SISSA
Scuola
Internazionale
Superiore di
Studi Avanzati

Neuroscience Area- PhD course in
Cognitive Neuroscience

Exploiting spatio-temporal patterns with neuromorphic systems

Candidate:
Alejandro Pequeño Zurro

Advisor:
Stefano Panzeri
Co-advisors:
Mathew Diamond

Academic Year 2022-23

“What I cannot create, I do not understand"

Richard Feynman

Acknowledgements

Thanks to my supervisor Stefano Panzeri, for giving me this opportunity with a Marie Skłodowska–Curie fellowship. To the Neutouch coordinator Chiara Bartolozzi for the training opportunities, and to Mathew Diamond as my co-supervisor for the help and support in SISSA. From the ITN network, I would also like to express my gratitude to Elisabetta Chicca for welcoming me to her lab during my lab secondment.

At the research level, I am grateful for the collaborators I have worked with during these years. In alphabetical order: Alisea Stroligo, Ella Janotte, Iacopo Hachen, Lyes Khacef, Sebastian Reinartz, Simon F. Muller-Cleve, Vittorio Fra. Many people have participated in this thesis in the form of interesting discussions about research topics, methodologies, manuscript ideas, and future collaborations and directions. Starting from my lab colleagues from the Neural Computation lab in Genova, acknowledging my Neutouch fellow colleagues and following, in chronological order, with my colleagues from the Bioinspired Systems and Circuits from RUG. Keep questioning. Thank you all.

Finally, on a personal level, thank you to all the people who inspired me to keep forward on my path, especially my parents, my life partner, and my grandfather, who ignited my curiosity at an early age.

Abstract

New demands in artificial intelligence, the increase of data available, and the forecasts of reaching Moore's law ceiling push algorithms towards the edge for low latency, low power, and highly intelligent devices. Neuromorphic systems mimic brain-like computations to capture the efficiency and adaptive behaviour exhibited in biological systems being promising candidates to lead the new generation of artificial systems. Sensory information is encoded with precise spatio-temporal patterns in the nervous system. Similarly, neuromorphic computation utilises event-based representations in its computational systems, inspiring its use for building artificial cognitive systems. In this work, we build Spiking Neural Networks (SNNs) in the tactile and auditory sensory modalities for classification tasks in the context of full event-based sensory systems. Compared with standard machine learning implementations in GPU, SNNs implemented in neuromorphic hardware output close to 500 times less energy consumption highlighting the strengths of the new hardware paradigm. Also, we explore how to encode spatio-temporal features from sensing devices emphasising the benefits of using full neuromorphic event-based sensors and systems. We further explore improvements in spiking networks based on the Leaky-Integrate-and-Fire (LIF) models. The proposed network's architecture implements Time Difference Encoders (TDEs), a cell model based on brain-inspired computations. We find promising results with a 92% reduction of synaptic operations and highly interpretable network results against typical current-based recurrent LIF networks. The results of this work contribute towards building neuromorphic sensing systems from sensor devices, algorithms and circuit design to quantify the strengths of the new computational paradigm with promising capabilities for intelligent machines on edge inspired by their biological counterparts.

Keywords: neuromorphic sensing, spatio-temporal pattern recognition, spiking neural networks, time difference encoder, energy efficiency, tactile encoding, keyword spotting

Note

Introduction. Extracts of the text in Section 1.2 have contributed to the following work.

S. Panzeri, E. Janotte, A. Pequeño-Zurro, J. Bonato, and C. Bartolozzi, "Constraints on the design of neuromorphic circuits set by the properties of neural population codes, Neuromorphic Computing and Engineering, vol. 3, no. 1, p. 012001, jan 2023. [Online]. Available: https://dx.doi.org/10.1088/2634-4386/acaf9c

Chapter 2 here corresponds to a version of the following publication.

S. F. Müller-Cleve, V. Fra, L. Khacef, A. Pequeño-Zurro, D. Klepatsch, E. Forno, D. G. Ivanovich, S. Rastogi, G. Urgese, F. Zenke, and C. Bartolozzi, "Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware," Frontiers in Neuroscience, vol. 16, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2022.951164

Chapter 3 corresponds to a manuscript with the same name, in preparation to be submitted.

