Title

Xyloni: Very Low Power Neural Network Accelerator for Intermittent Remote Visual Detection of Wildfire and Beyond

Authors

Chen, Jeffrey
Jun, Sang-Woo
Mundra, Aditi
et al.

Publication Date

2024-08-05

DOI

10.1145/3665314.3670810

Xyloni: Very Low Power Neural Network Accelerator for Intermittent Remote Visual Detection of Wildfire and Beyond

Jeffrey Chen
jeffrc2@uci.edu
University of California, Irvine
Irvine, California, USA

Jonathan Ta
tajk@uci.edu
University of California, Irvine
Irvine, California, USA

Aditi Mundra
aditimundra05@gmail.com
University of California, Berkeley
Berkeley, California, USA

Sang-Woo Jun
swjun@uci.edu
University of California, Irvine
Irvine, California, USA

ABSTRACT

Wildfires are one of the most catastrophic natural disasters, causing increasingly severe ecological and economic damage. Early response is critically important for wildfire management, but also difficult due to the wide geographical area to monitor, often far from utility infrastructures such as stable power and high-bandwidth network. In this work, we present Xyloni, a very low-cost, low-power neural network accelerator for sensor nodes, which improves the cost-effectiveness and scalability of real-time wildfire detection by drastically reducing wireless data transmission and overall power consumption. Xyloni uses low-power flash and FeRAM memories to store a hardware co-optimized Neural Network model for fire and smoke detection, as well as intermediate activations during inference. It also time-shares a Field-Programmable Gate Array across different model layers for power-efficient computation. The detection model prevents benign images from consuming network traffic, allowing the use of low-bandwidth, low-power network fabrics such as a LoRa mesh network with enough range for the necessary geographical coverage. Compared to a wide range of edge and sensor platforms capable of real-time data collection, Xyloni demonstrated an order of magnitude reduction in power consumption for the network transmission reduction task, leading to a corresponding reduction in battery and deployment cost.

CCS CONCEPTS

• **Hardware → Hardware accelerators**; **Chip-level power issues**; • **Computer systems organization → Embedded systems**; **Neural networks**.

KEYWORDS

Edge computing, wildfire detection, CNN, FPGA, NVM, LoRa, Mesh

ACM Reference Format:
Jeffrey Chen, Jonathan Ta, Aditi Mundra, and Sang-Woo Jun. 2024. Xyloni: Very Low Power Neural Network Accelerator for Intermittent Remote Visual Detection of Wildfire and Beyond. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '24), August 5–7, 2024, Newport Beach, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3665314.3670810

ISLPED '24, August 5–7, 2024, Newport Beach, CA, USA

ACM ISBN 979-8-4007-0688-2/24/08
https://doi.org/10.1145/3665314.3670810

1 INTRODUCTION

Wildfires are one of the most catastrophic natural disasters which regularly causes very large-scale ecological and economic damage. Within the US state of California alone, wildfires between 2017 and 2021 have, on average, caused 117 billion dollars of economic loss [7]. Unfortunately, the damage caused by wildfires is expected to increase in the foreseeable future due to the impact of the changing climate. The Fire Weather Index (FWI), which calculates the index of fire intensity potential, is expected to increase by 2.1 to 3.3% per decade in Southern Europe [6], and has increased by 20% over the last four decades in California [8]. The scale of potential damage emphasizes the importance of real-time detection of wildfire events. Reducing the average fire response times by 15 minutes is expected to reduce economic impacts by 3.5 to 8.2 billion dollars [7].

One of the most promising family of approaches to wildfire detection is using Computer Vision, via either unmanned aerial drones or ground-based cameras due to their versatility, accuracy, and the ability to observe a wide area from a strategic location [17]. Various machine learning models have been developed to classify images based on existence of fire and smoke, or detect and locate fire and smoke from images, with very high accuracy and versatility.

Unfortunately, the major hurdle of deploying such technologies is not the accuracy of detection, but the scalability of data collection and processing [17]. Real-time monitoring conventionally requires a network connection capable of transmitting image data, but such high-bandwidth wireless networks are typically short range, and power-hungry enough to quickly become the primary source of power consumption for embedded systems [9], increasing the cost of batteries and limiting the coverage of drones. Low-power long-range fabrics do exist, such as LoRa organized into a multi-hop mesh (Figure 1). However, the low bandwidth of each LoRa link (few KB/s) means the mesh nodes near the gateway can become a severe bottleneck. While edge processing, which migrates computation near the sensors, can mitigate the bandwidth issue, the added computation can also significantly increase the net power consumption and cost of nodes.

