This ICCV Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Toward Low-SWaP Cognitive Agents: Neuromorphic Intelligence and FPGA-Based Deployments of Event Neural Networks

Trevor Bihl, Rajashree Majumder, Zhewei Wang, Avinash Karanth, Jundong Liu

School of Electrical Engineering and Computer Science
Ohio University
Athens, OH 45701

bihlt, rm618221, wangz1, karanth, liuj1@ohio.edu

Abstract

Traditional artificial intelligence (AI) solutions are often high-SWaP (Size, Weight, and Power), data hungry, and lack the efficiency, resilience, and autonomy exhibited by biological intelligence. As a consequence, autonomous and cognitive agents suffer from various technical bottlenecks precluding their widespread use. Neuromorphic computing offers the potential to develop learning machines that perceive, adapt, and learn continually.

In this paper, we survey the state-of-the-art in neuromorphic intelligence, including advances in spiking neural networks (SNNs), emerging brain-inspired hardware platforms, supporting software stacks, and representative application domains. Key to these advances are deployment pipelines for development, evaluation, and testing of solutions. Thus, in parallel, we further present a preliminary FPGA-based deployment pipeline for both artificial neural networks (ANNs) and SNNs. This approach not only provides a practical low-SWaP alternative for edge AI but also enables head-to-head benchmarking of neuromorphic solutions against conventional neural models on reconfigurable hardware. Together, these perspectives define a roadmap toward robust, low-power cognitive systems capable of real-world autonomy across embedded and constrained environments.

1. Introduction

Artificial intelligence (AI) solutions are the result of complex algorithms where abilities general grow with computational complexity and scale and size of the data explored [3]. With traditional von Neumann hardware, these increasing abilities come with increasing power consumption demands [15, 28]. For example, a training a large deep model can have megawatt scale power demands [33].

Such concerns and limitations are exacerbated in edge applications, where power, hardware, and software constraints critically drive capabilities. One key example is how complex and accurate computer vision systems might have high latency while requiring significant data storage and power needs. Such limitations possibly precluding edge application, e.g. smartphone or unmanned aerial vehicle (UAV) or limit their usage time. Traditional solutions include making more compact, but less accurate, algorithms for the edge, e.g. [12, 35].

Neuromorphic computing has emerged as a new computational paradigm inspired by the brain, aiming to overcome the limitations of traditional AI hardware. In contrast to von Neumann approaches, neuromorphic architectures implement spiking neural networks (SNNs) with asynchronous, event-driven signaling, plus tightly collocated memory (as synaptic weights), enabling continuous real-time operation with sparse spikes. This paradigm shift trades traditional von Neumann sequential computation for massive parallelism and memory-in-situ, much like biological neural tissue, with the potential of faster computation and dramatically lower power consumption [26].

Motivating applications span the edge, internet of things (IoT), and mobile domains: neuromorphic processors excel at always-on sensory processing and on-device learning, where energy or latency is critical. While symbolic AI approaches from the past are brittle for real-world perception, and deep learning's data-driven methods deliver high accuracy at high cost, neuromorphic computing offers a third path through brain-like SNNs which can potentially learn in real time through local plasticity and exploit sparse temporal patterns, potentially enabling general-purpose AI that is continuously adaptive yet far more efficient.

This paper explores the four pillars of neuromorphic research: hardware (including vision, auditory, olfactory and other event-based sensors, as well as spiking ASICs and in-memory neurosynaptic chips), theory (spiking neuron models, neural coding and learning rules like STDP), software (development frameworks and training methods for SNNs), and applications (edge/IoT sensing, robotics, etc.). Key issues in this domain remain, including developing and deploying algorithms,

Demonstration is further given on a novel deployment and evaluation of a deployment pipeline of neural networks on FPGA. To validate the pipeline, we designed and tested three neural networks of increasing complexity. These results lay strong foundation for future work in head-to-head benchmarking of conventional edge AI models and neuromorphic platforms.

2. Background

Neuromorphic computing draws on biological and historical inspirations with the term coined by Mead in 1990 to describe VLSI circuits that mimic neural systems [26]. This work heavily focused on analog silicon sensors and systems, exploiting transistor physics for adaptive, low-power sensing and processing applications. Central to these are spiking behaviors (event-driven and/or spiking neural) and synaptic plasticity, both inspired by the sparse and complex biological neuron processes. Biological neuron modeling developed simplifications that describe neuron communication processes, such as leaky integrate-and-fire or more complex Hodgkin–Huxley dynamics. In such systems, synapses store weights and adapt based on activity as seen in the spike-timing-dependent plasticity (STDP) model whereby a synapse's strength when pre- and post-synaptic spikes occur in close succession [16]. Neuromorphic computing often incorporates variants of STDP for on-chip learning [21]. However, in practice, both the hardware and software both use simplified models (e.g. LIF neurons with linear leaky dynamics) of complex biological neurons for efficiency and practicality [33].

