

Kraken: An Open-Source RISC-V SoC for Ultra-Low Power Multi-Modal Perception

Viviane Potocnik

`vivianep@iis.ee.ethz.ch`

ETH Zurich

Alfio Di Mauro

ETH Zurich

Christoph Leitner

ETH Zurich

Moritz Scherer

ETH Zurich

Georg Rutishauser

ETH Zurich

Lorenzo Lamberti

University of Bologna

Luca Benini

ETH Zurich

Article

Keywords: RISC-V, multi-modal AI fusion, extreme-edge AI

Posted Date: April 4th, 2024

DOI: <https://doi.org/10.21203/rs.3.rs-4023416/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: There is **NO** Competing Interest.

Kraken: An Open-Source RISC-V SoC for Ultra-Low Power Multi-Modal Perception

Viviane Potocnik¹, Alfio Di Mauro¹, Christoph Leitner¹,
Moritz Scherer¹, Georg Rutishauser¹, Lorenzo Lamberti¹,
Luca Benini^{1,2}

¹Integrated System Laboratory, ETH Zurich, Gloriastrasse, 35, Zurich,
8092, , Switzerland.

²DEI, University of Bologna, , Bologna, Italy.

Contributing authors: vivianep@iis.ee.ethz.ch; adimauro@iis.ee.ethz.ch;
christoph.leitner@iis.ee.ethz.ch; scheremo@iis.ee.ethz.ch;
geogr@iis.ee.ethz.ch; lorenzo.lamberti@unibo.it; lbenini@iis.ee.ethz.ch;

Abstract

We introduce Kraken, a highly flexible heterogeneous system-on-chip (SoC) fabricated in 22 FDX technology demonstrating leading-edge energy efficiency and computational capabilities for ultra-low power extreme-edge applications. Inspired by the multi-modal perception system of mammals, Kraken supports multi-modal sensing and perception via event-based and sampling-based sensor interfaces coupled with a spiking neural network accelerator (SNE) and a ternary neural network accelerator (CUTIE), respectively. An octa-core 32-bit RISC-V cluster with ML and DSP extensions enables sensor fusion and flexible parallel data processing. A fabric controller (FC) featuring a dedicated core manages all compute subsystems and low-computation intensity tasks. We analyzed Kraken's performance and efficiency: SNE consumes 2.6 pJ/SOp and 470 μ J/inference for eye gaze prediction with 64% accuracy. CUTIE achieves 590 TOP/s/W and 2.72 μ J/inference for foveated object classification with 88% accuracy. The RISC-V cluster consumes 0.45 pJ per 8-bit MAC operation, and 12.01 μ J/inference for eye tracking with a mean average error of 0.24. All of Kraken's subsystems improve upon the state-of-the-art (SOA), while the heterogeneous architecture offers extreme flexibility in supporting ultra-low power multi-modal AI fusion applications.

Keywords: RISC-V, multi-modal AI fusion, extreme-edge AI

1 Introduction

The number of AI-enabled connected devices will double over the next decade [1], driven by the need for real-time decision-making, enhanced privacy, and network efficiency, under tight energy constraints dictated by the form factor, weight, and battery lifetime of internet of things (IoT) and wearable devices. In this context, multi-modal sensing and sensor fusion techniques are key, as they help overcome individual sensor limitations in contextual understanding applications [2, 3].

To meet efficiency demands at the edge, three dominant paradigms have emerged: *sparsity* [4], *quantization* [5], and *parallelization* [6]. To enable the execution of complex tasks on edge devices with limited energy and storage capacity, specialized accelerators [7, 8] and neuromorphic chips [9, 10] have been designed leveraging these paradigms for the dominant artificial intelligence (AI) workload; namely deep neural networks (DNNs). In augmented reality (AR) and extended reality (XR) applications, the challenges intensify, demanding real-time, high-fidelity processing and interaction with the physical environment, as well as a high degree of flexibility to handle a heterogeneous mix of sensory data sources and perception tasks.

In this paper, we introduce Kraken, a highly heterogeneous, flexible, and general-purpose SoC architecture inspired by the multi-modal perception and fusion capability of the mammalian cerebral cortex. Kraken features dedicated data acquisition pipelines for two classes of digital vision-sensor: sampling-based, i.e., RGB and gray-scale images collected from frame-based cameras over a camera parallel interface (CPI), and event-based, collected from a dynamic visual sensor (DVS) [11] over the dynamic visual sensor interface (DVSI). Furthermore, it supports the acquisition of digital samples over serial protocols, e.g., Electro-Oculography (EOG) signals recorded via an analog front-end, connected over the quad-serial peripheral interface (QSPI). Kraken utilizes specialized accelerators: the completely unrolled ternary neural inference engine (CUTIE), the sparse neural engine (SNE), and an eight-core RISC-V cluster with instruction set architecture (ISA) extensions for digital signal processing. An additional RISC-V core, called fabric controller (FC), orchestrates high-level, complex cognitive applications, such as sensor fusion, tracking, and planning, acting as the main SoC controller. Kraken’s heterogeneous architecture enables simultaneous processing and fusion of various sensors, achieving millisecond latency and microjoule energy consumption without compromising flexibility.

This work presents an extensive analysis of Kraken’s subsystems. We demonstrate Kraken’s effectiveness in executing the algorithmic building blocks of complex foveated vision algorithms. An exemplary end-to-end application classifies objects based on the user’s gaze focal point, a common task for XR/AR. Kraken achieves leading-edge performance and efficiency, measured on silicon, on all its subsystems within a total power envelope of only 46.3 mW:

1. Sparse Neural Engine (SNE) - the digital spiking neural network (SNN) accelerator for event-based sensors estimates pupil position, size, and orientation with a four-layer convolutional spiking neural network to predict the gaze vector directly on a DVS event stream, consumes only 2.6 pJ/SOP and 470 μ J/inf at 94 inf/s, with an accuracy of 64%.

2. CUTIE - the ternary neural network (TNN) accelerator for sampling-based sensors performs object classification on the CIFAR-10 dataset with an eight-layer ternary neural network at 3.33-fJ/OP, 2.7 μ J/inf, and 0.2 ms/inf, with an accuracy of 88%.
3. The parallel multi-core cluster with ISA extension for tunable-precision programmable processing in integer and floating point arithmetic, executes a linear regression task for eye gaze tracking from EOG data at 2.76-pJ/OP, and 12.01- μ J/inf and 0.35 ms/inf, achieving 0.24 and 0.09 mean average error (MAE) on x and y coordinates, respectively.
4. The fabric controller (FC), consumes only 7.1 mW when executing sensor fusion workload based on a Kalman filter, with a 33 μ s filter update latency, ultimately enabling the combination of multiple sensing modalities and complex application orchestration.

2 Kraken Chip Architecture

Kraken is based on the PULP open-source platform [12]. Its architecture consists of three subsystems: i) the FC, ii) an 8-core RISC-V compute cluster, and iii) the hardware acceleration domain hosting two accelerators for machine learning (ML)-based perception. Each operates on independent clocks and power supplies. Figure 1 shows a block diagram of the architecture.

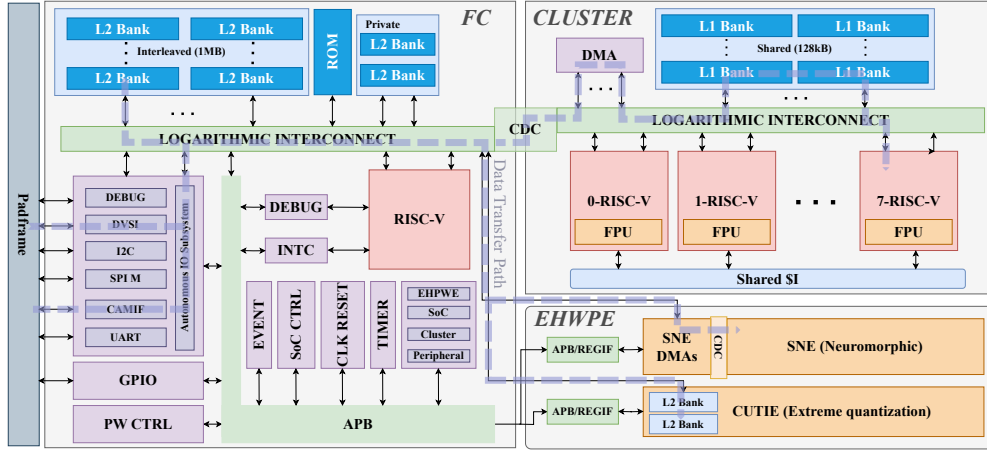


Fig. 1: Architectural Block Diagram of the Kraken System on Chip (SoC). The diagram shows the FC, cluster, and accelerator domain hosting two accelerators, SNE and CUTIE for ML-based perception. The peripherals are centered around the FC, and data paths detailed in Section 4 are highlighted in a lilac shade.

The FC, the central programmable control unit for the SoC, manages all system operations and peripherals. It features a 32-bit RI5CY core supporting the XPULPv2 RISC-V extension [13]. The FC hosts a 1MiB L2 SRAM memory and various peripherals, including CPI, DVS camera interfaces, QSPI, Inter-Integrated Circuit (I²C), and

universal asynchronous receiver transmitter (UART). To minimize the FC core intervention while acquiring sensor data, it integrates a μ DMA (Direct Memory Access) [14], capable of performing data transfers between the I/O peripherals and the L2 memory and/or the accelerators.