To address these concerns, we present Xyloni, a neural network accelerator designed for minimizing power consumption within real-time performance requirements. Xyloni uses low power flash memory and Ferroelectric memory (FeRAM) to store the neural network weights and intermediate activations. It also uses a very low-power Field-Programmable Gate Array (FPGA) in a time-shared fashion between neural network layers. For wildfire detection, Xyloni hosts an accurate, hardware co-optimized and 8-bit quantized Convolutional Neural Network (CNN) for classifying images with fire or smoke. We note that the CNN's task is not to offload the

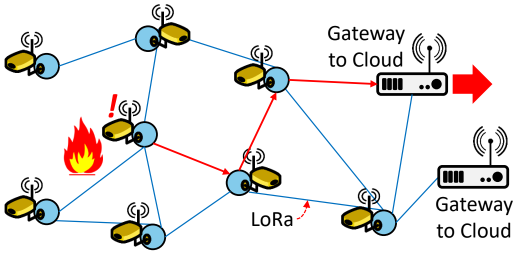

Figure 1: LoRa nodes organized into a multi-hop mesh.

central server's analytical workload, which will probably involve more complex models, but to prevent images with low probability of fires from consuming network resources.

Our Xyloni prototype demonstrates extreme network filtering via full-fledged CNN on images, within a half-Watt power budget (430 mW), and within the minute-scale real-time requirements of wildfire monitoring [19]. Compared to a wide range of off-the-shelf systems capable of real-time filtering, Xyloni achieves a multifold reduction of power consumption, resulting in a corresponding reduction in battery and power harvester cost. Compared to the next best system, the Raspberry Pi Zero 2, Xyloni reduces the recommended power system capacities by over 10× [18]. While the power-hungry Raspberry Pis and edge GPUs can achieve higher throughput, and sometimes even higher throughput per Watt compared to Xyloni, the network bottleneck limits the utility of higher performance. Meanwhile, high-current batteries are expensive, making Xyloni the more attractive system as long as real-time goals are met. Xyloni is also efficient under intermittent power availability thanks to progress checkpointing via nonvolatile FeRAM. The claimed contribution of this paper is demonstrating that a machine learning accelerator using low-power flash memory and FeRAM for memory, and time-sharing a low-power FPGA can:

- Sufficiently minimize data transmission via filtering.
- Perform real-time filtering per domain requirements.
- Minimize power consumption for low deployment cost.

The rest of this paper is organized as follows: We present background and related works in Section 2. We present the detailed architecture of Xyloni in Section 3, and evaluate it against a spectrum of available edge systems in Section 4. We conclude with discussion in Section 5.

2 BACKGROUND

Early detection of wildfires over a wide area is critical for mitigating their catastrophic impacts [7]. Real-time monitoring platforms aim for latencies at scale of a few minutes [16, 19, 20].

Computer Vision for Wildfire Management. Real-time wildfire detection via computer vision is a rapidly emerging technology, by analyzing image data collected by a variety of sources including satellite [3], autonomous drones [2], and ground-based stationary cameras [17]. Vision-based approaches have been repeatedly proven useful thanks to the accurate and detailed analysis possible, as well as the wide range covered by a well-placed camera, compared to more local sensing technologies such as temperature or smoke sensors. Many machine learning approaches to computer vision have proven effective, and a wide variety of models being developed.

Network Bandwidth and Cost. Vision-based approaches are often not feasible at the scale of deployment to achieve necessary geographic coverage, as high-bandwidth transmission required for image transmission is power-hungry (~1s W) and short-range [17], and necessitate a direct connection to a gateway such as 5G cell towers. Deployment costs then increase due to the cost of necessary batteries and power harvesters, as well as gateway construction.

Other approaches try to exploit very wide-range, low-power (~100s mW) network fabrics such as LoRa meshes, which use peer-to-peer mesh routing instead of a central gateway [13], using lower-bandwidth sensors including temperature and air quality sensors [16]. However, such networks have severely limited bandwidth, since they are often limited by the low bandwidth (~few KB/s) of a small number of links between the gateway and nearby nodes, as seen in Figure 1.