The brief history of neuromorphic computing begins with the analog computing VLSI chips of the 1980s and extends to the current generation of both analog and digital methods, as well as recent advances in neuromorphic photonics. Early neuromorphic concepts employed transistor physics for neuron-like integration operations with a focus on sensor applications [31]. These sensors further focused on emulating biological sensing and includes silicon retina cameras for vision, silicon cochleae for hearing, and chemical noses [31]. These designs were notably power-efficient but sensitive to noise and fabrication variability. Beginning in the 1990s, developments in asynchronous address-event representation (AER) and STDP learning approaches were made to make more extensive computational spiking neural networks possible [21]. These advances, coupled with developments in spiking neural networks [18] made necessary algorithmic foundations for neuromorphic computing. Beginning in the 2000s, large investments began to be made in neuromorphic processors, e.g. TrueNorth and Loihi, with support for large numbers of simple spiking neurons with asynchronous, event-driven operation [8, 10, 26].

In addition to further hardware advances, recent innovations have been on software and algorithms. Notably, spiking neural network algorithmic libraries lag traditional neural network libraries in scope and applicability [20]. Further difficulties in training algorithms for spiking operations abound, largely due to the nondifferentiability of spiking operations, have led to mappings from traditional neural networks to spiking and surrogate-gradient methods to enable algorithmic development [21].

Despite these advances, further gains are possible as seen in the development of neuromorphic photonics methods which provide additional gains in computational speed, but are currently highly theoretical and not miniaturized [25]. Neuromorphic photonics combines neuromorphic computing principles (such as spiking neurons and parallel, event-driven processing) with light-based (photonics) hardware to achieve ultra-fast and energy-efficient computation. A general comparison of computing paradigms by computational speed and efficiency is seen in Fig 1.

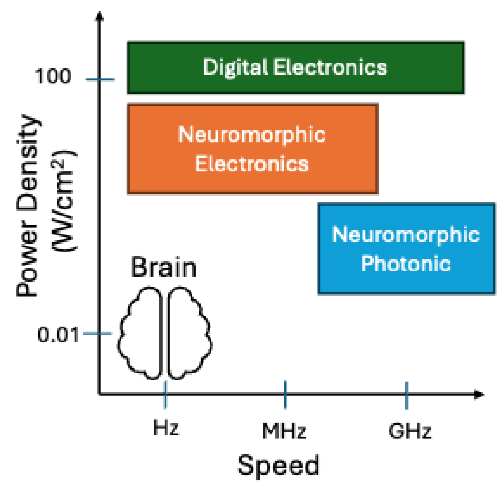


Figure 1. General comparison of computing paradigms by computational speed and efficiency, adapted from [11, 22].

3. Key Issues in Neuromorphic Computing

While neuromorphics offer promises of advantages in adaptability and energy efficiency, the field faces numerous challenges across hardware, theory, software, and applications. Below we detail the key issues in each area.

3.1. Hardware Challenges

Hardware challenges exist in availability, use, and scale when compared to traditional von Neumann processors.

Neuromorphic imaging systems have relatively coarse pixel arrays and limited color capability, resulting in lower visual fidelity and clarity than conventional high-definition cameras [36]. Achieving higher resolution in event sensors is costly and can introduce latency or data loss, and they remain sensitive to noise (especially in low light) and it is on-trivial to integrate their data into existing, frame-based vision systems [34].

Similarly, specialized neuromorphic chips currently operate at a much smaller scale than mainstream computing hardware. Typical designs, like Intel's Loihi-2 or IBM's TrueNorth, contain less than one million spiking neurons per chip, while a significant achievement the scale is still modest compared to modern deep neural networks and preclude their use for many applications/demonstrations [30]. Because of size and connectivity limits, today's neuromorphic hardware cannot yet match the complexity or capacity of GPU-accelerated neural networks [3]. Moreover, most neuromorphic chips are research prototypes or niche products. When consideration neuromorphic photonics, the systems are even more prototypical in nature [11, 25]. In total, his means neuromorphic hardware is not widely available or standardized.

