Embedded neuromorphic attention model leveraging a novel low-power heterogeneous platform

Amélie Gruel, Alfio Di Mauro, Robin Hunziker, Luca Benini, Jean Martinet, Michele Magno

Embedded neuromorphic attention model leveraging a novel low-power heterogeneous platform

Amélie Gruel[1] Alfio di Mauro[2] Robin Hunziker[2] Luca Benini[2] Jean Martinet[1] Michele Magno[2]

[1]*i3S / CNRS, Université Côte d'Azur*, Sophia Antipolis, France
[2]*Department of Information Technology and Electrical Enginering, ETH Zürich*, Zürich, Switzerland
Contact email: amelie.gruel@univ-cotedazur.fr

***Abstract*—Neuromorphic computing has been identified as an ideal candidate to exploit the potential of event-based cameras, a promising sensor for embedded computer vision. However, state-of-the-art neuromorphic models try to maximize the model performance on large platforms rather than a trade-off between memory requirements and performance. We present the first deployment of an embedded neuromorphic algorithm on Kraken, a low-power RISC-V-based SoC prototype including a neuromorphic spiking neural network (SNN) accelerator. In addition, the model employed in this paper was designed to achieve visual attention detection on event data while minimizing the neuronal populations' size and the inference latency. Experimental results show that it is possible to achieve saliency detection in event data with a delay of $32ms$, maintains classification accuracy of 84.51% and consumes only $3.85mJ$ per second of processed input data, achieving all of this while processing input data 10 times faster than real-time. This trade-off between decision latency, power consumption, accuracy, and run time significantly outperforms those achieved by previous implementations on CPU and neuromorphic hardware.**

***Index Terms*—Neuromorphic, Embedded system, Event camera, Academic platform, Visual attention**

I. Introduction

Event-based cameras, often called silicon retinas, bring an emerging vision paradigm by mimicking the biological retina. Instead of measuring the intensity of every pixel in a fixed time interval like RGB/grayscale frame-based cameras, an event-based vision sensor reports events of significant pixel intensity changes [1]. Because of the asynchronous operation principle of event-based cameras, the natural match to process this information is another emerging paradigm, namely Spiking Neural Networks (SNN) [2]. These are advantageously replacing standard Deep Neural Networks (DNN) for event-based sensors, particularly event-based cameras. Although state-of-the-art DNNs provide excellent results for vision tasks with traditional cameras, asynchronous event sequences require special handling for which SNNs are better suited. Indeed, many previous works have demonstrated the benefits of SNN and the excellent match with event-based cameras to reach both exceptionally low latency and high energy efficiency [1] — all of this, however, without achieving equivalent performance to DNNs in terms of accuracy.

Pushed by the objective of closing the gap with DNN accuracy, many previous works are not considering the limits of memory and complexity that state-of-the-art embedded neuromorphic processors pose. However, previous works show that detecting Regions of Interest (RoIs) in the original visual scene to select the corresponding Objects of Interest (OoIs — i.e., the events belonging to the detected RoIs) is a more promising approach to increase accuracy, reducing both latency and memory requirements. Many computer vision tasks (classification, object tracking, or autonomous navigation) could rely on small salient items in the global scene [3]. Only a few models genuinely take advantage of the intrinsic dynamics of SNNs and the uniqueness of event data. In particular, Renner et al. [4] make use of the mathematical model of Dynamic Neural Field [5] as a soft Winner-Takes-All (WTA) to implement saliency tracking of pre-activated objects; Gruel et al. [6] implement a similar mechanism with a significantly low number of neurons and connections.

Another drawback that limits the success of today's event-based cameras and neuromorphic computing, especially for embedded low-power systems, is that the promised low power, low latency, and energy efficiency are diminished and often compromised by the interface's high power consumption, often due to non-standard communication protocol. Although nowadays data is mainly acquired using a high-power Field Programmable Gate Array (FPGA) and a USB interface [7], novel embedded processors are being developed, providing the research community with reference platforms and resources contained to address this issue. Large-scale, neuromorphic processors like Intel Loihi [8] can simulate hundreds of thousands or even millions of spiking neurons in real-time. To achieve truly power-efficient real-time operation, such platforms must be smaller, reach even lower power consumption, and integrate the camera interface directly on board. Kraken [9] is a prominent example of such a platform: this novel RISC-V-based System on Chip (SoC) includes a heterogeneous parallel ultra-low power (PULP) processor [10] and two hardware accelerators for both spiking neural networks and frame-based convolutional neural networks. Most notably, Kraken can interface directly with event-based cameras without power-hungry FPGAs or external interface adapters.