Edge Processing for Filtering. One promising solution to the bandwidth concern is edge processing, where collected data is processed closer to the sensors where data is collected, and only the the distilled analysis results get transmitted, allowing the use of wide-range, low-bandwidth network fabrics. A spectrum of architectures are explored, including a flexible fleet of edge servers between the central cloud and the sensor nodes ("fog computing") [12] to sensor nodes themselves augmented with computation [15].

While edge processing can reduce network transmission, it can actually end up increasing cost and power consumption, due to the high computation requirements of machine learning models. Specialized edge accelerators aim to remedy this overhead via embedded GPUs (e.g., NVIDIA Jetson) TPUs (e.g., Coral Edge TPU [4]), and FPGAs [10, 21, 22]. Unfortunately, accelerators targeting *highly accurate* complex machine learning tasks like vision still suffer multi-Watt power budgets [10, 21], excluding extreme cases like binarized neural networks which often trade accuracy for efficiency [1]. This is partially because the on-board DRAM capacity necessary to host models comes with a large power budget [14].

3 XYLONI ARCHITECTURE

Figure 2 shows the overall architecture of Xyloni, an FPGA accelerator augmented with non-volatile memories, and attached to the host microcontroller unit (MCU) over the Serial Peripheral Interface (SPI) and Quad Serial Peripheral Interface (QSPI). Since Xyloni aims to minimize cost and power consumption, it uses the Lattice iCE40 UP5K, a low-cost (~$5), low-power (20$\mu W$ budget) FPGA with small amounts of on-chip memory (128 KB) and Digital Signal Processing (DSP) blocks. The host MCU is in hibernation most of the time, only waking periodically to initiate accelerator kernels, wirelessly transmit alive indications to the server, or to transmit *raw image* data in the rare case when fire or smoke is detected.

Two types of low-power nonvolatile memory are used to extend memory capacity: multiple 16-MB QSPI Flash Memory chips with sufficient total capacity to host read-only model parameters, due to their limited overwrite lifetime; and a multi-MB Ferroelectric memory (FeRAM) with much better write durability for intermediate activations. Both memories are fastest with large block accesses,

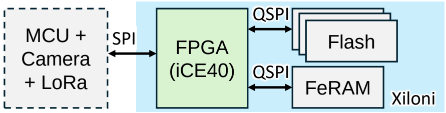

Figure 2: Xyloni node architecture

where bursts of dozens of bytes are useful to minimize command overhead.

3.1 Hardware Co-optimized CNN Model

For Xyloni, we have developed an image-based wildfire detection CNN model based on CaffeNet [11], co-optimized with the Xyloni architecture. We call the resulting model *Shallow CaffeNet*, and its structure is compared against the original CaffeNet in Table 1. Shallow CaffeNet reduces model weight capacity from **300.2 MB** to **43.4 MB**, without significant loss of accuracy. Details about the optimization approaches are described below.

Table 1: Model Architecture Changes to Wildfire Detection CaffeNet (*W is window, D is dimensionality, →↓ is stride*)

CaffeNet [11]	**Shallow CaffeNet**
Input:224x224x3 (32-bit)	Input:250x250x3 (8-bit)
Conv2D W:11x11 D:96 →↓4,4	Conv2D W:12x12 D:96 →↓4x4
ReLU	BatchNormalization
MaxPool2D W:3x3 →↓2,2	MaxPool2D W:4x4 →↓2,2
BatchNormalization	ReLU
Conv2D W:5x5 D:256→↓1,1	Conv2D W:5x5 D:256 →↓1,1
ReLU	BatchNormalization
MaxPool2D W:3x3→↓2,2	MaxPool2D W:4x4 →↓2,2
BatchNormalization	MaxPool2D W:4x4 →↓2,2
Conv2D W:3x3 384→↓1,1	ReLU
ReLU	Dense D:4096
Conv2D W:3x3 384→↓1,1	ReLU
ReLU	Dense D:4096
Conv2D W:3x3 384→↓1,1	ReLU
ReLU	Dense D:1
MaxPool2D W:3x3 →↓2,2	Sigmoid
Dense D:4096	
ReLU	
Dense D:4096	
ReLU	
Dense D:4096	
Softmax	
Size: 300.2 MB	Size: 43.4 MB