3.2. Theoretical and Algorithmic Challenges

A fundamental challenge in neuromorphic computing is the lack of consensus on how best to model neurons and encode information. There are many spiking neuron models (leaky integrate-and-fire, Hodgkin–Huxley, Izhikevich, etc.), each balancing biological realism against computational simplicity. It remains unclear which level of detail is necessary or optimal for useful computation – simpler models are easier to implement in hardware, but more complex models might capture functionalities that simpler ones miss. Very limited work exists in benchmarking and comparing these models [19]. Likewise, neural coding in spiking systems is not fully understood: brains may encode information in precise spike timings, spike rates, or combinations of both. SNNs can leverage temporal codes (exact spike sequences and latencies) or rate-based codes (average firing rates over time), but there is no general theory for which coding scheme to use for a given task.

Perhaps the greatest theoretical hurdle is learning in spiking networks. Traditional deep learning relies on gradient backpropagation, which is hard to directly apply to spiking neurons because spikes are non-differentiable events. While STDP enables unsupervised or self-organizing learning in SNNs, it alone is not sufficient for training deep SNNs to perform complex, large-scale tasks. Developing effective training algorithms for SNNs is an active research area, but no approach has become a clear standard. Methods can be grouped into a few categories: (1) ANN-to-SNN conversion, where a standard artificial neural network is first trained with backpropagation and then converted to a spiking equivalent; (2) direct training with surrogate gradients, where one defines a differentiable approximation of spike activation to allow backpropagation in SNNs; (3) reward-based or unsupervised local learning (including STDP variants, spike-based reinforcement learning, etc.) [23]; and (4) generative models for combined data and algorithm generation [2]. But, each approach has drawbacks. For example, converting a deep ANN to an SNN often results in some loss of accuracy [5] and surrogate gradient training is computationally expensive when compared to standard ANN training [23].

3.3. Software and Ecosystem Challenges

Because neuromorphic computing draws on multiple disciplines (neuroscience, electrical engineering, computer science, etc.), there is further a steep learning curve for new practitioners [5]. This general unfamiliarity means fewer developers contributing neuromorphic code or applications, and less shared repository of best practices. As a result, the developer community and support resources for neuromorphic computing are small compared to the explosive growth in the deep learning community. This issue is self-reinforcing: without clear success stories or widely available hardware, few people learn neuromorphic engineering; but without a critical mass of developers, it's hard to create the success stories or software tools that would grow the field.

The software and hardware ecosystem for neuromorphic computing is far less developed than that for standard AI models (including deep learning). There is a shortage of well-supported development frameworks, programming languages, and tools tailored to spiking neural networks and neuromorphic hardware [20]. In many cases, researchers still use conventional AI libraries (e.g. PyTorch or TensorFlow) with custom modifications to simulate SNNs, or they rely on specialized research simulators (Brian, Nengo, SpiNNaker's API, etc.) that are not widely known outside the neuromorphic community. Beyond these issues, neuromorphic algorithms should not just leverage concepts from traditional AI algorithms but fully embrace the data and structure unique to these systems as seen in the Spiking ResNet [32]

Notably, most algorithms for neuromorphic systems are still implemented on von Neumann machines using software not originally designed for asynchronous spiking computation and then simulated or ported to neuromorphic devices [5]. The pipeline to developing spiking neural algorithms is not seamless and can lead to loss in accuracy or efficiency [5]. Critically, while challenge problems exist, there are few established common benchmarks, datasets, and challenge problems that could drive progress in the way benchmarks like ImageNet did for deep learning [5].

Additionally, neuromorphic vision sensors (e.g., dynamic vision sensors) have begun to see limited commercialization, other bio-inspired sensory modalities such as silicon cochleas and electronic noses remain largely unavailable as commercial products. Similarly, extending these concepts to other sensing modalities (multispectral or radio frequency) is uncommon. When sensors for other modalities are explored, these systems are often confined to academic prototypes or laboratory-scale implementations, with limited accessibility, inconsistent interfaces, and little to no industry-standard packaging or supply chains [14, 17, 27]. While their development remains a subject of active research, particularly in the context of event-based auditory and olfactory sensing, their commercial readiness is currently lacking.

3.4. 3.4. Application and Adoption Challenges

To date, neuromorphic computing has not found a definitive, must-have application that drives its widespread adoption. In deep learning, vision tasks like image recognition served as killer apps that demonstrated clear superiority over previous methods and spurred massive investment [9]. Neuromorphic systems, by contrast, have seen only limited real-world deployment and current applications are mostly experimental or in niches, such as foosball [6]. There are many potential applications where neuromorphic solutions show advantages over tradition methods (autonomous vehicles, robotics, smart sensors, graph search, constraint satisfaction, controllers, etc.) [9].