A. Pequeno-Zurro, L. Khacef, S. Panzeri, E. Chicca, "Towards efficient keyword spotting using spike-based time difference encoders" [In Preparation].

Contents

List of Figures

List of Tables

List of Acronyms

ADAM	ADAptive Moment estimation
AER	Address-Event Representation
API	Application Programming Interface
ANN	Artificial Neural Network
ANNs	Artificial Neural Networks
BMI	Brain Machine Interface
BPTT	Backpropagation Through Time
CDC	Capacitance to Digital Converter
CNN	Convolutional Neural Network
CUBA	current-based
CUBA-LIF	Current-Based Leaky Integrate-and-Fire
DVS	Dynamic Vision Sensor
EPSC	Excitatory PostSynaptic Current
FC	Fully-Connected
FCN	Fully Convolutional Network
FFSNN	Feed-Forward SNN
GAP	Global Average Pooling
HMM	Hidden Markov Model
HMMs	Hidden Markov Models
HPO	Hyperparameters Optimization
ISI	Inter-Spike Interval
ITN	Innovative Training Network

MFCCs	Mel Frequency Cepstrum Coefficients
MSE	Mean Squared Error
MTB	Microcontroller Board
MTS	Multivariante Time Series
NNI	Neural Network Intelligence
LIF	Leaky Integrate and Firing Neuron Model
LIFr	Leaky Integrate and Firing Neuron Model with recurrency
LIFrec	Leaky Integrat-and-Fire with recurrency
LSTM	Long Short-Term Memory
OS	Operating System
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PReLU	Parametric Rectified Linear Unit
ResNet	Residual Neural Network
RSNN	Recurrent Spiking Neural Network
sEMD	spiking Elementary Motion Detector
SIMD	Single-Instruction Multiple-Data
SoC	System-on-Chip
SOM	System-on-Module
SNN	Spiking Neural Network
SNNs	Spiking Neural Networks
SVM	Support-Vector Machine
TDE	Temporal Difference Encoder
TCNN	Time-CNN
TTC	Time-to-Classify

Introduction

1 Preface

Biological systems such as humans, mammals or insects exploit their environment through sensory systems. Natural signals are inherently perceived as spatio-temporal through our senses. In the spatial dimension, given the distributed, parallel processing of the sensory receptors and in the time domain, sensory information fluctuates in time at a highly variable pace. This spatio-temporal signal from nature is transduced into precise spatio-temporal patterns of spikes in the nervous system. Once transmitted, the brain can combine, compute, and transform this spatio-temporal code into the sensory percept, which is ultimately read out and utilised for a perceptual task. While biological systems are seamlessly exploiting the natural environment with their sensory systems, in the artificial domain, even the most powerful algorithms and systems are only able to display a subset of their capabilities, with a higher order of magnitude in terms of energy consumption. Capabilities such as navigation in uncontrolled environments, self-organising collective behaviours, generalisation of previous knowledge for problem-solving, or spatial and body awareness are only a few examples of the potential that brain-inspired algorithms can bring to the artificial domain [1–3].

The neuromorphic field aims to harness the robust, distributed, asynchronous, and adaptive nature of biological intelligence into in-silico systems by building circuits that emulate the biological counterparts and simulates neural processing. The biological process has been extensively studied and has inspired a generation of artificial systems that aims to distil the brain's capabilities to exploit complex spatio-temporal patterns and apply this knowledge to neuromorphic circuits [3, 4]. Two fundamental needs in intelligent systems point to the neuromorphic field as a promising

candidate to power the new generation of algorithms and devices: (i) The increasing requirement of data processing capabilities for highly complex artificial intelligence models is moving the computation to the device's edge; (ii) the computational cost, in term of energy expenditure, that requires the training and usage of models bonded to traditional Von Neumann architectures [5–8].

Neuromorphic technologies encompass a wide range of scientific areas such as neuroscience, computer science, artificial intelligence, materials, electronics, robotics and bioengineering in a multidisciplinary collaborative effort for its development. Each of the neuromorphic engineering areas have shown individually very promising results: new neuromorphic platforms with digital and mixed-signal digital-analog circuits [9], sensor devices that exploit event-based representations at circuit level [2, 10], and learning algorithms based on gradient descend methods [11, 12]. In order to push event-based systems of Spiking Neural Networks (SNNs) to achieve the high-performant applicability reached by Artificial Neural Network (ANN), the neuromorphic community needs to find applications, standardised benchmarks, and build use-cases. This requires a multidisciplinary effort in which the neuromorphic community combines technologies and systems to solve problems unsolved to date [3, 13].