The Shallow CaffeNet design was the result of design space exploration aiming to optimize non-volatile storage accesses, accelerator resource efficiency, as well as model size. First, convolution filter sizes are increased from 11×11 to 12×12, to reduce fragmentation due to fixed-size bursts from QSPI Flash Memory, while more parameters helped retain accuracy with fewer layers. Second, three convolution layers are removed, since they did not have a significant impact on accuracy. Third, the max pooling window is increased from 3 to 4, so that the pooling window dimensions are proportionally 2-to-1 to the stride dimensions. This enables efficient architectural optimizations presented in Section 3.2.2 for more efficient on-chip memory resource management. Finally, we moved the ReLU activation functions to after max pooling to reduce the number of activation function calls.

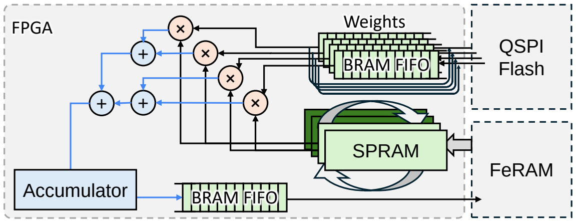

Figure 3: 2D Convolution configuration microarchitecture.

The resulting shallow model was quantized using quantization-aware training on TensorFlow, TensorFlow Lite, and TensorRT. We target uniform symmetric linear quantization, where weights are centered around zero, the quantization scale is linear, and every layer has the same scale factor. Symmetric linear scales greatly simplify quantized arithmetic. And a shared scale factor can be efficiently embedded into hardware in a static fashion.

3.2 Xyloni Accelerator Microarchitecture

Xyloni time-shares the FPGA across different stages, and focuses all chip resources into each layer being processed. A separate FPGA configuration bitfile is created for each stage and stored in the onboard SPI flash. For each image, the MCU copies it to the accelerator FeRAM and initiates the first stage. When one stage is done, the FPGA wakes the host MCU from hibernation to initiate the next stage. The Shallow CaffeNet in Table 1 is divided into three different stages: Conv2D, Batch+Pool, and Dense.

3.2.1 Configuration 1: Conv2D. Figure 3 illustrates the microarchitecture of the Convolution configuration. Either input image or the previous layer's feature map is received from the FeRAM, into the four available SPRAM units organized into an alternating double buffer. Weights are loaded from the QSPI Flash module into a set of 4 BRAM FIFOs, where loaded weights can be re-used for each input feature map. The SPRAM accesses on the UP5K have 16-bit ports, so we can retrieve four 8-bit input values every cycle from the two active SPRAM blocks currently being read. This data, coupled with the weights from the four FIFOs, are then fed into a multiply-accumulate pipeline, with output being accumulated for each convolution. The output values are cached in output BRAM and then written efficiently via bursts to the FeRAM.

3.2.2 Configuration 2: Batch Normalization and Max Pooling. Due to relatively light computation requirements compared to other stages, Xyloni implements both batch normalization and max pooling units in the same stage. Figure 4 illustrates the microarchitecture of this stage, where both functions are implemented.

Batch normalization. To reduce computation and storage and storage access, the runtime behavior of batch normalization is simplified. As the batch normalization weights (β, γ) are constants

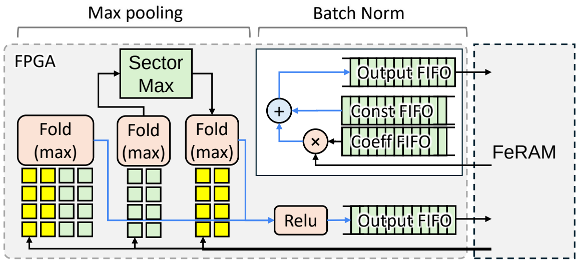

Figure 4: Batch normalization + pooling microarchitecture.

during inference, the original Equation 1 is re-organized into Equation 2, such that most computation between static values can be pre-computed. During runtime, each input only needs to go through one multiplication and one addition against two weights (a coefficient and a constant, highlighted gray in Equation 2).

$$output = (input - mean)/stdev * \gamma + \beta \tag{1}$$

$$= input * \boxed{(\gamma/stdev)} + \boxed{(\beta - (mean * (\gamma/stdev)))} \tag{2}$$

Inside the microarchitecture, the weights are stored and retrieved from a special region reserved at the back of the FeRAM address space. Weight pairs (const, coeff) are loaded into two BRAM FIFOs, and the the input feature map is loaded from FeRAM and streamed into the operator tree. Output is written in bursts using an output BRAM FIFO.