The cluster comprises eight 32-bit RISC-V cores executing parallel processing ML and signal processing applications. The cores implement the XPULPv2 and XPULPNN RISC-V ISA [15], enabling efficient 2- to 8-bit vector single instruction multiple data (SIMD) multiply-accumulate (MAC) operations. Each core has a private 32-bit floating-point unit (FPU) and shared 128 KiB tightly coupled data memory (TCDM) for fast, contention-free memory access.

The accelerator domain is optimized for ML-based perception. It comprises two compute engines: i) an all-digital sparse neural engine (SNE) [16], tailored for sparse neuromorphic workloads, specifically SNNs. SNE is designed for streamlined interaction with event-based sensors generating spikes, i.e., binary and time-stamped input events, that SNE processes in an event-driven fashion, and ii) CUTIE, designed for maximum inference energy efficiency and low latency by exploiting extreme quantization in TNNs to perform convolutions with a completely unrolled data path.

Two clocks are used to ensure efficient operation and allow power saving. The first, responsible for timing the interface between the accelerators and the FC subsystem, is generated by the FC frequency locked loop (FLL). A second FLL generates the clock for SNE and CUTIE.

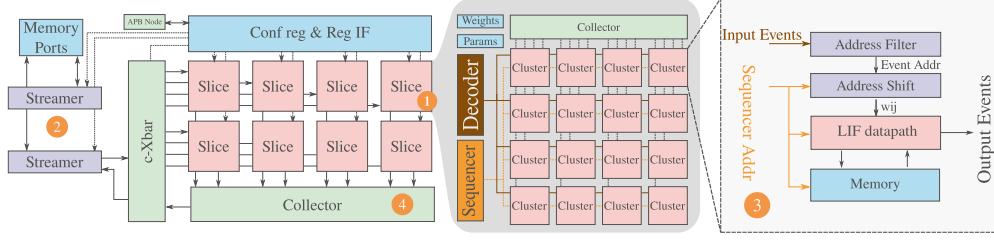
2.1 Sparse Neural Engine

SNE’s architecture is illustrated in Figure 2a. Kraken features eight parallel processing engines called *slices* ❶. Each connects to a central crossbar, linked to two programmable streamers ❷, operating as direct memory access (DMA) engines. The streamers work with the re-programmable crossbar, implementing versatile data transfer schemes between the L2 memory and SNE slices. Supported dataflows include streaming events from L2 memory to multiple slices (input event broadcast), streaming events from one slice to multiple slices (internal event redirection), or streaming events from multiple slices to the main memory (output event streaming).

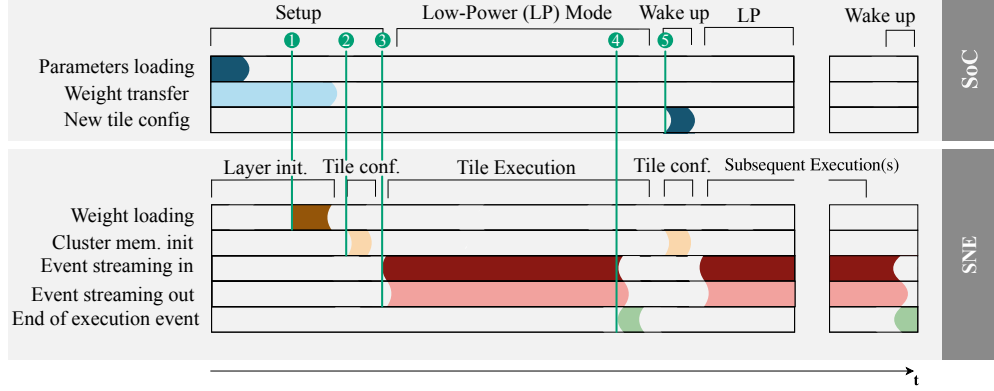
SNE accepts two kinds of events: *i) Spike events*, in the “list of coordinates” (COO) format, containing x , y , and channel (c) event coordinates, also employed by Kraken’s event-based peripherals, such as DVSI; the DVS produces positive and negative events mapped to two channels of the COO format, *ii) Time events*, i.e., 28-bit timestamps introduced between successive event-frames acquired from the DVS. Figure A1 shows a sample event package stored in an L2 event buffer consumed by SNE.

Slices contain 16 clusters ❸, each executing 64 concurrent leaky-integrate and fire (LIF) neuron functions using a time-domain multiplexing (TDM) based single digital neuron data path. Every neuron state during TDM execution is stored in a local standard cell-based memory inside the cluster, directly connected to the datapath ❹.

SNE accelerates two types of SNN layers: 3x3 convolutional layers (up to 256 input channels) and linear layers (maximum input dimension of 2304). Neurons are mapped on SNE’s computational units in output stationary format. The weights and parameters of LIF neurons, including threshold potential or layer-wise exponential decay time



(a) Data-path Schematic View of the Sparse Neural Engine (SNE). The diagram illustrates the time-domain multiplexed neural processing architecture, centralized event-driven crossbar system, streamers for efficient data handling, and clusters comprising LIF neurons.



(b) Scheduling diagram of the SoC (top) and SNE (bottom) for one layer inference. The neuron parameter and weight loading are executed once per layer. Once SNE starts its operation, the host is put into low-power mode until the tile is executed, and an end-of-execution event is issued to wake up the SoC. Then, new tile parameters are loaded, and the streamers are re-programmed to load the events of a new tile.

Fig. 2: Sparse Neural Engine (SNE) Architecture and Timing

constant, are pre-loaded via streamers into a shared buffer ①, accessible by all clusters simultaneously.

Events are broadcast to all slices and internal clusters, where each cluster responds only to input events falling within the receptive field of the neurons mapped to that cluster. The LIF data path in each cluster can process two types of events: temporal events, transmitted to advance the SNN time and induce exponential decay in neuron potential, and spike events, leading to an increase or decrease of the neuron potential based on the respective input-to-output neuron weight.

The clusters use a highly efficient update mechanism where only 9 (3×3 receptive field window) of 64 neurons respond to spike events by changing their potential. As the SNN timesteps progress, all 64 neurons must adjust their potential according to an exponential decay factor. SNE fuses neuron potential updates from spike and time events, performed in an event-driven fashion when a spike event is present. The *time of*

last update is registered per neuron and used to select a decay coefficient from a look up table (LUT), corresponding to the cumulative decay that would have happened from the *time of last update* to the current time step. This saves 55/64 neuron update operations on each cluster at every time step, which balances silicon area and throughput. To implement the LIF function for a cluster of 64 neurons, SNE uses a single data path in TDM for 12 cycles: 1 cycle to update each of the nine activated neurons with the fused spike-time update and 3 cycles to restart the mechanism before processing the next input event.

The cluster operates in lockstep, orchestrated by a sequencer ③. Multiple output events might be produced by the clusters in the same cycle. These are collected and transferred outside the slice through a buffering and arbitration unit called collector ③. A similar mechanism merges the event streams of multiple slices ④.

SNE is energy-proportional, i.e., the execution time of a layer-wise network inference is proportional to the number of input and output events of each layer, reflecting the network’s sparsity. Figure 2b shows the timing diagram of the SNE tiled inference.

SNN layer execution starts by pre-loading the weights into weight buffers ① and zero-initializing the cluster memories ②. Events are then broadcast to all slices and clusters ③. Concurrently, spikes produced by the slices are streamed to L2 memory. During tile computation on SNE, the FC enters a low-power state. Post-tile execution, SNE sends an interrupt to the FC ④, which re-configures SNE for a new tile execution (② - ④). After completing all tiles, the FC merges the spike streams into a single time-ordered stream of events. This process repeats for the next layer by loading a new set of weights and input spikes ①.

2.2 Completely Unrolled Ternary Inference Engine

The block and timing diagrams of CUTIE are shown in Figure 3a and Figure 3b. CUTIE leverages TNN memory efficiency, where weights and activations compress to less than 2 bits per symbol ⑥. Including a zero state, reduces switching activity by *turning off* network weight connections, thus lowering energy use at equivalent memory usage [17]. Compressed neural network parameters are decompressed ⑤ while streamed into the output compute units (OCUs) ③, ②, using an efficient scheme grouping values into sets of five, stored in 8 bits to achieve a 1.6 bits per symbol compression rate [18].

CUTIE supports ternary weight convolutional layers with optional pooling, using two thresholds for ternarizing outcomes. Each layer’s weights are consecutively stored in weight memories ①. The OCUs ② with dual latch-based buffers for filter weights, facilitate output pixel computation and layer overlap ①. Simultaneous weight loading for upcoming layers and current layer computation ② - ④ optimize efficiency. The *end of inference* signal indicates to the host controller that the final class scores are available and valid, and CUTIE is ready to receive the next feature map input ⑤. The OCU’s ternary MAC unit calculates single weight and activation products, summing them into a 16-bit value. This value is passed through a threshold decider, producing 2-bit outputs, accumulated to five output values, and compressed into an 8-bit binary representing five trits, optimizing data representation.

Neural networks are executed layer-wise on CUTIE, featuring a flat design hierarchy. This allows channel-wise output feature maps computation, with a total of 96 OCUs,

amounting to 82944 ternary MAC/cycle (3×3 convolutions). Private memory for each OCU ③ buffers weights to reduce energy consumption spent on data transfers, while shared feature map storage ④ across OCUs optimizes activation data reuse. A tile buffer ⑤ creates square windows of the input feature map using sliding windows, which are then broadcast to the OCUs. This allows the processing of 24 million frames per second for the CIFAR-10 dataset (32×32 pixels), achieving throughput up to 5.4 TOP/s.