However, none have yet proven compelling enough to pull neuromorphic technology into the mainstream. One reason is that conventional AI is already performing sufficient in many of these domains; another is that neuromorphic hardware and software aren't mature enough for turnkey solutions. The absence of a killer app goes hand-in-hand with the lack of benchmarks: without a high-profile success to point to, neuromorphic computing remains a solution looking for a problem. However, the key killer app might not be in massive supercomputing, but perhaps in ultra-low-power edge computing.

A key issue in all these challenges is that any neuromorphic solution must compete with rapidly improving conventional AI hardware. which have enormous industry momentum and benefit from decades of software optimization. This dominance means that for many applications, it's easier and less risky to use a conventional AI approach than to attempt a neuromorphic one. Neuromorphic computing must therefore find scenarios where it offers a dramatic advantage that outweighs the maturity gap.

4. Future of Neuromorphic Computing

The four challenge areas (hardware, algorithms, software ecosystem, and application adoption) are deeply interdependent. Progress will come from coordinated efforts that cut across these areas.

An important enabler to future neuromorphic maturity is creating teams or "collaboratives" focused on grand challenges. This collaborative ethos will accelerate innovation and prevent siloed development. As neuromorphic computing inherently spans neuroscience, electrical engineering, computer science, and materials physics, cross-disciplinary collaboration is likely key to success. Taken to a logical extreme, this includes creating multi-user neuromorphic fabrication and testing facilities [24].

4.1. Open Standards and Interoperability

Establishing open standards is foundational to unifying the community and accelerating adoption. While this is occurring naturally, the community should converge on standards for neuromorphic programming interfaces, model description, and data formats to enable cumulative progress. By exploring interoperability constructs, e.g. [7], we can develop interfaces and deployments across various architectures and systems. For instance, a common programming API would allow researchers to run the same code on an FPGA testbed or a custom chip, fostering comparability. Additionally, standardizing benchmarks is equally important and we need to define and publish benchmark tasks that cover sensor processing, control, and sequential learning tasks relevant to neuromorphic strengths [21].

These basic and initial steps will provide neutral grounds to measure improvement across hardware and algorithms. From this, we should see standardization in more specialized areas, such as a standard neuron model library (so hardware can advertise support for a set of standard neuron types), and maybe standard communication protocols for neuromorphic devices and extensions/integration to robotics (through ROS) and other applications. Open standards will also ensure that photonic neuromorphic systems, when they emerge, can interface with electronic ones.

4.2. Shared Testbeds and Infrastructure

Just as the early internet benefited from shared testbeds, neuromorphic computing will benefit from open-access platforms where researchers can evaluate ideas at scale. The mention of the DeepSouth large-scale FPGA system to be publicly available in 2025 is a step in this direction [13]. Over the next few years, we should expand such infrastructure: for example, international neuromorphic research labs could host neuro-cloud services (Intel's Loihi cloud is an early example, allowing researchers remote access to Loihi boards). Eventually, researchers could run a job on thousands of neuromorphic cores distributed worldwide. These testbeds would not only accelerate research (no need for every lab to build their own hardware) but also help develop standards and software in a collaborative way. In addition, shared sensorimotor testbeds, e.g. a fleet of neuromorphic-enabled robots available for experiments, could unify application development efforts similar to how early internet enabled controls laboratories improved controller modelling through system access [1]. Cross-cutting projects might include a neuromorphic benchmarking lab that continuously evaluates new hardware and algorithms on a common suite of tasks, publishing results openly. This can guide the field by identifying what works best and where the bottlenecks are.

4.3. Education and Outreach

Developing a skilled workforce and informed user base is a necessity. In the short term, we need to better integrate neuromorphic concepts into university curricula for computer science, electrical engineering, and neuroscience. This can include new courses on spiking neural networks, hands-on labs with neuromorphic kits (possibly FPGA-based), and interdisciplinary programs. As one strategy, existing Edge AI and embedded ML courses could incorporate neuromorphic approaches, exposing the next generation of engineers to event-driven thinking [21]. Dedicated cross-disciplinary workshops, hackathons, and competitions targeting students and professionals can spark interest and facilitate creating thousands of developers proficient in neuromorphic tools, reducing one barrier to adoption.

As seen in the annual Global Game Jam [29], such activities not only train individuals but also generate publicity and enthusiasm. Such community engagement can be a key to driving wider developer interest [21], and must be an ongoing effort. Long-term, the goal is that neuromorphic computing becomes an integral part of the broader AI/computer engineering education, just as deep learning is today; ensuring a continuous pipeline of talent and a community that can maintain and grow the ecosystem.

