1.4K) | FINN | 10,738 | 0.36 | 91.47 |
| CNN (1.4K) | DPU | 6,500 | 0.70 | 86.74 |
| MLP (53K) | FINN | 10,338 | 0.35 | 97.35 |
| MLP (53K) | DPU | 6,547 | 0.71 | 97.14 |
| CNN (94K) | FINN | — (no fit) | — | — |
| CNN (94K) | DPU | 3,811 | 1.29 | 99.15 |
| MLP (536K) | FINN | 428 | 8.56 | 98.14 |
| MLP (536K) | DPU | 3,337 | 1.51 | 97.85 |

1.4K-parameter CNN. The reason is folding: a convolutional layer reuses each weight across many spatial positions, so its multiply-accumulate count far exceeds its parameter count and reaching a target throughput exhausts the device's logic quickly, while a fully connected layer uses each weight once and folds much further before any resource runs out.

### 4.2 Latency

Throughput and latency results diverge. FINN's pipeline overlaps consecutive inferences to raise throughput at the cost of latency. The DPU instead processes one image at a time at a lower single-inference latency on both families: about 0.15 ms on the smallest CNN and MLP alike, against FINN's 0.23 ms on that CNN and 0.19 ms on that MLP. Across the full sweep the DPU's latency stays low and changes little with model size, from 0.15 to 0.30 ms, while FINN's latency is set by how each design is folded rather than by model size.

### 4.3 Resource Usage

The DPU's resource use is fixed and model-independent. The same bitstream uses 54.79% LUT, 33.33% BRAM, and 37.22% DSP for every model. Over a more than tenfold range of model size, its throughput varies only about twofold across the sweep and its energy per image from about 0.70 to 1.5 mJ. FINN's resource use, however, is dependent on the model. Its logic and memory use change with how much a model is folded and with the shape of its layers, so two models of similar size can land in very different places. FINN's INT8 footprint scales with the model until it approaches or exceeds the device; the largest CNN exceeds it outright, while the largest MLP fits only at its lowest throughput target. The 94K-parameter CNN does not fit at any folding, exceeding the BRAM budget at high folding (101.9%) and the logic budget at low folding (153.1% LUT). The overlay runs every size at the same fixed cost, including the 94K-parameter CNN that FINN cannot fit. Even on the larger models FINN can fit, it uses more of the device than the overlay's fixed cost, so the overlay is the lighter option on resources across the whole range, even at the smaller sizes where FINN takes the throughput lead.

### 4.4 Structure Sensitivity

FINN's resource cost depends on a network's depth and layer widths rather than its parameter count, unlike the overlay's fixed footprint. At a matched fold, the wider 20K-parameter CNN uses more LUTs (87.9%), while the deeper 24K-parameter CNN uses more memory instead (68.5% BRAM, 42.4% LUT). Such sensitivity appears within the same network as well: as the 1.4K-parameter CNN is pushed to higher throughput, its logic use rises and its memory use falls, so even a single model has no fixed binding resource. More broadly, an MLP's cost is dominated by weight storage (binding resource = BRAM), while a convolutional network's cost is dominated by compute (binding resource = LUTs). Parameter count alone therefore does not predict whether a model fits FINN, so the boundary that decides FINN-versus-overlay has to be read from a network's structure. Parameter count misleads on the DPU too, since its throughput tracks multiply-accumulate work per image rather than model size, which is why it runs faster on the 24K CNN than on the 20K one that does nearly twice the work.

### 4.5 Iso-area and Iso-performance Comparisons

Two deployment situations provide interesting insights. In a resource-constrained scenario, the question is which framework delivers more throughput within the same resource budget; this is the iso-area comparison. In a performance-critical scenario, the question reverses: which framework meets that throughput target using less of the device; this is the iso-performance comparison. For

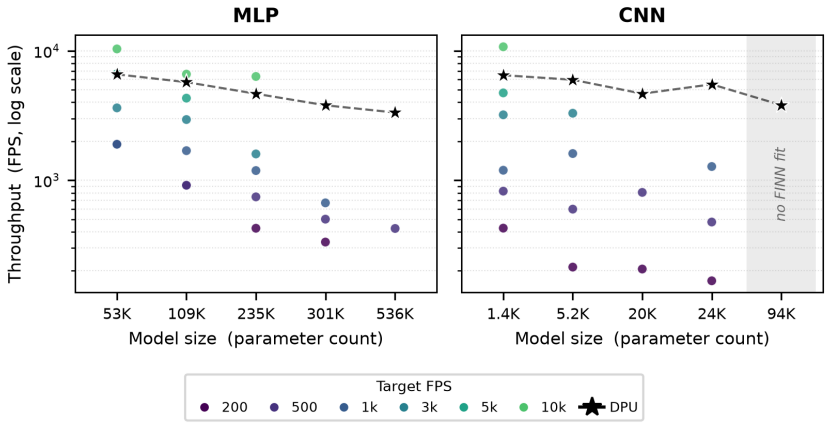


Figure 1: Throughput versus model size. Multiple FINN results (circles) per model are from the folding sweep; color is the requested throughput target, which sets the folding factor used by FINN; a higher target means less folding. The DPU (stars) is one fixed point per model.