This thesis aims to provide light on how all these pieces fit in a full-scale sensory system, understanding the problem from the application point of view to evaluate the true potential of SNNs as a system. In this work, we focus on identifying how we can build artificial systems that better exploit the spatio-temporal structure of sensory information with SNNs. In particular, our effort focuses on providing tools for benchmarking spiking neural networks and finding use-cases in the sensory categorisation tasks where the actual developments in the field can improve the existing system's implementations: (i) finding the gaps in the field to build full-scale sensing systems that utilise event-based representations, (ii) analysing the conditions in which neuromorphic systems can excel, (iii) and providing new ideas that utilise the brain-like computations of neuromorphic systems.

Artificial systems that exploit sensory information are designed similarly to their biological nervous system counterparts. We transduce the natural signals in space and time with highly precise spatio-temporal code that conveys the information needed for perceptual discrimination tasks. We can utilise the high parallelism of event-based computations in a similar way that

neural population codes compute stimulus information for perceptual discrimination tasks. Therefore we hypothesise that sensory signal processing may be an excellent candidate to harness the advantages of the event-based representations of neuromorphic algorithms and systems. In terms of sensory modalities, visual and auditory signals have a large number of studies in terms of applications in the neuromorphic field [14]. More work needs to be done in the tactile domain; since this modality is one of the largest spatially distributed in the biological body, generating an immense amount of information and being crucial in many challenges we are still not able to resolve (e.g. active exploration, object placing, human interaction) [15, 16]. We understand that the tactile modality will be one of the modalities most highly impacted by the development of neuromorphic systems that provide highly scalable, highly parallel processing capabilities. To build perceptual systems that mimic not only the capabilities of humans but can also deal with artificial skin with a similar number of receptors and heterogeneity of responses of the human counterpart [17–19]. In the second part of this thesis, we focus our efforts on the auditory domain, a modality with many studies in sensory devices and systems [20]. Choosing the auditory domain aims to evaluate and benchmark the typical models of Spiking Neural Networks with recent brain-inspired neuron models. We are deepening the knowledge constructed in the first work about spiking neural network algorithms to understand how to increase the sensing capabilities to implement autonomous agents on-device.

1.1 Neuromorphic systems

Algorithms and networks based on the ANN have increased their popularity with the appearance of deep learning systems [21]. This type of network, and other standard artificial intelligence algorithms, are associated with the Von Neumann computer architecture characterised by the separation of computation and memory. This architecture heavily depends on the memory bandwidth bottleneck that delays the processing time in case big amounts of data are exchanged between the computing and the memory units. Furthermore, this exchange process is also responsible for the system's most considerable energy consumption. Industry trends demand new computational paradigms to satisfy the requirements towards low-power computation at the edge devices. We need machines that interact with their environment, adapt on the go, and build their own models as they experience the world [3, 6]. In order to build more general adaptive

algorithms with the standard approaches and computation paradigms, it is required to increase the complexity of the model and training data over a different range of conditions to generalise. This effect, consequently, leads to a significant increase in the power consumption and latency of the system [7, 22].

Neuromorphic cognitive systems are a promising candidate to represent the new frontier in artificial systems by mimicking the capabilities of biological systems, intending to capture the adaptive, highly efficient nature of biological intelligence. The neuromorphic term was first coined by Carver Mead in the late 80s, referring to mixed analog-digital signal artificial circuits that reflect the neurobiology in the brain [23, 24]. Nowadays, the term encompasses a wide range of technologies, from spike-based processing systems that emulates biophysical models of neural circuits to the design of electronic circuits from neural architectures that resemble, in some degree, brain-inspired computations [3]. The field has already demonstrated promising results in finding new computational paradigms out of the Von Neumann architecture. Some of the neuromorphic hardware examples include Truenorth from IBM [25], Loihi from Intel [26], the SpiNNaker chip from the University of Manchester [27], and the neuromorphic chip developed in the ETH of Zurich [28] between others.