4.4. Interdisciplinary Collaboration

Since neuromorphic computing inherently sits at the crossroads of multiple fields, achieving goals of adoption and use will require unprecedented cross-disciplinary e.g., [4, 24], which attributes progress to diverse hardware platforms and conceptual approaches. Practically, this means forming teams and consortia that bring together experts from different domains. For example, developing a future photonic neuromorphic application might involve photonics experts, neuromorphic algorithm designers, and FPGA engineers (for intermediate prototypes) working in tandem. Funding bodies should encourage this by funding larger center-scale projects with multiple specializations under one roof. Conferences and journals should continue to welcome interdisciplinary work, a neural algorithm paper should ideally be discussed by hardware folks and vice versa. Creating common language and metrics across fields is part of this enabler; for instance, neuroscientists providing insights on brain computations could inform new computational paradigms, while hardware constraints can feedback into theoretical models of what's feasible.

4.5. Policy, Funding, and Commercial Roadmapping

While not explicitly focused, it's worth noting that cross-cutting support also comes from policy and funding initiatives. Countries and regions should consider neuromorphic computing a strategic technology for the next decade and align research priorities so they have sustained support. Having a unified agenda can help make the case for such programs by showing a clear vision. Additionally, open collaboration between academia and industry in pre-competitive research can de-risk the technology. While academic research can be focused on research objectives, commercialization needs roadmaps and a realistic goal of when neuromorphic chips will reach certain capacities or cost points.

In summary, the next decade for neuromorphic computing is envisioned as a coordinated advance on multiple fronts. Early successes in hardware and algorithms will feed into better software, which will enable broader applications, which in turn justify further hardware investment. FPGAs can play an instrumental role throughout as both a development accelerator and a bridge to deployment, ensuring progress need not stall due to hardware availability [13]. By explicitly including FPGAs and photonic technologies in our agenda, we keep the field flexible and open to innovation from many angles. The integrated roadmap calls for community cohesion: through standards, shared platforms, education, and collaboration, the neuromorphic field can replicate and learn from the success of the GPU and AI ecosystem while avoiding some of its pitfalls (such as fragmentation and closed development).

5. Pipeline to Deploy Neural Networks onto FPGA

Deploying neural networks, both ANNs and SNNs, onto an FPGA requires several key steps that bridge the gap between high-level machine learning models and efficient hardware implementations. In this work, we develop a end-to-end pipeline for training and deploying event-based neural networks on FPGA platforms. As shown in Fig 2, the proposed pipeline consists of six stages: (1) network design and training on a PC; (2) weight extraction and quantization; (3) automatic conversion of trained models into Verilog HDL; (4) hardware integration and synthesis using Xilinx Vivado; (5) bitstream generation; and (6) deployment and testing via Vitis.

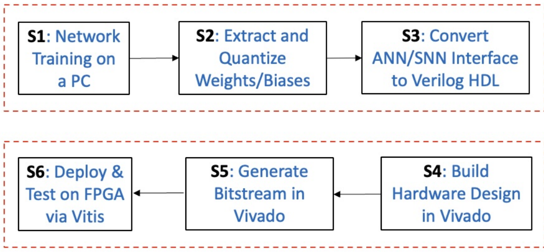


Figure 2. Six-stage pipeline to Deploy Neural Networks onto FPGA cards. **S1** stands for *Stage 1*.

5.1. Method: Individual Steps

The first step of our pipeline (**S1**) is to train a neural network (either an ANN or SNN) using a high-level framework on a local PC. The network is described using Python code and trained with data. Once the model is trained and validated, it is typically exported to an interoperable format such as ONNX. In this work, however, we export the trained weights and biases in *Memory Initialization File* (.mif) format. Fig. 3 shows the FPGA card used in this work, which is a Zybo Z7 (Zynq-7000) ARM/FPGA SoC development board.

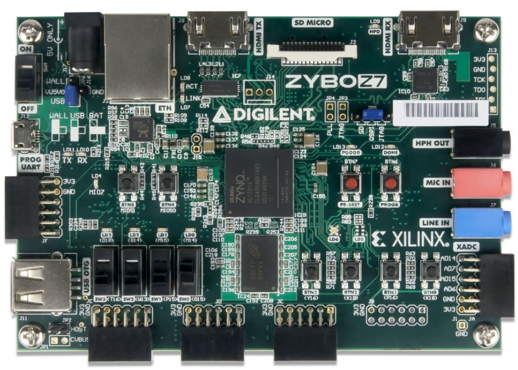


Figure 3. The FPGA Zybo Z7 card used in our work.

The next step (**S2**) is model weight extraction and quantization. Since FPGAs have limited resources and typically operate more efficiently with fixed-point arithmetic, it is common practice to convert the model's floating-point parameters to fixed-point or other lower-precision formats. Quantization-aware training can be employed during the model development phase to help preserve accuracy after quantization. In this step, the weights and biases are extracted and quantized for transfer to the FPGA.