This work was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008 and SNF 20CH21_186991. It was partially supported by the SNF Innosuisse 103.364 IP-ICT. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

This work presents a novel implementation of a lightweight, energy-efficient neuromorphic model for visual attention on an embedded device with limited memory and computational capabilities. The computational primitives of the algorithm have been deployed and executed on the Kraken platform, showing that the proposed saliency detection mechanism can be performed on the Kraken SoC platforms in $140ms$, consuming $4.73mJ$.

The main contributions of this paper are as follows:

- The computational primitives of the algorithm are evaluated on the Kraken platform and show that the proposed algorithm is amenable to resource-constrained embedded neuromorphic platforms;
- An accurate experimental evaluation demonstrates both the benefits of the proposed model and the Kraken platform;
- The deployment of [6]'s attentional model on Kraken allows to detect OoIs with a significatively low decision latency and to maintain a classification performance of 84.51%, results that are achieved by processing input data up to 10 times faster than real-time.

II. Material

A. Spiking neural networks

Also known as the third generation of neural networks, SNN differs from the previous generations by its high degree of bio-inspiration [2]. Indeed, whereas information in classical models was encoded and processed as multi-bit numbers, an SNN encodes input signals and internal network layer activations using sequences of binary spikes distributed across a fixed number of timesteps. Each neuron emits spikes depending on an internal state variable, the membrane potential, which increases whenever a spike arrives at the neuron (much like the biological membrane potential increases when the biological neuron is electrically activated) and decreases over time in the absence of stimulation (thus simulating the leak observed in the biological neurons) [11]. This work uses the Leaky Integrate-and-Fire (LIF) model, the most common neuromorphic artificial neuron software and hardware implementation.

B. Academic platform Kraken

The Kraken chip [12] comprises three main subsystems composing a heterogeneous architecture. The first subsystem, the Fabric Controller (FC), is built around a 32-bit RISC-V core that acts as the central control unit for the entire SoC. FC contains the main interconnection busses to the main L2 memory and the Advanced Peripheral Bus (APB), which controls all the SoC peripherals. Additionally, the FC hosts a complete set of on-chip peripherals and a power management unit.

The second subsystem is a RISC-V-based general-purpose accelerator known as the "Cluster," which hosts eight RISC-V cores enhanced with specialized Instruction Set Architecture (ISA) extensions. The Cluster also features a $128\,\mathrm{KiB}$ L1 Tightly Coupled Data Memory (TCDM) and a dedicated hardware block for fast event management, parallel thread dispatching, and synchronization.

The third subsystem, the External Hardware Processing Engine (EHWPE), hosts two accelerators, including the Sparse Neural Engine (SNE) neuromorphic accelerator and CUTIE, a ternary weight neural network accelerator. The accelerators operate on two independent clock domains and are programmed via memory-mapped register interfaces.

The chip features a vast set of IO peripherals that can generate interrupts depending on data transmission events. An autonomous IO subsystem called the *µDMA* hosts all the IO peripherals. It can be programmed to orchestrate data transfers from the peripherals to the L2 memory and vice versa. The IO subsystem includes a dedicated hardware peripheral to directly interface an ULP event camera. Events, i.e., active DVS pixels, can be directly transferred from the event camera to the Kraken TCDM, where they are accessible to the EHWPE and the cluster domain for processing. Such an on-chip integrated interface reduces the event acquisition power consumption by more than one order of magnitude, compared to what is shown in [7], improving the overall energy efficiency. Moreover, Kraken also implements hardware power management strategies, e.g., power gating, that allows the on/off switching of unused system parts to reduce the overall power consumption.

The Kraken chip is mounted on an evaluation board incorporating external components essential to execute complete applications. A USB-C connector is used to power the board and for JTAG and UART data transfer. The three power domains of Kraken are supplied by buck converters that are individually runtime-configurable, permitting application-controlled DVFS. External memory is present, including a combined HyperFlash/HyperRAM chip and a quad-SPI flash memory chip that can store application code for standalone booting. Additional connectivity is provided by Arduino headers and a Camera Parallel Interface (CPI) ribbon connector and level shifters are present between each off-chip connector and Kraken's I/O pins.