New brain-inspired hardware architectures should be accompanied by brain-inspired sensors and algorithms to embrace the strengths of natural systems fully. In analogy to the sensory-motor control loop of cortical circuits, robotic actuators can sense the world and adapt their motor behaviour through their sensors. Neuromorphic systems provide the most suitable solution to develop sensory systems with highly cognitive capabilities [1, 2, 4]. From the acquisition of sensory information with event-based sensors to compute event-based spatio-temporal patterns with Spiking Neural Networks (SNNs) and motor-control representations [4, 29]. Relevant examples exploiting a wide range of sensory modalities include neuromorphic retinas [30–33], cochleas [34–36], artificial olfaction [37], and somatosensory systems [38].

1.2 Lessons from neuroscience

Neuromorphic engineering also aims to extract principles from biological circuits and brain computations, to design artificial circuits and systems. In the implementation of SNNs into

neuromorphic chips, neurons are modelled into parallel processes that perform their own computations, are assigned their own memory, and asynchronously communicate with each other through events (Address-Event Representation (AER) protocol) [39]. Similar to the biological brain, we can also model synapses and implement neuron models of synaptic computation into full-scale networks. We can even build decoders from brain signals when we adapt the timescale of the artificial neurons to the biological neurons' signals [40, 41]. In a complementary manner, we can also extract lessons from neural populations to build neuromorphic systems that exploit spike-based computations.

In the nervous system, in vivo, information is encoded and transmitted at the level of populations of neurons [42–46]. From the point of view of information encoding, it is becoming clear that information is distributed in the brain across large populations of neurons with remarkably fine spatio-temporal resolution. In particular, population information is encoded with single-neuron and individual action potential resolution (spatial domain). The timing and not only the presence of individual action potentials seem to carry behaviourally relevant information (temporal domain). Finally, several studies suggest that how neural activity is organized across time and neurons influences not only the encoding of information but also how efficiently the information is read-out by other neural circuits and ultimately informs behaviour [46–48]. While there are several studies about the properties of the neural code, little work has been done to transfer this knowledge into artificial systems.

The brain encodes, transmits, and reads-out spatio-temporal information from spike trains. Understanding how the brain performs these capabilities have a direct implication in the design of artificial systems that exploit natural signals from the world into event-based data. But it also benefits new approaches to decode and communicate with signals within the brain itself. In this direction, we reviewed and studied how the main features of the population code (sparseness, heterogeneity, activity correlations, timescales of information encoding and timescale of information consistency) have a direct impact on the design of neuromorphic circuits for communication with the brain (Brain Machine Interface (BMI)) [49]. In the cited manuscript, we review the properties of the neural codes, and we link with methods applied to the design of neuromorphic circuits, how to align the means of communication between brain-machine to the specific cortical region and functionality (read/write/closed-loop).

2 Research goals

The main research topic of this thesis explores how to exploit spatio-temporal information with full sensory neuromorphic systems. With that in scope, we defined two research questions that will be explored in the following chapters of this thesis. The following questions, research approach, and contributions are summarised in Figure 1. In combination, this work proves the strengths and capabilities of Spiking Neural Networks (SNNs) for exploiting spatio-temporal information in the context of neuromorphic sensing systems and circuits.

Research question. What are the strengths and weaknesses of event-based neuromorphic systems versus traditional spatio-temporal classifiers, and how can we leverage these strengths to design more effective systems?

The first contribution refers to the tactile domain with data obtained from a robotic fingertip sliding over braille letters, imitating a human braille reader. In this work, we established a benchmark for algorithms using spatio-temporal information from tactile sensors. We compare event-based neuromorphic systems with standard classifiers in event-based and time-based data (also called frame-based in reference to vision cameras). We explore different encoding schemes of the delta coding and analyse the trade-offs between encoding parameters from the frame-based and event-based data. Then, we implement the networks in hardware and compare the performance and energy consumption of the most high-performing machine learning classifiers in GPU-based architectures against event-based SNNs implemented in neuromorphic chips.