3 RISC-V Multi-core Compute Cluster

The cluster comprises eight 32-bit RISC-V cores, supporting the RV32-IMCF and XpulpNN [15] ISA extensions. These include hardware support for fast loop programming, standard and narrow-bit MAC operations, and SIMD instructions tailored for low-precision ML workloads, down to 2-bits. Each core has a private FPU, useful for near-sensor tasks, that cannot be easily quantized, like sensor data conditioning, filtering, and control tasks. The cluster features an L1 TCDM with 128 kB capacity with a banking factor of two with word-level address interleaving [19]. Up to eight concurrent memory requests with single-cycle access latency with a banking conflict probability generally lower than 10% can be handled.

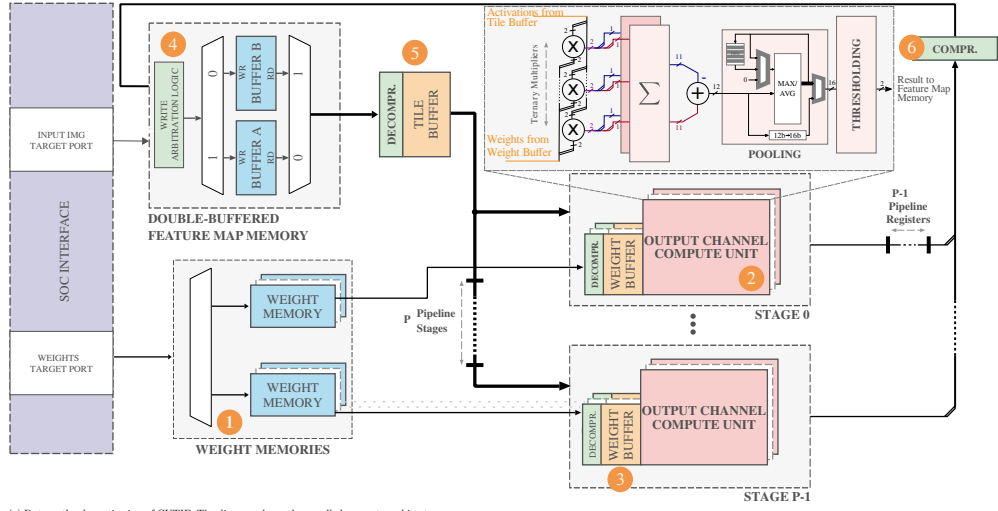
The cluster incorporates a dedicated *Hardware Synchronizer* supporting fast event management, parallel thread dispatching, and synchronization [20]. This enables fine-grained parallelism and high energy efficiency in parallel workloads. The cluster can be clock-gated with a single core granularity, reducing the dynamic power consumption when cores are synchronizing.

4 Fabric controller and I/O-DMA subsystem

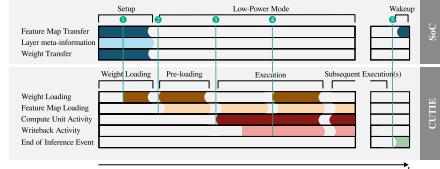
The FC, centered around a 32-bit RISC-V core, serves as the programmable control unit for the SoC. It manages the main system interconnects linked to L2 memory and the advanced peripheral bus (APB) for all peripherals. An event unit routes peripheral-generated interrupts to the interrupt controller.

The FC features four real-time counters (timers) for pulse width modulation (PWM) or time reference signals, and a power management unit, accessible via a register interface. The L2 memory is partitioned into two 32 KiB banks with core-private sections for the RISC-V executable, core stack, critical variables, and a 1 MiB shared section. The shared memory is split into eight banks and two sections: contiguous and interleaved. The interleaved region assigns every successive adjacent 4-byte word from a base address to consecutive L2 memory banks, minimizing access contention when multiple peripherals access the memory in parallel.

The FC hosts various microcontroller I/O peripherals: $4 \times$ I²C, $4 \times$ QSPI, $4 \times$ UART, $44 \times$ general purpose input outputs (GPIOs), all connected to the I/O subsystem, the μ DMA[21] for autonomous data transfers between peripherals and L2 memory without core involvement. The data transfer paths are visualized in Figure 1. The I/O subsystem streamlines external and internal data interactions via logarithmic interconnects and clock domain crossings (CDCs) directly to the accelerator domain or



(a) Data-path schematic view of CUTIE. The diagram shows the unrolled compute architecture, encoding/decoding blocks, storage for weights and feature maps, and the tile buffer module. The accelerator orchestrates data flow by buffering entire feature map windows in the tile buffer and then computing the convolution with pre-loaded weights in the OCUs. Eventually, the result is stored back in the feature map memory.



(b) Scheduling diagram of the SoC (top) and CUTIE (bottom) for continuous inference. The setup and weight loading are only needed for the first layer. Subsequent loading phases are overlapped with the execution phase to hide the latency of scheduling a new layer to a single cycle. Once CUTIE starts its operation, the host is put into low-power mode since it carries all layer information inside its dedicated memory locations.

Fig. 3: Completely Unrolled Ternary Inference Engine (CUTIE) Architecture and Timing

L2 memory. The cluster's DMA engine transfers data from L2 memory to the cluster's shared L1 memory.

The μ DMA integrates two advanced interfaces towards low-power imaging sensors: an 8-bit parallel bus (CPI) operating at speeds up to 20 MHz and a custom DVSI for event cameras using the synchronous address-event representation (SAER) protocol. The CPI allows connecting Kraken to ultra-low power 60/120FPS VGA/QVGA imagers, e.g., *Himax HM01B0*, consuming approximately 2 mW [22]. Data acquired through CPI can be efficiently buffered into L2 memory or streamed directly to CUTIE with a latency of a few clock cycles.

The DVSI, operating at clock frequencies of 5-20MHz, connects Kraken to event cameras such as the *IniVation DVS132* [23], a low-resolution ultra-low power imager, consuming 250 μ W in low-activity conditions. Leveraging the dedicated I/O subsystem and the μ DMA, events from the DVSI can be continuously streamed to SNE or L2 memory at low latency.

The chip features 76 digital pads, 44 of which are peripheral-agnostic. The I/O peripheral pin to I/O pad mapping is configurable via a memory-mapped register interface. Each peripheral can generate interrupts depending on data transmission events; events can then be routed to the interrupt controller of the FC.

5 Kraken at work

5.1 Event-frame acquisition

Kraken’s unique capability is acquiring data from event-based sensors at very high throughput. We characterized Kraken’s I/O subsystem latency for event acquisition from the DVS. In our experimental setup, Kraken was connected to an IniVation DVS132. Multiple event frames were recorded during a 1 second time window, and the acquisition latency was logged using a timer. Repeated acquisitions allowed to span a wide range of event-frames ranging from 1 to 132x104/4 events, the maximum number produced by the DVS camera. These frames were ordered by the number of events to correlate the frame latency versus the number of events. The event-frame request rate was set to 300 event frames per second to not saturate the DVSI input bandwidth in case of DVS maximum activity (104x132 active events).

Figure B2 shows the proportionality between the number of events collected from the event camera and the latency to acquire such data and store them in L2 memory, demonstrating the effectiveness of such sensors in achieving high frame rates in applications such as eye-tracking.

5.2 SNN Inference for Gaze Vector Prediction

Neuromorphic computing models, like SNNs, inspired by the human brain, mimic neuron activity through discrete information spikes, reflecting the asynchronous behavior of biological neurons. These networks efficiently process temporal data and event streams, leveraging their inherent temporal representation and energy efficiency due to sparse data and computations.

Our SNN is tasked with determining gaze vector orientation and predicting the user focal area. Figure 4 shows the network consisting of four convolutional layers and one linear layer ❶. It predicts the pupil position and orientation from 5 output

neurons, each representing a parameter of an ellipse in a 2D space; the ellipse center (x , y) coordinates, minor and major axis lengths, and rotation angle. Our SNN, trained as outlined in Section 8.4 on the dataset described in Section 8.3 uses four 1 ms timesteps to determine a new pupil position.

Table C1 presents the layer-wise energy and latency breakdown of the SNN. The inference process operates in a TDM manner, partitioning each layer into spatial sub-tiles, matching the maximum capacity of SNE. The tiles are executed on SNE; intermediate results are stored in the L2 memory. Event streams from one layer are merged to form the next layer’s input, coordinated by the FC.

In our experiments, SNE operates at 0.65 V and 140 MHz, consuming 470 μ J per inference and 2.6 pJ/SOp for inputs with less than 10% average activity. A single inference takes 10.6ms, with rates of 94 inf/s for a 4ms data window and 376 inf/s for a single 1ms timestep. The initial layers, with larger spatial dimensions, require more tiles, increasing execution latency, as shown in Table C1.

We use the Intersection over Union (IoU) metric against the ground truth to evaluate our network’s precision in predicting the pupil’s position, which may give overly pessimistic results for small objects within a large frame, where the values representing the object’s center are highly quantized. We consider predictions with an IoU above 30% as valid, indicating significant overlap with the ground truth. Our network demonstrates an average precision of 64%, showing effective pupil location identification. Gaze vector calculation incurs minimal computational cost.

The comparison with recent SNN accelerator engines is presented in Table 1. SNE achieves a performance of 17.9 GSOp/s and an energy/SOp of 2.6 pJ/SOp, measured on the development board presented in Section 8.2 at 0.65 V and 140 MHz. Compared to works reporting energy efficiency metrics (Energy/SOp) and performance metrics (GSOp/s), SNE achieves $1.25\times$ lower Energy/SOp and $4.3\times$ higher throughput. With respect to works reporting only the Energy/SOp, SNE achieves $1.13\times$ lower Energy/SOp.