C. Event camera

Instead of measuring the intensity of every pixel in a fixed time interval, event sensors report events of significant pixel intensity changes asynchronously. Every such event is represented by its position, sign of change, and timestamp, accurate to the microsecond. An event is triggered by the sensor whenever the log luminance for a given pixel changes over a threshold. Formally, an event is a tuple (x, y, t, p) that indicates the pixel x, y where the event occurred, a timestamp t (with $10^{-6}s$ time resolution), and a binary polarity p that indicates the direction of change: positive for brighter and negative for darker. Event sensors show many advantages over frame-based cameras, such as high temporal resolution, low power consumption, and high dynamic range sensing. Also, they do not suffer from motion blur or dazzling. On Kraken, the spatial and temporal position of each event acquired from the event camera is explicitly stored in memory by the DVSI *µDMA* peripheral in a sparse, low memory footprint coordinate

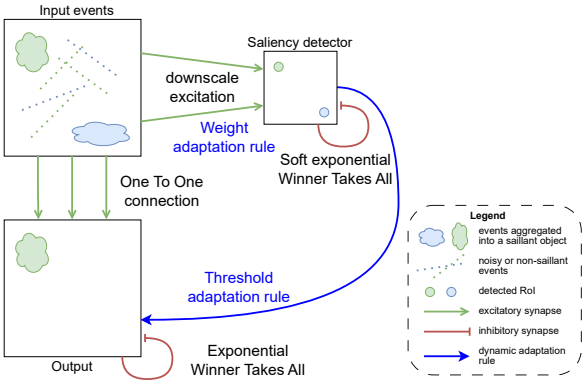

Fig. 1. Architecture of the spatiotemporal attention model introduced in [6]. (Image taken with permission from [6].) The spiking neurons are placed in rectangular layers according to a 2D orthogonal frame of reference.

list format (COO). The Cluster or the EHWPE computing domain can directly consume a variable-length stream of such events.

III. Method

A. Reference model

Authors in [6] introduce a novel neuromorphic model of spatio-temporal attention on event data, implemented on both the CPU simulator PyNN [13] and Intel's neuromorphic hardware Loihi [8]. This model, illustrated in Fig. 1, detect the salient regions of the input sensor and outputs the first corresponding object of interest (i.e., the spikes corresponding to the detected region). It performs this task using a network composed of three SNN layers linked by two sets of excitatory synapses (the downscale connection between the input and the RoI detector and the One-to-One connection between the input and the output layers) and two sets of inhibitory synapses implementing WTA mechanisms. Additionally, the model can adapt to any input data thanks to two adaptive mechanisms: a weight adaptation rule — which favors the activation of the RoI detector depending on its previous activations, and a threshold adaptation rule — which promotes the passage of spikes in the output regions corresponding to the salient regions identified by the RoI detector and hinders this passage in other places. All in all, this neuromorphic event-based spatiotemporal attention model achieves interesting results by solely relying on intrinsic SNN dynamics.

B. Implementation on Kraken

The model described in [6] was initially implemented using PyNN, a simulator-independent Python interface for SNN simulators [13], combined with NEST (NEural Simulation Tool) [14], a Python simulator for SNN on CPUs; then implemented on Intel's neuromorphic hardware Loihi [8]. The main limitations of these tools are the simulation time and the impossibility of modifying neuronal and synaptic parameters during the simulation, thus hindering the implementation of dynamic adaptation rules. Both those limits can be overcome using the academic platform Kraken [12], which allows for the adaptation of neuronal and synaptic parameters mid-run.

The Sparse Neural Engine (SNE), Kraken's dedicated neuromorphic energy-proportional accelerator, is primarily designed to accelerate feed-forward neural networks. Such networks are trained using the framework PyTorch [15]. This framework differs significantly from PyNN: in the latter, each neuronal population is implemented first and then interconnected; PyTorch implements a neural network by instantiating layers including a specific synaptic connectivity. Therefore an adaptation of the architecture's implementation is required to allow the reference model to be deployed on Kraken. Pytorch-defined networks are deployed on Kraken by converting them into an intermediate neural network representation format (ONNX). Executable C code is then auto-generated by using a dedicated Kraken deployment toolchain. The low-level generated C code is entirely available for the programmer and, therefore, can be modified to include dynamic threshold adaptation rules and custom interconnection schemes among neurons belonging to different populations.