In Step 3 (**S3**), the trained ANN or SNN models are converted into a hardware description language (HDL) for FPGA implementation. This conversion can be carried out either manually or through automated tools. For automated workflows, High-Level Synthesis (HLS) tools enable developers to express the network's inference logic in C/C++ or SystemC, which is then automatically translated into HDL. In our work, we use Verilog as the target HDL. We adopt a manual Verilog-based approach for simple networks and leverage automated toolchains for deploying larger, more complex models. Fig. 4 shows a segment of Verilog code that describes the structure of one of our networks.

```
reg signed [31:0] h [0:HIDDEN_SIZE-1];
reg signed [31:0] acc_h [0:HIDDEN_SIZE-1];
reg signed [31:0] acc_out;
integer i;

reg signed [31:0] mul0 [0:HIDDEN_SIZE-1], mul1 [0:HIDDEN_SIZE-1];

always @(posedge clk) begin
  for (i = 0; i < HIDDEN_SIZE; i = i + 1) begin
    mul0[i] <= $signed(in1) * w_hidden_0[i];
    mul1[i] <= $signed(in2) * w_hidden_1[i];
  end
end

always @(posedge clk) begin
    for (i = 0; i < HIDDEN_SIZE; i = i + 1) begin
        acc_h[i] <= mul0[i] + mul1[i] + b_hidden[i];
        h[i] = (acc_h[i] > 0) ? acc_h[i] : 32'd0;  // shift down to prevent overflow
        $display("h[%0d] = %d", i, h[i]);
    end
```

Figure 4. Sample Verilog code illustrating a portion of a network description.

The network designed in Step 1 and its hardware-translated version from Step 3 are synthesized, placed, and routed onto the FPGA in Step 4 (**S4**) using Xilinx Vivado. Fig. 5 shows a block diagram of the overall FPGA setup. The zoomed-in "RTL" block represents our neuron module, named XOR_ann_0. This neuron has two inputs and one output and was specifically and manually instantiated in the block diagram. Its structure is defined by the Verilog code generated during Step 3.

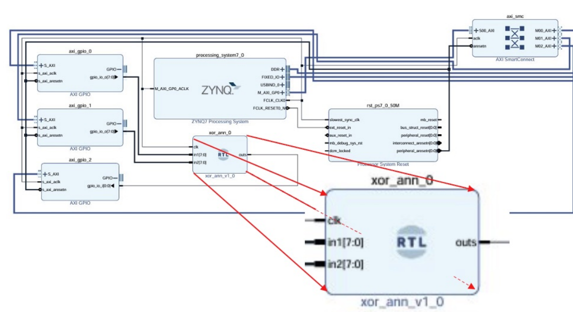


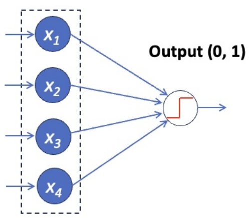
Figure 5. Illustration of a block diagram generated using Vivado.

For this simple neuron, input signals are delivered via *General-Purpose Input/Output* (GPIO) blocks, as specified in the design. For larger input data (e.g., images), input will instead be provided through a *Direct Memory Access* (DMA) block. The entire process culminates in the generation of a bitstream in Step 5 of our pipeline (**S5**), which can be loaded onto the FPGA for deployment.

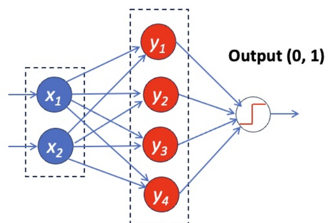
After deployment, the network should be tested with representative data to verify both accuracy and performance, which is the content of our step 6 (**S6**). Further optimizations may be necessary, such as refining quantization parameters or adjusting pipeline stages, to meet application requirements.

6. Experiments and Results

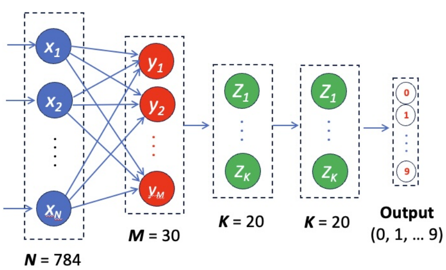
To evaluate the effectiveness of our FPGA deployment pipeline, we conducted experiments using three neural network designs of increasing complexity, as shown in Fig. 6. The first experiment featured a single-layer network (Fig. 6(a)), serving as a baseline to verify correct logic synthesis and hardware integration. The second experiment implemented a two-layer network (Fig. 6(b)) to solve the classic XOR problem. Finally, we deployed a four-layer multilayer perceptron (MLP) for image classification (Fig. 6(c)), which required higher data throughput and a more complex architecture. This final model leveraged DMA to efficiently stream image data to the hardware.