	SpiNNaker 2 (2021) [24]	Loihi 2 (2022) [25]	Tan et. al (2023) [26]		Wang et. al (2023) [27]		SNE	
Technology	22nm	14nm EUV	40nm CMOS		40nm CMOS		22nm FDX	
Neuron Type	Prog	Prog	LIF		–		ALIF	
Neurons	250	1000000	–		256 $\times$ 12		8193	
Synapses	20K	1200000	–		64K $\times$ 12		18432	
Network Type	SNN CNN	SNN	SNN CNN		SNN		SNN	
Weight Precision [bits]	Var	1-9	8		5		4	
Supply Voltage [V]	0.45 - 0.9	0.75 1.10	0.65	0.9	0.5	0.9	0.52	0.65
Frequency [MHz]	200	1000	105	259	12.5	100	85	
Power [mW]	–	–	3.12	12.14	0.5	10.26	17	46
Performance [GSOp/s]	–	–	1.70	4.15	–		11	17.9
Operation Energy [pJ/SOp]	1700	7.8 –	1.89	2.96	1.71	4.38	1.5	2.6

Table 1: State-of-the-Art comparison of SNN accelerator engines.

5.3 TNN Inference for Object Detection

TNNs, utilizing ternary weights (-1, 0, +1), enable significant model compression and energy-efficient computation, suitable for edge-based keyword spotting, voice control, and low-power image classification on edge devices with constrained memory [28].

TNNs enhance devices’ capabilities to process larger, more complex models through improved computational efficiency and reduced power and memory footprint, as supported by previous works [28–30]. Ternary weight binary input CNNs show high accuracy in embedded systems [29] and allow transformer models to be quantized while maintaining floating-point comparable accuracy [30].

The CUTIE-deployed object classification network ③, based on Scherer et al.’s work, is shown in Figure 4. It combines pooling layers with convolutions to streamline data processing and minimize memory transfers, following training methods outlined in Section 8.5. It achieves a peak energy efficiency of 590 TOP/s/W and latency of 0.2 ms per inference on the CIFAR-10 dataset, as shown in Table C2. With Magnitude-Inverse quantization, CUTIE achieves 88% accuracy on CIFAR-10 with an average efficiency of 345 TOP/s/W. At 0.65 V and 66 MHz, CUTIE’s energy consumption averages 2.72 μ J per inference, demonstrating efficient high-throughput object detection in AR/XR settings.

Table 2 compares CUTIE with recent TNN and binary neural network (BNN) accelerator engines. Knag et al. and Ryu et al. show higher throughput in their fastest configuration, but CUTIE maintains higher throughput than all other works at low-voltage operating points and achieves the highest accuracy of 88.6% on the CIFAR-10 dataset. CUTIE consumes up to 25% less energy per inference compared to the state-of-the-art (SOA) architectures by leveraging reduced switching activity in TNNs. Regarding energy efficiency, CUTIE outperforms binary accelerators by a factor of 12 $\times$ and 1.68 $\times$ in peak efficiency mode, respectively, surpassing the efficiency achieved at more advanced technology nodes. Cheon et al.’s hybrid accelerator exhibits high peak efficiency through its use of compute-in-memory (CIM), which minimizes data movement and suits highly parallel workloads like ML algorithms. However, CUTIE’s throughput exceeds theirs by 6.4 $\times$, making it Pareto optimal without compromising performance for improved power consumption dependency tradeoff.

5.4 Linear Regression for Eye Movement Tracking

The characterization of the RISC-V cluster targets eye movement tracking, using EOG data from electrodes near the eyes, as shown in Figure D4. It tracks vertical, horizontal, diagonal, and circular eye movements, with EOG signals recorded at a rate of 1000 samples per second.

We developed a neural network (NN) for this task to process this data, evaluated on the parallel ultra low power (PULP) cluster. The network ② comprises three linear layers with 32 inputs and outputs, as depicted in Figure 4. It was trained employing the *smooth L1* loss function, which merges L1 and L2 losses for improved outlier resistance and balanced error reduction [34]. The NN processes 16 EOG samples to predict eye movement, outputting 16 timesteps for both horizontal (x) and vertical (y)

	Knag et. al (2021) [31]	Ryu et. al (2023) [32]		Cheon et. al (2023) [33]		CUTIE	
Technology	10nm CMOS	28nm CMOS		28nm CMOS		22nm FDX	
Computation Method	digital	digital		digital		digital	
Weight Precision	binary	binary		binary		ternary	
Activation Precision	binary	binary		ternary		ternary	
Dataset	CIFAR-10	CIFAR-10		CIFAR-10		CIFAR-10	
Accuracy	86%	–		88.5%		88.6%	
Energy/inf [μJ]	3.0	3.2		–		2.72	
Core Area [mm^2]	2.1	0.4		0.04		2.96	
Throughput [TOP/s]	101	3.4	163	2.5	–	16	56
Peak Efficiency [TOP/s/W]	86.7	617	269	2941	943	1036	446
Supply Voltage [V]	0.9	0.37	0.75	0.6	1.0	0.5	0.9

Table 2: State-of-the-Art comparison of TNN accelerator engines.

coordinates. By averaging these outputs, we achieve a smooth, accurate eye movement representation. The network has been dynamically quantized to 8-bit integers [35].

The network’s performance was assessed using the MAE, reflecting the normalized average difference between predicted and actual gaze coordinates. The results indicate an MAE of 0.246 for the x coordinate and 0.095 for the y coordinate, as detailed in Figure C3 across 150 samples.

We measured the PULP cluster’s efficiency during inference, noting an averaged power consumption of 34.3 mW during inference at 0.65 V and 160 MHz. The inference latency is 0.35 ms, with 12.01 μ J energy consumption per inference. The cluster’s peak computational efficiency reaches 1.6 TOPS/W, higher than other SOA PULP architectures [36], as seen in Table 3. Although the PULP cluster is not equipped with any accelerator engine, it achieves 8-bit peak efficiency comparable to SOA architectures [36–38] and outperforms peak performance SOA PULP systems [36] by a factor of $2.7\times$.

5.5 Fusion of multi-modal data

In AI’s dynamic field, integrating different sensory modalities is key for developing efficient and robust systems. Joshi et al. and Abrash et al. highlight how multimodality improves robustness, transparency, accuracy, and functionality in neural networks and AR devices, respectively [41, 42]. Inspired by the human brain’s multi-level information processing, we leverage this approach by combining multiple modalities to enhance energy efficiency and context understanding for more accurate and efficient application performance.

We evaluate the FC’s performance on a Kalman filter designed to combine data from SNE and the PULP cluster for high-speed and robust user gaze determination.

	VEGA (2022) [36]	DIANA (2023) [39]	MARSELLUS (2023) [38]	Chang et. al (2022) [37]	Zhang et al. (2022) [40]	SIRACUSA (2023) [7]	KRAKEN PULP Cluster
Technology	22nm FDX	28nm CMOS	28nm CMOS	22nm FDX	22nm	16nm FinFET	22nm FDX
Application	IoT GP + DNN SoC	AI-IoT SoC	IoT GP + AI-IoT SoC	Edge AI ASIC	GP + DNN Robot SoC	XR SoC	GP + DNN SoC
Cores	10× RV32IMCFXpulp + Convolution Engine	1× RV32IMCFXpulp + digital NE + AIMC SRAM	1× RV32IMCFXpulp + 16× RV32IMCFXpulpnn + NE	1× 1× Cortex-M3 + MAC Array	Cortex-M33 + NPVU	256 KiB (L1) + 2 MiB (L2) + 4 MiB (SRAM)	1× RV32IMCFXpulp + +8× RV32IMCFXpulpnn
On-chip SRAM	128 KiB (L1) 1600 KiB (L2)	896 KiB(in/out+weights)	128 KiB (L1) 1024 KiB (L2)	768 KiB	1428 KiB	4 MiB MRAM (L1)	128 KiB (L1) 1 MiB (L2)
INT Precision	8, 16, 32	2, 4, 8, 16, 32	2, 4, 8, 16, 32 (RISC-V) 2-8 (RBE)	1, 4, 8 (MAC Array) 32 (Cortex-M3)	16	2, 4, 8, 16, 32 (RISC-V) 2-8b W, 8b in/out (N-EUREKA)	2, 4, 8, 16, 32
Supply Voltage [V]	0.5 - 0.8	0.5 - 0.9	0.5 - 0.8	0.9	0.5 - 1.0	0.65 - 0.8	0.5 - 0.8
Max Frequency [MHz]	450	320	420	200	190	360	330
Power Range	1.7 uW - 49.4 mW	10mW - 129 mW	12.8 mW - 123 mW	2.6 mW - 131 mW	468 uW - 158 mW	151 mW - 332 mW	2mW - 300 mW
8-bit Peak Performance [GOP/s]	32.2	140 (NE) ^a	90 ^a	—	146 ^a (16b in/W)	698 ^a	21.5
8-bit Peak Efficiency [TOP/s/W]	1.3 ^a	2.07 ^a	1.8 ^a	0.94 ^a	0.7 ^a (16b in/W)	2.68 ^a	0.38
Best Peak Performance [GOP/s]	32.2	140 (NE) 29500 (AIMC-SRAM)	637 ^a (2b in/W)	—	146 ^a (16b in/W)	19500 ^a (8b in, 2b W)	87 (2b in/W)
Best Peak Efficiency [TOP/s/W]	1.3 ^a (8b in/W)	4.1 (NE, 2b in/W) 600 (AIMC-SRAM)	12.4 ^a (2b in/W)	60.64 ^a (1b in/W)	0.7 ^a (16b in/W)	8.84 ^a (8b in, 2b W)	1.6 (2b in/W)

^a Value measured for the accelerator engine.