iso-area, the smallest 1.4K-parameter CNN is the one model where FINN's logic use nearly matches the overlay's (FINN = 56.1% of available LUTs, DPU = 54.7%), and at that matched logic, FINN runs 1.65× faster (Table 1). For iso-performance, FINN's throughput advantage on the MLPs narrows as the models grow. The two come closest on the 235K-parameter MLP, where FINN runs at 6,336 FPS against the DPU's 4,660; below that size FINN is faster and above it the DPU is. The CNNs cross more sharply: FINN leads only on the 1.4K-parameter CNN, and the overlay is faster from the 5.2K-parameter CNN onward, with no CNN in our set sitting near equal throughput.

### 4.6 Programming Effort and Time

Deployment cost differs for the two frameworks. FINN's cost is hardware development, paid up front and once per model, since each network is compiled to a separate accelerator and bitstream [4]. A FINN bitstream took about 30 minutes to build on average, and 60 minutes or more for the largest models that fit. The DPU overlay bitstream is built only once and took about 15 minutes. After that a new model is added by recompiling it against the fixed overlay in software, which took under a second per model. That software path has its own maintenance cost, since the driver, runtime, and kernel are pinned to a single Vitis AI release and a mismatch among them can force extensive changes. A FINN accelerator, by contrast, runs on the device without such a runtime dependency.

### 4.7 ResNet-8 Case Study

ResNet-8 has a residual topology the default dataflow pipeline does not handle. FINN ships the fork-join transforms but the default flow never calls them, so a residual cannot be placed on the FPGA and the build fails when it tries to split the model between fabric and CPU. Invoking those transforms manually does compile ResNet-8 to a valid dataflow graph with all three residual joins on fabric [7], so the failure is one of deployability, not expressiveness. Even so, no folding of that graph fits the ZU3EG for FINN in our evaluation. In contrast, the overlay runs the same network directly at 2,639 FPS and 86.33% accuracy.

## 5 Conclusion

In this work, we compare two popular deployment methodologies for small neural networks: FINN's dataflow-based per-model accelerators and Vitis AI DPU overlay. At matched INT8 and batch-1, neither the dataflow accelerator nor the overlay dominates. FINN wins the smallest models of both families on throughput and energy, where folding enables the models to fit the device. Vitis AI wins on the larger models, deploying ones FINN cannot fit. The factors that determine the choice are invisible to parameter count: whether a network's shape lets a fast fold fit FINN, and how much multiply-accumulate work it gives the overlay's fixed engine. The overlay also has the lower single-shot latency, even where FINN wins throughput. Our results are limited by one constrained device, a small-model regime, and native quantization schemes (QAT versus PTQ) matched on bit-width and symmetry but not scale representation. Furthermore, sub-byte precision (not compared here) extends FINN's reach where the INT8-only DPU cannot.

## References


[1] F. Hamanaka, T. Odan, K. Kise, and T. V. Chu. An exploration of state-of-the-art automation frameworks for FPGA-based DNN acceleration. *IEEE Access*, 11, 2023.

[2] M. Machura, M. Danilowicz, and T. Kryjak. Embedded object detection with custom LittleNet, FINN and Vitis AI DCNN accelerators. *J. Low Power Electronics and Applications*, 12(2), 2022.

[3] M. Jaiswal, A. Sharma, and S. Saini. Hardware acceleration of Tiny YOLO deep neural networks for sign language recognition: a comprehensive performance analysis. *Integration, the VLSI Journal*, 100, 2025.

[4] A. Boutros, A. Arora, and V. Betz. Field-programmable gate array architecture for deep learning: survey and future directions. *arXiv preprint arXiv:2404.10076*, 2024.

[5] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers. FINN: a framework for fast, scalable binarized neural network inference. *FPGA*, 2017.

[6] AMD/Xilinx. *DPUCZDX8G for Zynq UltraScale+ MPSoCs Product Guide (PG338)*, v4.1, Jan. 2023.

[7] FINN-examples issue #67, residual-connection workaround. `https://github.com/Xilinx/finn-examples/issues/67`.