(a) Single layer network.



(b) Two-layer XOR network.



(c) Multilayer perceptron (MLP) for MNIST classification.

Figure 6. Three neural networks that have been trained and tested in this work.

Results The single-layer network has four possible inputs. We first trained it on a PC, and then deployed the weights into the FPGA card through the proposed pipeline. Testing is through a C program to send the inputs to the network on the card and retrieve the output, which turned out to be 100% accurate. These results confirm that our basic software-to-hardware pipeline is functioning correctly.

The training and testing of the two-layer XOR network were conducted in the same way. The results also achieved 100% accuracy, which provides additional evidence of the pipeline's effectiveness. In particular, it show the quantization process preserves the model's performance. To further demonstrate real-time execution, we present a snapshot of the hardware output (Fig. 7), showing input values, prediction results, and the system clock status.

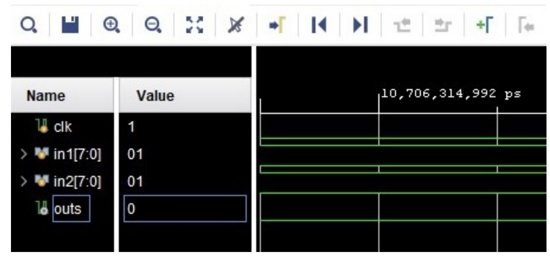


Figure 7. Snapshot of real-time network inference of the two-layer XOR network.

The third network is a multilayer perceptron (MLP) for MNIST image classification. As shown in Fig. 6, each input has 784 pixels (28×28). The model includes three hidden layers and a hardmax output layer. It was first trained in Python on a PC with about 98% accuracy. After deployment to the FPGA, we tested it with a small set of MNIST samples, using a DMA block to stream the input images. The model achieved reasonable accuracy, confirming that the full pipeline, from quantization to hardware integration and DMA transfer, is working properly.

7. Conclusions

In this paper, we reviewed recent progress in neuromorphic intelligence, including SNNs, brain-inspired hardware, supporting software, and key applications. Alongside, we introduced an FPGA-based deployment pipeline for neural networks, offering a low-SWaP solution for edge AI. We validated this pipeline by deploying and testing three neural networks of increasing complexity. This work provides a foundation for future comparisons between conventional edge AI models and neuromorphic platforms, moving toward energy-efficient, autonomous systems.

References

[1] T. Adami, R. Irwin, and J. Strahler. Flexlab: Internet based controls research. *IFAC Proceedings Volumes*, 34(9):471–473, 2001. 5

[2] A. Baietto and T. Bihl. Generative data for neuromorphic computing. In *Hawaii International Conference on System Sciences*, 2025. 3

[3] T. Bihl, P. Farr, G. Di Caterina, P. Kirkland, A. Vicente Sola, D. Manna, and K. Combs. Exploring spiking neural networks (snn) for low size, weight, and power (swap) benefits. In *Hawaii International Conference on System Sciences*, 2024. 1, 3

[4] D. Brunner, B. Shastri, M. Qadasi, H. Ballani, S. Barbay, S. Biasi, and G. Brennan. Roadmap on neuromorphic photonics. `https://arxiv.org/abs/2501.07917`, 2025. arXiv preprint arXiv:2501.07917. 5

[5] R. D. Caballar and C. Stryker. What is neuromorphic computing? https://www.ibm.com/think/topics/neuromorphic-computing, 2024. Accessed June 27, 2024. 3

[6] G. Cohen. Gooaall!!!: Why we built a neuromorphic robot to play foosball. *IEEE Spectrum*, 59(3):44–50, 2022. 4

[7] K. Combs, T. Bihl, and T. Pennington. Interoperability for autonomy. In *Hawaii International Conference on System Sciences*, 2024. 4

[8] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. Choday, and Y. Liao. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. 2

[9] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. Guerra, P. Joshi, and S. Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021. 4

[10] C. Frenkel, M. Lefebvre, J. Legat, and D. Bol. A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos. *IEEE Transactions on Biomedical Circuits and Systems*, 13(1): 145–158, 2018. 2

[11] E. Goi, Q. Zhang, X. Chen, H. Luan, and M. Gu. Perspective on photonic memristive neuromorphic computing. *PhotoniX*, 2020. 2, 3