Max pooling. Figure 10 also illustrates the microarchitecture for max pooling, for a 4×4 window and stride of 2, as used by the Shallow CaffeNet model. The design goal of this microarchitecture is to minimize redundant data loads despite the existence of overlapping strides, while also supporting random accesses. To support this, we implement two full sets of pooling windows. One full ("main") window, and two half windows. Input is read in as values from 4-byte contiguous bursts four times to fill its main pooling window in a 16-byte vector. While the main pooling window is being filled, the back halves and front halves of the window, split by the stride, are stored separately. Once filled, the max values of the main and back windows are calculated, and emitted. Then, the next non-overlapping block is read in, and during the process, the max of the new front window and the max of the previous back window can be joined to compute the max value of the overlapping window. The ReLU activation function is performed during output.

3.2.3 Configuration 3: Dense layer. Figure 5 shows the microarchitecture for the Dense unit, which executes three dense layers sequentially, and does its best to perform on-chip caching between layers. The Dense unit receives input from FeRAM to load into a BRAM FIFO, and performs multiply-accumulate with the weights received from the QSPI Flash. Input data is re-enqueued into the input FIFO for reuse across weight matrix rows. For the intermediate dense layers (e.g., first two layers in Shallow CaffeNet), ReLU activation is applied to each accummulated results and then pushed into the output FIFO. On the final layer, the raw accumulation results are pushed enqueued without ReLU. The output stream is collected in the output BRAM FIFO and re-used as input for the next dense layer (if any). The final output is emitted to the host MCU either via FeRAM or through SPI.

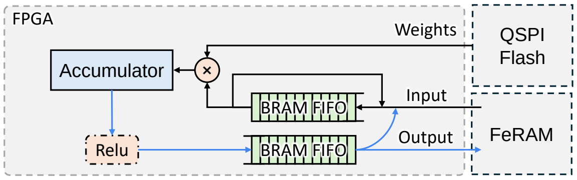

Figure 5: Dense layer microarchitecture.

Table 2: UP5K chip utilization for each accelerator instance.

Instance	LC	BRAM	SPRAM	DSP
Convolution	4111 (77%)	30 (100%)	4 (100%)	8 (100%)
Batch+Pool	2632 (50%)	30 (100%)	0 (0%)	2 (25%)
Dense	3201 (61%)	30 (100%)	0 (0%)	2 (25%)

The performance of the dense layer is primarily limited by how fast weights can be read from the QSPI Flash chips. This is partly because Xyloni is operating at batch size of 1, to minimize latency. Higher throughput can be attained if Xyloni is allowed to collect, say, 4 images each 5 to 10 seconds apart, for larger batch processing at the cost of dozens of seconds of latency. Image batching must be done at such large intervals, because immediately back-to-back images likely will not have much variance.

4 EVALUATION

4.1 Implementation Details

Table 2 shows the FPGA resource utilization of Xyloni across the three configurations. We achieve a balanced division of work across configurations. All designs achieve a 22 MHz clock speed.

We note that all designs make efficient use of on-chip resources, including 100% BRAM utilization for burst I/O, up to 100% SPRAM and up to 100% DSP usages for caching and parallel processing.

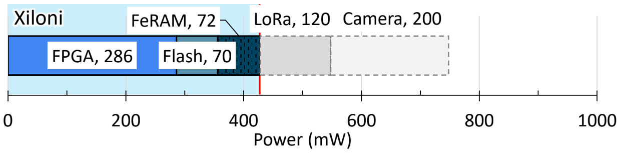

Figure 6: Component power consumption breakdown.

Figure 6 presents the component power breakdown. The external memory additions, the QSPI Flash and FeRAM modules, draw similar amounts of low power that combined are still less than half of the FPGA power consumption.

4.2 Model Accuracy Evaluation

Figure 7 presents the change in model accuracy, starting from the unmodified CaffeNet to the eventually deployed, hardware co-optimized and quantized version. All models were trained and tested on the Wildfire Detection Image dataset [5].