Table 3: State-of-the Art comparison of edge and near-sensor computing systems.

This process can potentially trigger an object detection task as detailed in Section 2.2. Figure 4 illustrates the multimodal perception pipeline within the Kraken SoC, starting with streaming DVS data to SNE and EOG data to the PULP cluster. The FC combines these predictions for precise focal point and eye movement classification identifying movements as *saccade* or *fixation*, with the latter defined by a lack of significant change within a 100 ms window [43]. *Fixation* activates object classification on CUTIE within a 32×32 pixel window centered around the gaze point.

The Kalman filter code, presented in Appendix E, estimates the gaze position with x and y coordinate state variables, updating continuously in an infinite loop. It predicts new positions every N timestep, with N corresponding to the sensor rates' least common multiple. Simulated data for the filter inputs mimic a random x and y trajectory, incorporating prediction errors from both EOG regression and DVS gaze vector estimation. The FC consumes 7.1 mW during filter operations at 0.65 V and 50 MHz, completing updates in 33 μ s.

and mixed-precision computations via a status-based RISC-V ISA extension. The FC, coordinates these subsystems, delegating compute-heavy tasks to the accelerators and serving as the main programmable control unit with a 32-bit RISC-V core.

We demonstrate a multimodal perception application where each accelerator is assigned to a specific task on three modalities of data streams. DVS data is processed by SNE to predict the user’s gaze vector and point, while EOG data is analyzed by the PULP cluster using an 8-bit quantized convolutional neural network (CNN) to monitor eye movements. These predictions are combined via a Kalman filter on the FC to refine the gaze point and identify eye movements (e.g., saccade or fixation), facilitating the cropping of a *region of interest* around the gaze point for object classification by CUTIE on a reduced window. This division enables efficient, fast gaze and eye movement predictions within a 46.3 mW power envelope and an 11 ms estimation time from input to prediction. The integration of SNE and PULP cluster outputs through a Kalman filter on the FC boosts system robustness, while CUTIE’s object classification adds functional depth, making Kraken a robust and versatile system for complex foveal vision pipelines.

The SNE, in comparison with recent SNN accelerator engines, shows 4.3x higher throughput and 1.13x lower Energy/SOp [26]. CUTIE achieves 6.4x higher throughput and 1.68x higher energy efficiency [31–33]. The PULP cluster, even without any accelerator engine, achieves a peak efficiency comparable to SOA architectures [36].

In conclusion, Kraken demonstrates a novel approach to multi-modal perception by leveraging the strengths of heterogeneous SoC design. The system’s ability to interact with a variety of sensors and dedicated accelerator engines to ensure optimal energy efficiency, high throughput, and robust perception.

7 Acknowledgements

This work was supported by EU project 101120726 — dAIEDGE — HORIZON-CL4-2022-HUMAN-02 and Innosuisse project Number: 103.364 IP-ICT. The authors would like to thank IniVation for kindly lending us a DVS camera.

8 Methods

8.1 Physical implementation

In this section, we provide the physical implementation details of the Kraken chip. Table 4 reports the salient characteristics.

The Kraken chip has been implemented in the GlobalFoundries 22nm FDX technology process. We used the *Synopsys Design Compiler 2020.09* tool to synthesize the design. The design’s physical implementation has been performed with the *Cadence Innovus Implementation System 19.1* tool. We used 8T, 20, 24, 28, L, and SL voltage threshold cells for the proposed design. The setup time has been optimized for the slow (SSG) 0.59 V -40°C process corner, while the hold time has been optimized for the (FFG) 0.72 V 125°C and -40°C process corners.

The FC domain has been constrained to operate at 160 MHz in the *worst-case* process corner. At the same process corner, the SNE and Cluster domains have been

Technology	GlobalFoundries 22nm FDX
Chip area	9mm ²
L2 memory	1MiB (SRAM) + 64kiB
L1 Memory	128KiB (SRAM)
VDD Range	0.5V - 0.9V
Cluster Frequency	380MHz
EHWPE Frequency	330MHz
FC Frequency	380MHz
Power Range	2mW-200mW

Table 4: Physical implementation details

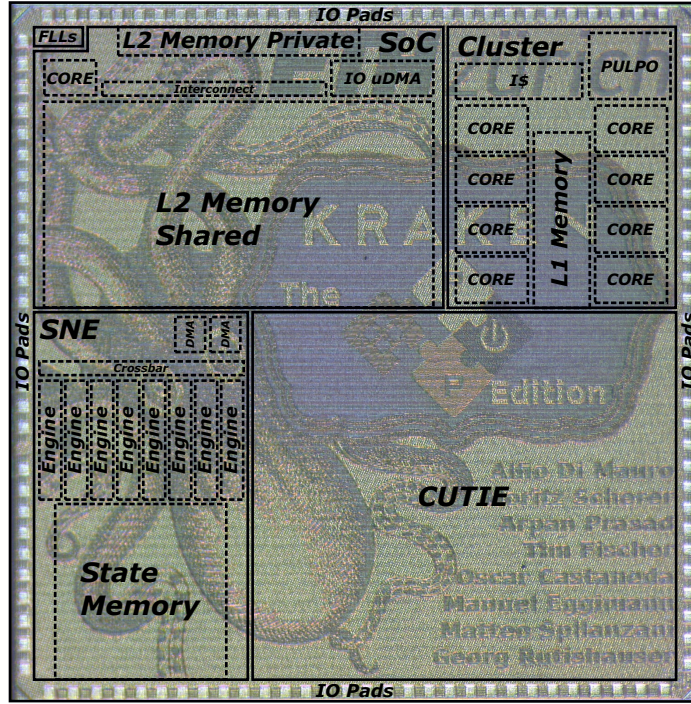


Fig. 5: Kraken micrograph

constrained to operate at a maximum clock frequency of 200 MHz. Figure 5 shows the chip micrographs where the different power domains have been highlighted. Moreover, the figure also shows the memory and core area occupation.

As Section 2 mentions, the PULP cluster, SNE, and CUTIE domains can be independently power-switched to reduce leakage consumption. This functionality has been implemented by exploiting dedicated power switching cells provided with the GlobalFoundries 22nm process development kit (PDK). Each switchable domain hosts

power switched, placed on the standard cell's row, and supplied from an always-on power supply, allowing the domain-specific cells to be disconnected from the power supply. Only the power switch leakage current contributes to the static consumption of a switched-off power domain. Power switches are controlled by the power management unit (PMU) hosted in the FC domain.

8.2 Kraken Development Board

The Kraken chip was soldered onto an evaluation board for testing and debugging purposes. The evaluation board incorporates essential ports, including joint test action group (JTAG) and UART, accessible via a USB-C interface, which ensures standardized access for developers. The USB-C connector enables data transfer and provides a 5V power rail from which all power supplies of the Kraken board components are derived.

Kraken utilizes three power supplies, each powered by individual runtime-configurable buck converters, enabling dynamic voltage and frequency scaling (DVFS) under application control. The connection to the DVS camera is established through an 80-pin low-profile board-to-board connector, which also supplies the necessary 1.2 V and 5 V rails to the camera board. External off-chip memory is available in two forms: a combined HyperFlash/HyperRAM chip and a quad-SPI flash memory chip. These memory chips serve multiple purposes, including storing application code for standalone booting. Arduino headers and a CPI ribbon connector are included to enhance connectivity, with level shifters between every off-chip connector and Kraken's I/O pins [44]. A photograph of the board is shown in Figure 6.

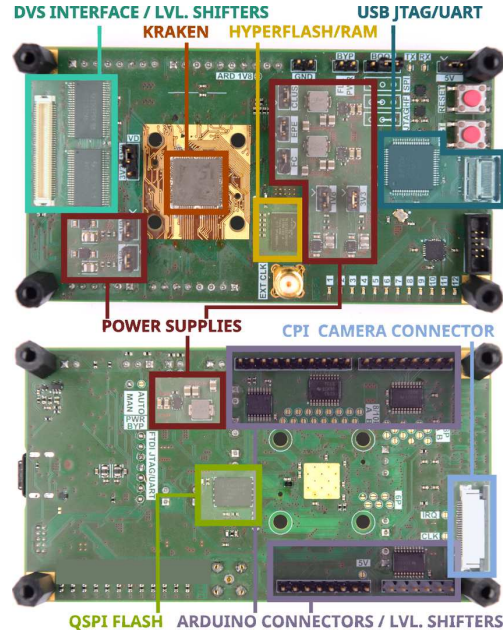


Fig. 6: Kraken evaluation board

8.3 Multi-modal dataset

We exploit the open-source dataset provided by Angelopoulos et al.[45], which is accessible on GitHub¹. On such a data set, we aim to tackle the regression task of identifying the pupil’s position. One challenge we encountered was the absence of ground truth labels for the pupil area within the dataset. To overcome this limitation, we implemented a post-processing strategy derived from the data processing pipeline already provided by the authors. Specifically, we have used their data pre-processing tools to read the event and frame data into data frames for further use. However, we have developed a custom pupil detection algorithm, documented in our open-source repository on GitHub². Our pipeline for determining ground truth labels is detailed in Figure 7; an example ground truth label in red is indicated on the right-hand side of Figure 7.