To map the saliency detection algorithm to Kraken, the original synaptic interconnection schemes connecting the various neuron populations, and performing the attention mechanism, have been mapped on equivalent neural network layers which account for the neuron synaptic connectivity supported by SNE, i.e., 1x1 and 3x3 convolutional and fully-connected synaptic connectivity, and the exact LIF neuron model implemented in SNE. Specifically, the downscale connection between the input and the RoI detector, originally implemented as two populations of LIF neurons connected *via* an excitatory downscaling connection (i.e., a convolutional layer with no padding — see Fig. 1), is translated into a fully-connected synaptic connection with no bias and with synaptic weights of the unconnected regions set to zero. Indeed, the PyTorch linear layer is the most appropriate to implement All-to-All connections. The excitatory One-to-One connection between the input and the output layers (see Fig. 1) is translated into a convolutional layer with a kernel of size 3, padding of size 1 and no pooling nor bias. All the weights are null, except the

TABLE I
Neuronal and synaptic parameters used to implement [6]'s model on PyNN and on Kraken.

Parameters	PyNN's values [6]	Values in our PyTorch implementation
RoI detector		
Resting membrane potential	$-65mV$	0
Neuronal threshold	$-25mV$	0.9
ω_{WTA}	1	0.1
ω_{reset}	0.7	10
$\Delta\omega$	0.01	5
Output layer		
Resting membrane potential	$-65mV$	0
Neuronal threshold	$-20mV$	0.7
ω_{WTA}	50	0.2
θ_{reset}	$-65mV$	50
θ_{max}	$0mV$	100
$\Delta\theta$	$10mV$	1

kernel's central weight, which is set to 1, thus allowing only a connection between corresponding pre and post-synaptic neurons. Additionally, exponential WTA mechanisms applied to the RoI detector and the output layer are implemented using linear layers with no bias. Since the inhibition is implemented in PyTorch by setting the weights to a negative value (the lower the value, the higher the inhibition), the exponential rule is updated to enable a correct WTA behavior as described in Eq. 1:

$$\omega_{WTA} = \max(\frac{-e^d}{w \times h} + \omega_{WTA}, 0) \quad (1)$$

where d corresponds to the Euclidean distance between the active and target neuron subject to inhibition, w and h to the width and height of the layer and w_{min} to the lower bound of the weight ω_{WTA}.

Finally, all neural network weights and parameters are quantized to be deployed to Kraken (see TABLE I).

IV. Experimental results

In this section, we describe the experiments conducted on the Kraken platform and discuss the results of such explorations compared to the performance achieved by [6] using the same event-based dataset: DVS 128 Gesture [16].

Fig. 2 compares the simulation time of the attentional algorithm on a CPU[1], Loihi and Kraken. We show that it is possible to achieve real-time saliency detection with Kraken. Furthermore, we compare the quality of the detected OoI by comparing the classification accuracy obtained on those detected in DVS128 Gesture [16], performed by the Parametric Leaky Integrate-and-Fire (PLIF) classifier [17]. The classification performance is 20% better on the OoIs detected by Kraken compared to PyNN: this marked increase both confirms the quality of the implementation of the original model on Kraken and can be explained by the greater number

[1]Intel 8-core i9-10885H at 2.4GHz.

TABLE II
SNE latency and energy consumption of the saliency detection functional modules

Layer	Energy (in J)	Latency (in s)
Input to RoI detector	3.46×10^{-4}	1.26×10^{-2}
WTA on RoI detector	2.16×10^{-5}	7.86×10^{-4}
Input to Output	3.63×10^{-4}	1.26×10^{-2}
WTA on Output	3.12×10^{-3}	1.13×10^{-1}
Total	3.85×10^{-3}	1.39×10^{-1}

TABLE III
Kraken platform energy consumption and latency

Kraken platform	Current (in A)	Simulation time	Energy (in J)
Fabric controller	9.5×10^{-3}	1.4×10^{-1}	8.62×10^{-4}
SNE - active (avg)	44.4×10^{-3}	1.39×10^{-1}	3.85×10^{-3}
SNE - idle	3.635×10^{-2}	3.6×10^{-4}	8.51×10^{-6}
Total	NA	1.4×10^{-1}	4.72×10^{-3}

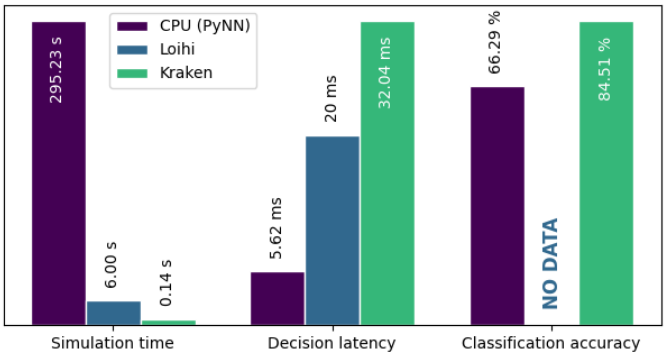

Fig. 2. Simulation time for an event input of $1s$, detection latency (i.e. how long should be simulated before identifying the OoIs) and PLIF classification accuracy [17] of the OoIs detected in DVS128 Gesture [16] by the model introduced in [6], compared between the PyNN and Loihi implementations presented in [6] and the current Kraken deployment. As Loihi's data comes from [6], no classification accuracy could be measured on this hardware.

of events contained in Kraken's OoIs. Indeed, the PLIF classifier [17] accumulates events in frames and performs better on a dataset rich in events.