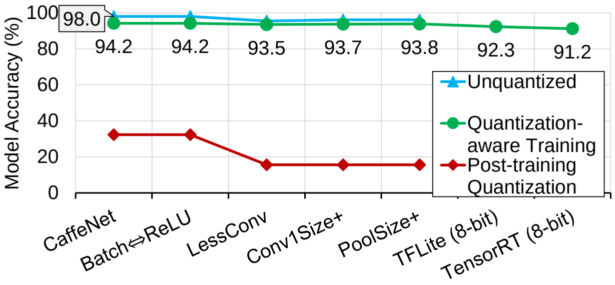

Figure 7: CNN model accuracy with model modifications.

The final hardware-optimized quantized model achieves **91.2%** accuracy, a mere 3% drop compared to the dynamically quantized version, and 6-7% lower than the unquantized original.

We emphasize that this model is only used on the edge device for network filtering. Much more capable models may be used at the central server to analyze raw images sent after filtering.

Table 3: Evaluated off-the-shelf platforms.

Platform	CPU	Cores	Clock (MHz)	Suff. Mem	Power (W)
Arduino Uno	M0	1	16	×	~0.3
Arduino Due	M3	1	84	×	~0.8
Arduino Teensy4	M7	1	600	×	~0.4
RPi Zero 2	A53	1	1000	○	~1.8
RPi 4 B	A72	4	1500	○	~5
NVIDIA Jetson AGX Xavier	Volta (GPU)	512	1211	○	~15

4.3 Evaluated Platforms

We compared Xyloni against a wide variety of embedded platforms typically used for sensor deployment, spanning 8-bit microcontrollers to embedded GPUs. Table 3 summarizes the systems. All platforms with sub-watt typical active power consumption could not support even our quantized model in memory ("Suff. Mem"). More powerful systems with sufficient memory capacity also consumed proportionally larger power.

4.4 Comparison Against Real-Time Systems

Figure 8 compares the performance and power efficiency of the systems capable of supporting the CNN model in memory. While Xyloni perform poorly compared to the much costlier and power-hungry systems, Xyloni does achieve sufficient performance for the domain-specific minute-scale real-time requirement (red hatches), as well as competitive power efficiency.

Once real-time requirements are met, however, higher performance has diminishing returns. Instead, low power consumption starts being more important, because it has direct impact on the cost and lifetime of the deployment. Figure 9 compares the real-time systems based on average active power consumption as well as peak power based on the recommended power system capacity [18], and shows that Xyloni achieves multifold reduction in power consumption, as well as sub-watt power consumption. Compared to the most low-power RPi Zero, Xyloni reduces power consumption by over 4× based on average power, and over 10× based on peak power. Furthermore, Xyloni supports intermittent computation using the non-volatile FeRAM module for checkpointing, and the consistent accelerator behavior results in very regular power consumption, allowing near-threshold, intermittent power budgets.

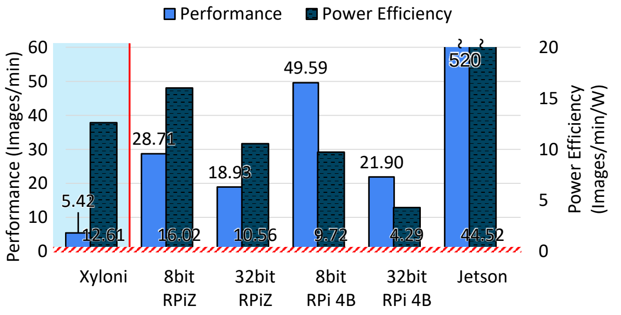

Figure 8: Performance and power efficiency comparison.

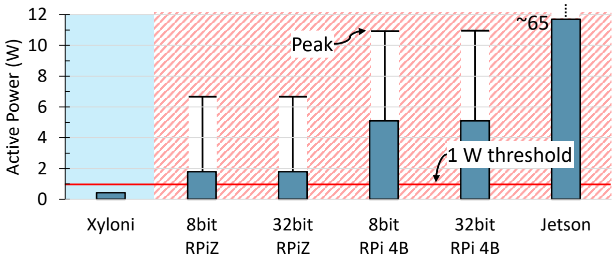

Figure 9: Power consumption comparison.