Our data processing pipeline begins with image smoothing and subsequent Canny edge detection [46]. Following this initial processing, we attempt to fit an ellipse to the detected contours of the edges. In cases where no ellipse can be successfully fitted, we introduce a refinement step incorporating more robust smoothing and image filtering techniques. Conversely, when multiple ellipses are detected within an image, we apply size and aspect ratio filters to narrow the selection to the most pertinent ellipse.

In situations where all previous methods fall short, we have implemented an interactive window. This feature allows the user to designate a region of interest (ROI) manually, and the corresponding ellipse within this region will be selected as the ground truth label. Additionally, the size and aspect ratio filters are updated based on user input.

The ground truth label holds the ellipse parameters denoted as $E = (x_0, y_0, a, b, \theta)$, where (x_0, y_0) is the center of the ellipse, (a, b) are the minor and major axes, respectively, and θ is the angle of rotation of the ellipse in two-dimensional space.

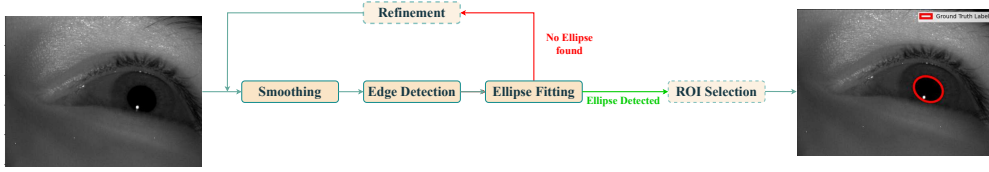


Fig. 7: Pupil Detection Pipeline.

8.4 Training and deployment of SNNs on SNE

The SNN deployed on SNE has been trained on a single subject extracted from the dataset reported in 8.3, on the event camera modality. The input event frames have been subsampled by $2\times$ to reduce the input resolution of the network. Subsampling the events collected by an event camera is a lightweight operation that can be easily performed at runtime by the FC. The event camera receives the events in order,

¹https://github.com/aangelopoulos/event_based_gaze_tracking

²placeholder

and a new sub-sampled event can be created by performing a logical OR operation between four neighboring events. The sub-sample event frame resolution used in our experiment is 130×173 , each event has two distinct polarities, that are mapped on two SNN input channels, and the input of the network is then a sparse event frame with dimensions $130 \times 173 \times 2$. The training has been performed offline for $N = 50$ epochs, in a framework derived from the spatio-temporal back propagation (STBP) supervised training framework presented in [47], based on the `Pytorch 1.0` library. STBP is a general strategy to overcome the limitation originating from the LIF spiking neuron by using a gradient surrogate around the non-differentiable point of the neuron membrane potential function. These mathematical artifacts practically allow using commonly adopted gradient-descent-based optimization techniques to train a neural network and feed-forward SNNs with convolutional or linear synaptic connections. On the Kraken chip, SNE requires the network parameter to be represented with signed 4-bit integers. We performed Quantization-aware Training (QAT) by training the network with single precision floating point parameters for $N = 25$ epochs, until convergence. From epoch $N = 26$, the weights are first modified with a gradient-descent algorithm at each training iteration then quantized and binned into $2^4 - 1$ bins, still represented by floating point values.

The trained network is then exported as an ONNX representation. The SNE accelerator deployment is performed using a dedicated Python-based deployment flow that adopts a template-based mechanism to compose the C source code that Kraken and SNE can execute. The inference is executed layer-wise on SNE using matrix tiling. The weights and inputs of the network are stored in Kraken’s L2 memory. The pre-processing of the events to generate the tiles is executed by the FC.

8.5 Training and deployment of TNNs on CUTIE

Quantized Neural Networks (QNNs) exploit low-precision data types to reduce the computational cost of inference [48]. The extreme edge cases of quantization are BNNs [49] and TNNs [50]. In BNNs, the network parameters are mapped to values in $\{-1, 1\}$ only, while in TNNs we also take the 0 value into account and map all parameters to the set $\{-1, 0, 1\}$. This allows for extremely hardware-friendly design [51] and considerable improvements in inference time and energy efficiency [52]. Nonetheless, this comes at the price of a non-negligible loss of accuracy [53].

To mitigate this impact, SOA algorithms have been developed. In general, there are two main classes of algorithms for quantization: *Post-Training Quantization (PTQ)* and *QAT* [54]. PTQ requires no re-training or labeled data and is a lightweight push-button approach to quantization. In most cases, PTQ is sufficient for achieving 8-bit quantization with close to floating-point accuracy. QAT requires fine-tuning and access to labeled training data but enables lower bit quantization with competitive results. Most QAT methods are derived from the straight-through estimator (STE) [55]. The STE allows backpropagation of non-differentiable operations by propagating an *approximated* gradient, i.e. we define the gradient of the quantized tensor as a clipped identity function as given in Equation 1. x denotes the quantized tensor, $[a, b]$ is the clipping interval when mapping the full precision value to n quantization levels spaced

by a uniform distance $\epsilon = \frac{b-a}{n-1}$.

$$\frac{\partial Q(x, a, b, n)}{\partial x} = \mathbb{1}_{a \leq x \leq b} \quad (1)$$

The main challenge lies in finding optimal clipping bounds $\{a, b\}$ for which several methods have been proposed such as *Learned Step Size Quantization (LSQ)* [56] and *Trained Quantization Thresholds (TQT)* [57]. These methods aim to optimize the quantization step ϵ or the logarithm of the clipping interval $\log_2(b - a)$, respectively. On the other hand, some methods forgo the STE like Incremental Network Quantization (INQ) [58]. This procedure is an iterative quantization scheme where a fraction of the weights are quantized and frozen while full-precision weights are retrained. This process is repeated until all weights have been quantized. INQ achieves higher compression rates with less accuracy loss at the cost of increased training time.

For this work, we follow the approach of [52] to offload the network to the CUTIE accelerator. Using the *QuantLib* training framework that supports INQ and STE quantization, we train the network for CUTIE.

8.6 Training and Deployment of DNNs on the PULP cluster

The quantized neural network (QNN) deployed on the PULP cluster is designed to predict eye movement trajectories, comprised of three linear layers, and processes 2-dimensional input and output. The network was trained on 880 samples of EOG data from a single subject. The smooth L1 loss function is employed during training, providing a balance between L1 and L2 loss functions, minimizing error while being robust to outliers. The network is trained in full precision with the *QuantLib* training framework. After training, we apply dynamic post-training quantization and export the model in ONNX format. The exported ONNX model is then fed into DORY. DORY processes the ONNX model, generating the necessary C code for deploying the QNN on the PULP platform. It ensures efficient mapping of the neural network’s computational graph to the PULP architecture, optimizing for performance, handling the matrix tiling, and organizing memory usage. Finally, the model is deployed on the PULP cluster. The EOG sensor comes with an analog-to-digital converter (ADC) and can be connected via the serial peripheral interface (SPI) to the Kraken SoC. The data is then streamed to the PULP cluster, which performs inference after seeing 16 samples on both channels of the differential electrodes, i.e. a total of 32 samples.

8.7 Data Availability

Experiment data and processing scripts are available open-source³.

³placeholder

Appendix A Sparse Neural Engine

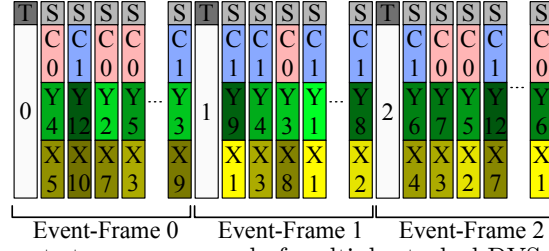


Fig. A1: Sample event stream composed of multiple stacked DVS event-frames, stored in an L2 event buffer and consumed by the SNE

Appendix B Event-frame acquisition

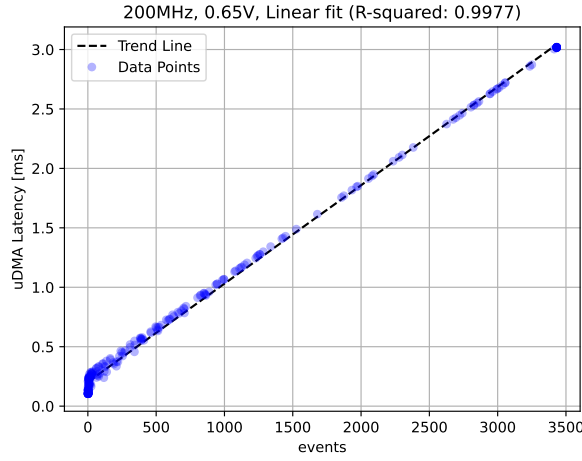


Fig. B2: Time proportionality between the number of events acquired from the event-based camera, and the latency of the event pre-processing pipeline, which includes the uDMA DVS peripheral, and a software pre-processing of the events on the FC

Layer	Tiles	Tile avg. power [mW]	Layer avg. energy [uJ]	Layer avg. latency [ms]
Conv 0	40	41	199	4.8
Conv 1	10	47	140	3
Conv 2	3	47	99	2.1
Conv 3	1	46	18	0.39
Linear 0	1	50	15	0.30
Total			470	10.6

Table C1: Layer-wise breakdown of the tile average power, energy and latency for executing SNN inference on SNE.