TABLE II provides the energy and latency breakdown for all the relevant algorithm functional modules benchmarked on the SNE accelerator. Our experiments used a 6% spiking activity, the highest spiking activity measured across DVS 128 Gesture. TABLE III reports the overall energy and execution latency for the entire Kraken platform, including the energy consumption of the RISC-V core (Fabric Controller) that orchestrate the algorithm execution on SNE. TABLE III also reports the SNE idle consumption, i.e., when the accelerator is waiting for Fabric Controller to reprogram the SNE accelerator for a new execution of the algorithms. Kraken achieves real-time simulation run-time, as it requires 0.14s to execute the saliency detection algorithm on 1s of input spiking activity from an event camera, significantly lower than both Loihi and CPU run times. Moreover, compared to the algorithm implementation executed on Loihi and on CPU, it is possible with Kraken to fully implement, on-site, the dynamic weight and threshold adaptation rules presented in [6], which are a crucial feature for online adaptation.

V. Conclusion

This paper introduces an embedded neuromorphic algorithm for saliency detection in event data. The proposed approach minimises the model size and inference latency, consumes only $3.85mJ$ per second of processed data, and detects objects of interest with quality such that it maintains an accuracy of 84.51% in a classification task while processing data 10 times faster than real-time. This proposal is the first saliency detection algorithm deployed on Kraken, an academic platform with a neuromorphic hardware accelerator.

Future work will compare the energy consumption of the algorithm deployed on Kraken to a CPU simulation and Intel Loihi. The parameters identified in this paper will also be used to compare this Kraken deployment to the SpiNNaker [18] neuromorphic hardware, which also allows for implementing dynamic adaptation rules.

REFERENCES

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[2] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[3] F. Moosmann, D. Larlus, and F. Jurie, "Learning saliency maps for object categorization," 2006.

[4] A. Renner, M. Evanusa, and Y. Sandamirskaya, "Event-based attention and tracking on neuromorphic hardware," *IEEE CVPRW*, 2019.

[5] Y. Sandamirskaya, "Dynamic neural fields as a step toward cognitive neuromorphic architectures," *Fr. in Neurosciences*, vol. 7, 2014.

[6] A. Gruel, A. Vitale, J. Martinet, and M. Magno, "Neuromorphic event-based spatio-temporal attention using adaptive mechanisms," in *AICAS*, 2022.

[7] A. Di Mauro, M. Scherer, J. F. Mas, B. Bougenot, M. Magno, and L. Benini, "Flydvs: An event-driven wireless ultra-low power visual sensor node," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1851–1854, IEEE, 2021.

[8] M. Davies *et al.*, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Pr. of the IEEE*, vol. 109, no. 5, 2021.

[9] A. Di Mauro, A. S. Prasad, Z. Huang, M. Spallanzani, F. Conti, and L. Benini, "Sne: an energy-proportional digital accelerator for sparse event-based convolutions," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 825–830, IEEE, 2022.

[10] X. Wang, M. Magno, L. Cavigelli, and L. Benini, "Fann-on-mcu: An open-source toolkit for energy-efficient neural network inference at the edge of the internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4403–4417, 2020.

[11] H. Paugam-Moisy and S. M. Bohte, "Computing with Spiking Neuron Networks," in *Handbook of Natural Computing*, Springer-Verlag, 2012.

[12] A. Di Mauro, M. Scherer, D. Rossi, and L. Benini, "Kraken: A direct event/frame-based multi-sensor fusion soc for ultra-efficient visual processing in nano-uavs," in *HCS*, pp. 1–19, 2022.

[13] A. P. Davison *et al.*, "PyNN: A Common Interface for Neuronal Network Simulators.," *Fr. in Neuroinformatics*, vol. 2, 2009.

[14] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," 2007.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[16] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *CVPR*, 2017.

[17] W. Fang *et al.*, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *ICCV*, 2021.

[18] S. B. Furber *et al.*, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, 2013.