Reduction of power budgets have a direct impact on the deployment cost, since the cost of off-the-shelf batteries and power harvesters are often directly proportional to their standard discharge current, as seen in Figure 10. The recommended batteries for Xyloni and the Raspberry Pi Zero 2 are highlighted (Both with 1.2 Ah capacity). The battery can end up accounting for the majority of the sensor node, considering the RPi Zero device itself costs less than $30 as of 2024. The production cost of Xyloni is difficult to

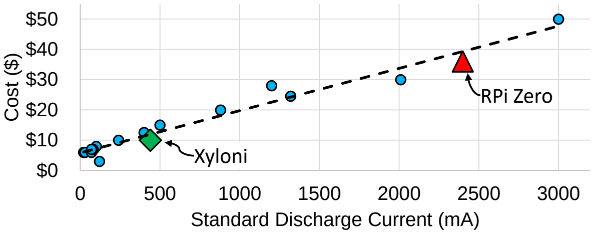

Figure 10: Battery price versus standard discharge current.

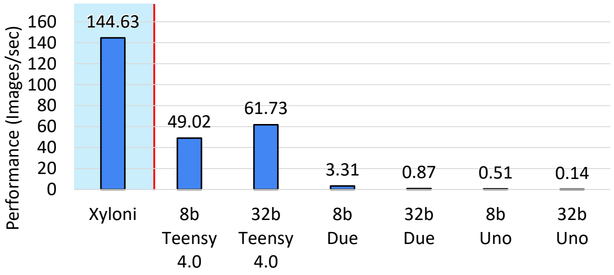

Figure 11: Embedded systems performance comparison

predict, but each component (e.g., FPGA, memory) is lower-cost compared to RPi Zero.

4.5 Comparison Against Embedded Systems

We first emphasize that the systems evaluated in this section lack sufficient memory to perform CNN inference. But for completeness, we evaluate some sub-components that can be run or estimated.

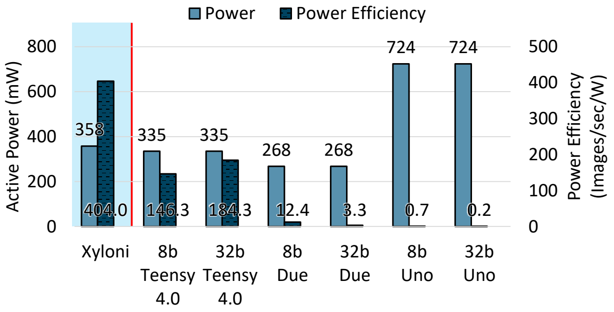

Figure 12: Embedded systems power and efficiency

As the dense layer weights exceeded the memory capacity of all embedded systems, we present a proxy evaluation by extrapolating on executing a subset of the convolution layers, only as much as can fit in memory. Effectively, we are evaluating the comparison systems under the unrealistically favorable scenario where they can magically procure sufficient on-chip SRAM. Figure 11 compares the convolution performance between embedded systems, and shows that Xyloni, using FeRAM for temporary storage, achieves superior performance even for memory-resident workloads.

Figure 12 compares the active power consumption and power efficiency of embedded systems, and shows Xyloni consumes comparable amounts power compared to embedded systems, with much higher power efficiency. We note that Xyloni power consumption numbers included the FeRAM but not the QSPI Flash module, to match the scale of memory capacity.

5 CONCLUSION

We present Xyloni, a very low power neural network accelerator for sensor nodes, capable of real-time filtering of captured image data for the purpose of minimizing network transmission. Thanks to Xyloni, a sensor node can take advantage of very long-range and low-power, but also very low-bandwidth network fabric such as LoRa, while also significantly reducing the cost and power consumption of each node. We believe this approach can finally enable the wide-area deployment of early detection of wildfires and beyond.

REFERENCES

[1] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. 2018. Hyperdrive: a systolically scalable binary-weight CNN inference engine for mW IoT end-nodes. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE.

[2] Abdelmalek Bouguettaya, Hafed Zarzour, Amine Mohammed Taberkit, and Ahmed Kechida. 2022. A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms. *Signal Processing* 190 (2022), 108309.

[3] Emilio Chuvieco, Inmaculada Aguado, Javier Salas, Mariano García, Marta Yebra, and Patricia Oliva. 2020. Satellite remote sensing contributions to wildland fire science and management. *Current Forestry Reports* 6 (2020), 81–96.