Layer	Input Dim	Energy eff. [TOp/s/W]	Layer avg. energy [nJ]	Layer avg. latency [μ s]
Conv 0	$126 \times 32 \times 32$	590	503	54.3
Conv 1	$128 \times 32 \times 32$	380	795	55.2
FusedConv 0	$128 \times 32 \times 32$	370	816	55.2
Conv 2	$128 \times 16 \times 16$	310	243	13.8
FusedConv 1	$128 \times 16 \times 16$	350	216	13.8
Conv 3	$128 \times 8 \times 8$	280	68	3.5
FusedConv 2	$128 \times 8 \times 8$	270	70	3.5
Conv 4	$128 \times 4 \times 4$	160	29	0.9
Total				200

Table C2: Inference results on the CIFAR-10 test dataset on CUTIE at 66 MHz and 0.65 V.

Appendix C Neural Network Architectures and Additional Results

C.1 Detailed Results of the Spiking Neural Network on SNE

C.2 Detailed Results of the Ternary Neural Network on CUTIE

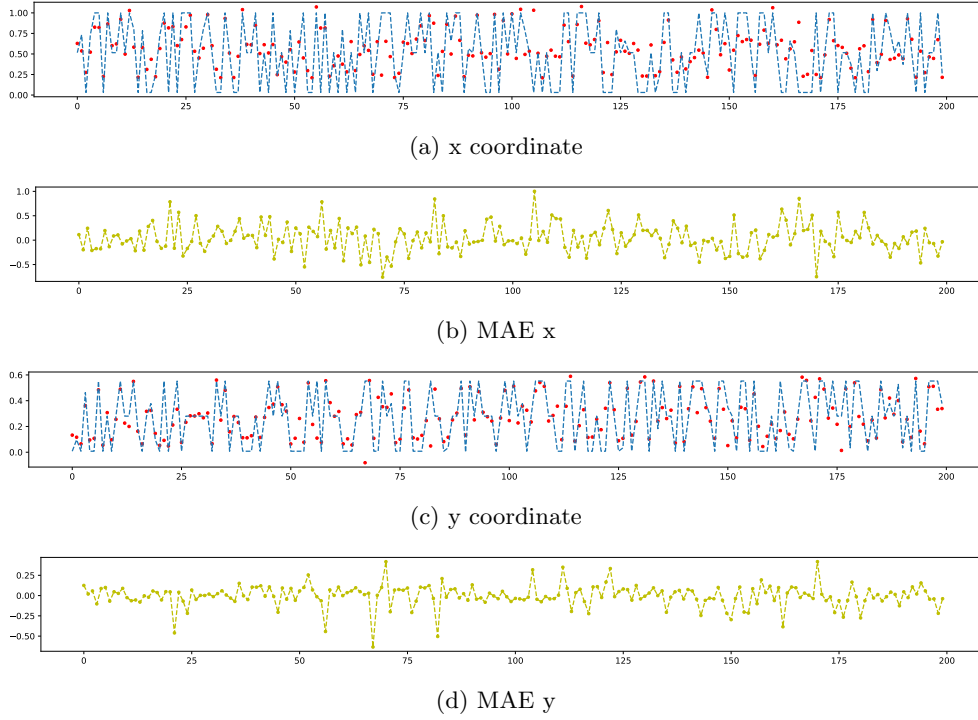


Fig. C3: Sample coordinate regression for 150 samples based on EOG traces.

Appendix D Experimental setup of the EOG dataset collection

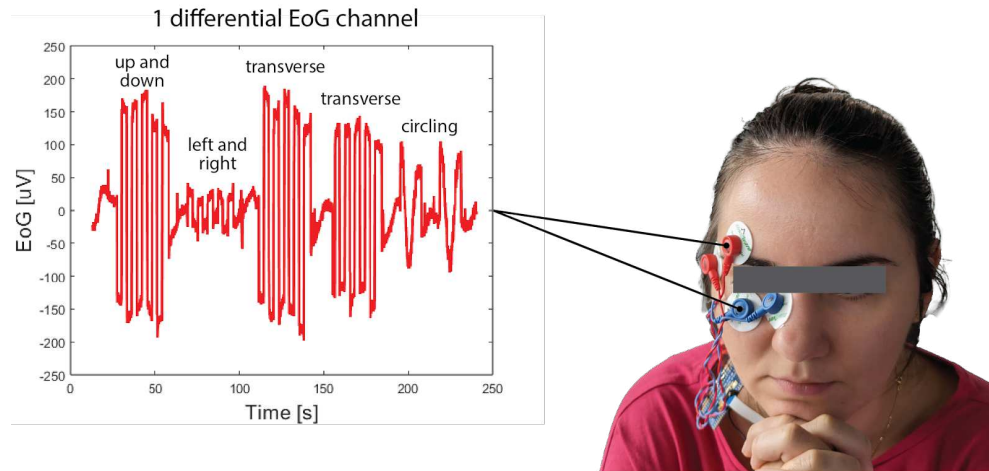


Fig. D4: Demonstration of EOG Measurement Technique and Signal Output. This image displays the placement of wet electrodes on a subject's face near the eyes to capture EOG data. The graph represents the differential EOG channel output, capturing the eye's up-down, left-right, and circular movements and transverse movements across the visual field during a dot-following exercise.

Appendix E Pseudo-code implementation of the Kalman Filter

Algorithm 1 2D Kalman Filter update and simulation

```

Structure KalmanFilter2D
  Array[2] pos – position
  Matrix[2][2] P – covariance
  Double Q – process noise
procedure UPDATE(KalmanFilter2D kalman, z[2],  $\Delta t$ , R)
  Input KalmanFilter2D kalman, the Kalman Filter object
  Input z[2], new measurement data
  Input  $\Delta t$ , the time interval
  Input R, measurement noise
  // Prediction step
   $\hat{x} \leftarrow \text{kalman.pos}$ 
   $\hat{P} \leftarrow P + Q \cdot I$  ▷ Assuming Q is added to the diagonal of P
  // Compute Kalman Gain
   $S \leftarrow \hat{P} + R$  ▷ Innovation covariance
   $K \leftarrow \hat{P} \cdot S^{-1}$  ▷ Kalman gain
  // Update step
   $y \leftarrow z - \hat{x}$  ▷ Innovation
   $\text{kalman.pos} \leftarrow \hat{x} + K \cdot y$ 
   $\text{kalman.P} \leftarrow (I - K) \cdot \hat{P}$ 
end procedure

procedure MAIN
  Initialize KalmanFilter2D instance kalman with pos = 0, initial P, and Q
  Define true motion parameters:  $x_c, y_c, r, \theta$ 
  Initialize true positions ( $x, y$ )
  // Simulated true pupil positions with a circular movement pattern
  for  $n \leftarrow 0$  to N do
     $x \leftarrow x_c + r \cdot \cos(n \cdot \theta)$ 
     $y \leftarrow y_c + r \cdot \sin(n \cdot \theta)$ 
  end for
  // Simulated measurements from neural networks at 30FPS and 200FPS
  for  $n \leftarrow 0$  to N do
     $z_{x,30fps} \leftarrow x + \varphi(0, 0.5)$ 
     $z_{y,30fps} \leftarrow y + \varphi(0, 0.5)$ 
     $z_{x,200fps} \leftarrow x + \varphi(0, 1.5)$ 
     $z_{y,200fps} \leftarrow y + \varphi(0, 1.5)$ 
  end for
  Set  $\Delta t_{30fps}$  and  $\Delta t_{200fps}$ 
  for  $n \leftarrow 0$  to N do
    if  $n \bmod N/20 = 0$  then
       $z_{\varphi,30fps} \leftarrow \{z_{x,30fps}, z_{y,30fps}\}$ 
      UPDATE(kalman,  $z_{\varphi,30fps}$ ,  $\Delta t_{30fps}$ , 0.1)
    end if
     $z_{\varphi,200fps} \leftarrow \{z_{x,200fps}, z_{y,200fps}\}$ 
    UPDATE(kalman,  $z_{\varphi,200fps}$ ,  $\Delta t_{200fps}$ , 0.5)
  end for
   $\hat{x} \leftarrow \text{kalman.pos}[0]$ 
   $\hat{y} \leftarrow \text{kalman.pos}[1]$ 
end procedure

```

References

- [1] Samikwa, E., Di Maio, A. & Braun, T. Adaptive early exit of computation for energy-efficient and low-latency machine learning over iot networks. *IEEE* **0**, 200–206 (2022).
- [2] Zoghalmi, F., Kaden, M., Villmann, T., Schneider, G. & Heinrich, H. Ai-based multi sensor fusion for smart decision making: A bi-functional system for single sensor evaluation in a classification task. *Sensors* **21** (2021). URL <https://www.mdpi.com/1424-8220/21/13/4405>.
- [3] Chitta, K. *et al.* Transfuser: Imitation with transformer-based sensor fusion for autonomous driving (2022). [2205.15997](https://arxiv.org/abs/2205.15997).
- [4] Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N. & Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.* **22** (2021).
- [5] Su, W., Li, L., Liu, F., He, M. & Liang, X. Ai on the edge: A comprehensive review. *Artif. Intell. Rev.* **55**, 6125–6183 (2022). URL <https://doi.org/10.1007/s10462-022-10141-4>.
- [6] Li, E., Zeng, L., Zhou, Z. & Chen, X. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications* **19**, 447–457 (2020).
- [7] Prasad, A. S. *et al.* Siracusa: A 16 nm heterogenous risc-v soc for extended reality with at-mram neural engine (2023). [2312.14750](https://arxiv.org/abs/2312.14750).
- [8] Samikwa, E., Di Maio, A. & Braun, T. Guo, Peng and Ma, Hong and Chen, Ruizhi and Li, Pin and Xie, Shaolin and Wang, Donglin. *IEEE* **0**, 51–513 (2018).
- [9] Akopyan, F. *et al.* Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **34**, 1537–1557 (2015).
- [10] Davies, M. *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
- [11] Li, C., Longinotti, L., Corradi, F. & Delbruck, T. A 132 by 104 10 μ m-pixel 250 μ w 1kefps dynamic vision sensor with pixel-parallel noise and spatial redundancy suppression C216–C217 (2019). 33rd Symposium on VLSI Circuits, VLSI Circuits 2019 ; Conference date: 09-06-2019 Through 14-06-2019.
- [12] Samikwa, E., Di Maio, A. & Braun, T. Conti, francesco and rossi, davide and pullini, antonio and loi, igor and benini, luca. *IEEE* **0**, 1–6 (2014).