[12] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861, 2017. arXiv preprint arXiv:1704.04861. 1

[13] M. Karamimanesh, E. Abiri, M. Shahsavari, K. Hassanli, A. van Schaik, and J. Eshraghian. Spiking neural networks on fpga: A survey of methodologies and recent advancements. *Neural Networks*, 2025. 4, 5

[14] T. Koickal, A. Hamilton, T. Pearce, S. Tan, J. Covington, and J. Gardner. Analog vlsi design of an adaptive neuromorphic chip for olfactory systems. In *IEEE International Symposium on Circuits and Systems*, 2006. 4

[15] D. Li, X. Chen, M. Becchi, and Z. Zong. Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In *IEEE International Conferences on Big Data and Cloud Computing (BDCloud)*, pages 477–484, 2016. 1

[16] J. Lisman and N. Spruston. Questions about stdp as a general model of synaptic plasticity. *Frontiers in Synaptic Neuroscience*, 2, 2010. 2

[17] S. Liu and T. Delbruck. Neuromorphic sensory systems. *Current Opinion in Neurobiology*, 20(3):288–295, n.d. 4

[18] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. 2

[19] D. Manna, A. Vicente-Sola, P. Kirkland, T. Bihl, and G. Di Caterina. Simple and complex spiking neurons: perspectives and analysis in a simple stdp scenario. *Neuromorphic Computing and Engineering*, 2(4), 2022. 3

[20] D. Manna, A. Vicente-Sola, P. Kirkland, T. Bihl, and G. Di Caterina. Frameworks for snns: a review of data science-oriented software and an expansion of spyketorch. In *International Conference on Engineering Applications of Neural Networks*, pages 227–238, 2023. 2, 3

[21] D. Muir and S. Sheik. The road to commercial success for neuromorphic technologies. *Nature Communications*, 16(1), 2025. 2, 4, 5

[22] D. Owen-Newns, A. Adair, D. Black, G. Donati, J. Robertson, and A. Hurtado. Photonic spiking neurons and spiking neural networks. In *SPIE Proceedings*, 2024. 2

[23] M. Pfeiffer and T. Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12, 2018. 3

[24] T. Potok, C. Schuman, R. Patton, T. Hylton, H. Li, and R. Pino. Neuromorphic computing, architectures, models, and applications. a beyond-cmos approach to future computing. Technical report, USDOE Office of Science (SC), Oak Ridge, TN, 2016. 4, 5

[25] J. Robertson, M. Hejda, J. Bueno, and A. Hurtado. Ultrafast optical integration and pattern classification for neuromorphic photonics based on spiking vcsel neurons. *Scientific Reports*, 10(1), 2020. 2, 3

[26] C. Schuman, T. Potok, R. Patton, J. Birdwell, M. Dean, G. Rose, and J. Plank. A survey of neuromorphic computing and neural networks in hardware. https://arxiv.org/abs/1705.06963, 2017. arXiv preprint arXiv:1705.06963. 1, 2

[27] A. Shrestha, H. Fang, Z. Mei, D. Rider, Q. Wu, and Q. Qiu. A survey on neuromorphic computing: Models and hardware. *IEEE Circuits and Systems Magazine*, 22(2):6–35, 2022. 4

[28] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13693–13696, 2020. 1

[29] J. Summers. Game jams from around the world. In *Game Jams–History, Technology, and Organisation*, pages 69–86. Springer International Publishing, 2022. 5

[30] T. Urista. 10 key insights into neuromorphic computing: Challenges and potential, 2024. Accessed Oct. 4, 2024. 3

[31] A. Vanarse, A. Osseiran, and A. Rassau. A review of current neuromorphic approaches for vision, auditory, and olfactory sensors. *Frontiers in Neuroscience*, 10, 2016. 2

[32] A. Vicente-Sola, D. Manna, P. Kirkland, G. Di Caterina, and T. Bihl. Keys to accurate feature extraction using residual spiking neural networks. *Neuromorphic Computing and Engineering*, 2(4), 2022. 3

[33] S. Wahid, A. Asad, and F. Mohammadi. A survey on neuromorphic architectures for running artificial intelligence algorithms. *Electronics*, 13(15), 2024. 1, 2

[34] G. Wallington. Seeing the future: How neuromorphic imaging is redefining machine vision, 2024. Accessed Nov. 24, 2024. 3

[35] D. Wofk, F. Ma, T. Yang, S. Karaman, and V. Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *International Conference on Robotics and Automation (ICRA)*, pages 6101–6108, 2019. 1

[36] P. Zhang, S. Zhu, C. Wang, Y. Zhao, and E. Lam. Neuromorphic imaging with super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024. 3