[4] Coral. [n. d.]. Coral Dev Board Micro datasheet. https://coral.ai/static/files/Coral-Dev-Board-Micro-datasheet.pdf

[5] Baris Dincer. [n. d.]. Wildfire Detection Image Data For Machine Learning. https://www.kaggle.com/datasets/brsdincer/wildfire-detection-image-data/

[6] Jean-luc Dupuy, Hélène Fargeon, Nicolas Martin-StPaul, François Pimont, Julien Ruffault, Mercedes Guijarro, Carmen Hernando, Javier Madrigal, and Paulo Fernandes. 2020. Climate change impact on future wildfire danger and activity in southern Europe: a review. *Annals of Forest Science* 77 (2020), 1–24.

[7] Gordon and Betty Moore Foundation. [n. d.]. The economic, fiscal, and environmental costs of wildfires in California. https://www.moore.org/article-detail?newsUrlName=the-economic-fiscal-and-environmental-costs-of-wildfires-in-california

[8] Michael Goss, Daniel L Swain, John T Abatzoglou, Ali Sarhadi, Crystal A Kolden, A Park Williams, and Noah S Diffenbaugh. 2020. Climate change is increasing the likelihood of extreme autumn wildfire conditions across California. *Environmental Research Letters* 15, 9 (2020), 094016.

[9] Sotirios K Goudos, Panagiotis I Dallas, Stella Chatziefthymiou, and Sofoklis Kyriazakos. 2017. A survey of IoT key enabling and future technologies: 5G, mobile IoT, sematic web and applications. *Wireless Personal Communications* 97 (2017), 1645–1675.

[10] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, Song Han, Yu Wang, and Huazhong Yang. 2017. Angel-eye: A complete design flow for mapping CNN onto embedded FPGA. *IEEE transactions on computer-aided design of integrated circuits and systems* 37, 1 (2017), 35–47.

[11] Simon Philipp Hohberg. 2015. *Wildfire smoke detection using convolutional neural networks.* Ph. D. Dissertation.

[12] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa A Cherif, Hossam Afifi, and Mohsen Guizani. 2021. Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications* 180 (2021), 210–231.

[13] Huang-Chen Lee and Kai-Hsiang Ke. 2018. Monitoring of large-area IoT sensors using a LoRa wireless mesh network system: Design and evaluation. *IEEE Transactions on Instrumentation and Measurement* 67, 9 (2018), 2177–2187.

[14] Sukhan Lee, Hyunyoon Cho, Young Hoon Son, Yuhwan Ro, Nam Sung Kim, and Jung Ho Ahn. 2018. Leveraging power-performance relationship of energy-efficient modern DRAM devices. *IEEE Access* 6 (2018), 31387–31398.

[15] Qianlin Liang, Prashant Shenoy, and David Irwin. 2020. Ai on the edge: Characterizing ai-based iot applications using specialized edge architectures. In *2020 IEEE International symposium on workload characterization (IISWC)*. IEEE, 145–156.

[16] Dryad Networks. [n. d.]. Ultra Early Wildfire Detection | Dryad Networks. https://www.dryad.net/

[17] Leo Ramos, Edmundo Casas, Eduardo Bendek, Cristian Romero, and Francklin Rivas-Echeverría. 2024. Computer vision for wildfire detection: a critical brief review. *Multimedia Tools and Applications* (2024), 1–44.

[18] raspberrypi. [n. d.]. Raspberry Pi: Power supply. https://github.com/raspberrypi/documentation/blob/develop/documentation/asciidoc/computers/raspberry-pi/power-supplies.adoc

[19] Google Research. [n. d.]. Using AI to provide accurate wildfire information to affected communities and fire authorities. https://sites.research.google/wildfires/

[20] FIRMS US/CANADA. [n. d.]. Fire Information for Resource Management System US/Canada. https://firms.modaps.eosdis.nasa.gov/usfs/active_fire/

[21] Xiaofan Zhang, Anand Ramachandran, Chuanhao Zhuge, Di He, Wei Zuo, Zuofu Cheng, Kyle Rupnow, and Deming Chen. 2017. Machine learning on FPGAs to face the IoT revolution. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 894–901.

[22] Zhichao Zhang, MA Parvez Mahmud, and Abbas Z Kouzani. 2022. Fitnn: A low-resource fpga-based cnn accelerator for drones. *IEEE Internet of Things Journal* 9, 21 (2022), 21357–21369.