- [13] Gautschi, M. *et al.* Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices. *IEEE TVLSI* **25**, 2700–2713 (2017).
- [14] Pullini, A., Rossi, D., Haugou, G. & Benini, L. μ dma: An autonomous i/o subsystem for iot end-nodes. *PATMOS* 1–8 (2017).
- [15] Garofalo, A., Tagliavini, G., Conti, F., Benini, L. & Rossi, D. Xpulpnn: Enabling energy efficient and flexible inference of quantized neural networks on risc-v based iot end nodes. *IEEE Transactions on Emerging Topics in Computing* **9**, 1489–1505 (2021).
- [16] Di Mauro, A. *et al.* Sne: an energy-proportional digital accelerator for sparse event-based convolutions. *DATE* (2022). Design, Automation and Test in Europe Conference (DATE 2022); Conference Location: Online; Conference Date: March 14-23, 2022; Conference lecture held on 22 March 2022.
- [17] Na, T. Ternary output binary neural network with zero-skipping for mram-based digital in-memory computing. *IEEE Transactions on Circuits and Systems II: Express Briefs* **70**, 2655–2659 (2023).
- [18] Muller, O., Prost-Boucle, A., Bourge, A. & Pétrot, F. Efficient decompression of binary encoded balanced ternary sequences. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **27**, 1962–1966 (2019).
- [19] Capotondi, A. & Marongiu, A. On the effectiveness of openmp teams for cluster-based many-core accelerators. *HPCS* 667–674 (2016).
- [20] Glaser, F. *et al.* Energy-efficient hardware-accelerated synchronization for shared-l1-memory multiprocessor clusters. *IEEE Transactions on Parallel and Distributed Systems* **32**, 633–648 (2021).
- [21] Pullini, A., Rossi, D., Loi, I., Tagliavini, G. & Benini, L. Mr.wolf: An energy-precision scalable parallel ultra low power soc for iot edge processing. *IEEE Journal of Solid-State Circuits* **54**, 1970–1981 (2019).
- [22] Himax Technologies. Hm01b0 - always-on vision sensors. <https://www.himax.com.tw/products/cmos-image-sensor/always-on-vision-sensors/hm01b0/> (2024). Accessed: 2024-01-19.
- [23] Li, C., Longinotti, L., Corradi, F. & Delbruck, T. A 132 by 104 10um-pixel 250uw 1kefps dynamic vision sensor with pixel-parallel noise and spatial redundancy suppression. *2019 Symposium on VLSI Circuits* C216–C217 (2019).
- [24] Höppner, S. *et al.* The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing (2022). [2103.08392](#).

- [25] Human Brain Project. Presentation at human brain project. <https://flagship.kip.uni-heidelberg.de/jss/HBPM?m=displayPresentation&mI=235&mEID=8950> (2023). Accessed: 2024-01-20.
- [26] Tan, P.-Y. & Wu, C.-W. A 40-nm 1.89-pj/sop scalable convolutional spiking neural network learning core with on-chip spatiotemporal back-propagation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **31**, 1994–2007 (2023).
- [27] Wang, B. *et al.* 1.7pj/sop neuromorphic processor with integrated partial sum routers for in-network computing. *ISCAS* 1–5 (2023).
- [28] Alemdar, H., Leroy, V., Prost-Boucle, A. & Pétrot, F. Ternary neural networks for resource-efficient ai applications. *IJCNN* 2547–2554 (2017).
- [29] Yonekawa, H., Sato, S. & Nakahara, H. A ternary weight binary input convolutional neural network: Realization on the embedded processor 174–179 (2018).
- [30] Zhang, W. *et al.* Ternarybert: Distillation-aware ultra-low bit bert (2020). [2009.12812](#).
- [31] Knag, P. C. *et al.* A 617-tops/w all-digital binary neural network accelerator in 10-nm finfet cmos. *IEEE Journal of Solid-State Circuits* **56**, 1082–1092 (2021).
- [32] Ryu, S., Oh, Y. & Kim, J.-J. Binaryware: A high-performance digital hardware accelerator for binary neural networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **31**, 2137–2141 (2023).
- [33] Cheon, S., Lee, K. & Park, J. A 2941-tops/w charge-domain 10t sram compute-in-memory for ternary neural network. *IEEE Transactions on Circuits and Systems I: Regular Papers* **70**, 2085–2097 (2023).
- [34] Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks (2016). [1506.01497](#).
- [35] Sun, X. *et al.* Dynamic network quantization for efficient video inference. *ICCV* 7375–7385 (2021).
- [36] Rossi, D. *et al.* Vega: A ten-core soc for iot endnodes with dnn acceleration and cognitive wake-up from mram-based state-retentive sleep mode. *IEEE Journal of Solid-State Circuits* **57**, 127–139 (2022).
- [37] Chang, M. *et al.* A 40nm 60.64tops/w ecc-capable compute-in-memory/digital 2.25mb/768kb rram/sram system with embedded cortex m3 microprocessor for edge recommendation systems. *ISSCC* **65**, 1–3 (2022).
- [38] Conti, F. *et al.* Marsellus: A heterogeneous risc-v ai-iot end-node soc with 2–8 b dnn acceleration and 30%-boost adaptive body biasing. *IEEE Journal of*

Solid-State Circuits **59**, 128–142 (2024).

- [39] Houshmand, P. *et al.* Diana: An end-to-end hybrid digital and analog neural network soc for the edge. *IEEE Journal of Solid-State Circuits* **58**, 203–215 (2023).
- [40] Zhang, Q. *et al.* A 22nm 3.5tops/w flexible micro-robotic vision soc with 2mb emram for fully-on-chip intelligence. *IEEE Symposium on VLSI Technology* **0**, 72–73 (2022).
- [41] Joshi, G., Walambe, R. & Kotecha, K. A review on explainability in multimodal deep neural nets. *IEEE Access* **9**, 59800–59821 (2021).
- [42] Abrash, M. Creating the future: Augmented reality, the next human-machine interface. *IEDM* **0**, 1–11 (2021).
- [43] Rayner, K. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* **124**, 372–422 (1998).
- [44] Rutishauser, G. *et al.* Colibries: A milliwatts risc-v based embedded system leveraging neuromorphic and neural networks hardware accelerators for low-latency closed-loop control applications (2023). [2302.07957](https://arxiv.org/abs/2302.07957).
- [45] Angelopoulos, A. N., Martel, J. N., Kohli, A. P., Conradt, J. & Wetzstein, G. Event-based near-eye gaze tracking beyond 10,000 hz. *IEEE Transactions on Visualization and Computer Graphics* **27**, 2577–2586 (2021).
- [46] M, S. & G, N. Realization of human eye pupil detection system using canny edge detector and circular hough transform technique. *ICAAIC* 861–865 (2023).
- [47] Wu, Y., Deng, L., Li, G., Zhu, J. & Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience* **12** (2018). URL <https://www.frontiersin.org/article/10.3389/fnins.2018.00331>.
- [48] Guo, Y. A survey on methods and theories of quantized neural networks (2018). [1808.04752](https://arxiv.org/abs/1808.04752).
- [49] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. & Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1 (2016). [1602.02830](https://arxiv.org/abs/1602.02830).
- [50] Deng, L., Jiao, P., Pei, J., Wu, Z. & Li, G. Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework (2018). [1705.09283](https://arxiv.org/abs/1705.09283).
- [51] Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. Leibe, B., Matas, J., Sebe, N. & Welling, M. (eds) *Xnor-net: Imagenet classification using binary convolutional neural networks*. (eds Leibe, B., Matas, J., Sebe, N. & Welling, M.) *Computer Vision – ECCV 2016*, 525–542 (Springer International Publishing, Cham, 2016).

- [52] Scherer, M., Rutishauser, G., Cavigelli, L. & Benini, L. Cutie: Beyond petaop/s/w ternary dnn inference acceleration with better-than-binary energy efficiency. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **41**, 1020–1033 (2022).
- [53] Gholami, A. *et al.* A survey of quantization methods for efficient neural network inference (2021). [2103.13630](#).
- [54] Nagel, M. *et al.* A white paper on neural network quantization (2021). [2106.08295](#).
- [55] Bengio, Y., Léonard, N. & Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation (2013). [1308.3432](#).
- [56] Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R. & Modha, D. S. Learned step size quantization (2020). [1902.08153](#).
- [57] Jain, S. R., Gural, A., Wu, M. & Dick, C. H. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks (2020). [1903.08066](#).
- [58] Zhou, A., Yao, A., Guo, Y., Xu, L. & Chen, Y. Incremental network quantization: Towards lossless cnns with low-precision weights (2017). [1702.03044](#).