We analyse the information encoded in the time domain in the generated datasets in order to motivate the use of SNNs and understand how efficient they are at exploiting the spatio-temporal structure. We generate a novel dataset for the community to benchmark spatio-temporal event-based classifiers in the tactile domain that supports the use of tactile sensory information and the development of neuromorphic algorithms. We reveal the trade-offs in encoding event-based signals from time-based sequences, highlighting the importance of neuromorphic event-based sensors for full event-based computation systems. We display a performance comparative between machine learning training systems and event-based computation with SNNs. We remark on the high energy efficiency capabilities of event-based computations with

SNNs implemented in neuromorphic hardware, against traditional computational and hardware architectures. The capabilities found demonstrate the fitness of neuromorphic systems for spatio-temporal pattern recognition in the tactile domain, matching the performance for event-based information. This performance comparison does not consider the loss of information for the sensory device and the transformation of the information to event-based data. Whose effects, per se, motivate the use of event-based sensory systems from the source.

Research question. How to use insights from neuroscience and neuromorphic engineering to design event-based neuromorphic models of neurons that most efficiently utilise the capabilities of brain-inspired computations in neuromorphic chips?

In the second contribution of this thesis, we explore the use of Temporal Difference Encoder (TDE) as a bio-inspired cell model, building SNNs that exploits spatio-temporal information from speech signals. We compare the TDE cell, which derives from the LIF model, with typical LIF architectures. LIF models are typically used in SNNs, and they have proven to be models successfully emulated with neuromorphic circuits. We analyse the contributions of the TDE model in SNNs in terms of inference efficiency, scalability, and interpretability of the results, in the context of spatio-temporal pattern classification with spoken digits.

In the contributions of this work, we open new perspectives for building SNNs with cell models inspired by brain-like computations. We prove the importance of the temporal domain in the encoded features of the speech signals and quantify the information about time and space in the patterns. We classify with high performance the encoded spatio-temporal patterns showing evidence that the TDE model can improve the performance of SNNs in terms of energy efficiency, scalability, and interpretability of the results. The results of the implementation of the TDE model support the design of neuromorphic analog-digital mixed signal circuits since the increase in complexity of the model can be emulated without real computational cost. However, the TDE model exploits the spatio-temporal structure of the data without network recurrence. The outcomes obtained show promising results in building highly efficient computation, low-power, and low-latency sensory systems.

Exploiting spatio-temporal patterns in neuromorphic systems

What are the strengths and weaknesses of event-based neuromorphic systems versus traditional spatio-temporal classifiers, and how can we leverage these strengths to design more effective systems?

Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware

Approach	Contribution
* Setup a robotized finger for experiments with tactile features * Encoding information for time-based to event-based * Train SNNs and typical machine learning approaches for spatio-temporal classification * Implement SNNs and typical classifiers in neuromorphic and GPU based hardware architectures	* Dataset of tactile spatio-temporal patterns for benchmarking neuromorphic algorithms * Evidence of tactile features encoded in the time domain * Understanding the trade-off performance/event-sparsity in time-to-event encoding and motivation for the use of event-based sensors and circuits * Outstanding high energy efficiency capability of event-based computations with neuromorphic hardware against typical GPU based machine learning approaches

How can we use insights from neuroscience and neuromorphic engineering to design event-based neuromorphic models that most efficiently utilises the capabilities of brain-inspired computations in neuromorphic chips?

Towards efficient keyword spotting with temporal difference encoders

Approach	Contribution
* Encoding of spatio-temporal features to event-based data from audio speech * Simulation of bio-inspired cell model in SNNs that exploit time difference between sensory inputs (TDE) * Training of SNNs for classification of audio speech with typical cell models against TDE cell * Analysis in terms of efficiency, performance, and interpretability	* Insights for the design and testing of bio-inspired models based on neural circuits for building SNNs * Evidence for high efficiency gain in the number of computations, improved interpretability, and better scaling in the use of the TDE cell model for event-driven processing of spatio-temporal features * Motivation for the design of full sensory systems with mixed analog/digital circuits

Fig. 1: Research questions and contributions of this work.

3 Neutouch (ITN)

The research presented in this manuscript was performed within the Innovative Training Network (ITN) of NeuTouch, a project supported by the European Union's Horizon 2020 Marie Skłodowska-Curie Actions (grant agreement 813713). 15 Early Stage Researchers (ESR) and 16 principal investigators build NeuTouch, including 9 different institutions, such as universities, research institutes, and industrial companies.

The Neutouch project aims to improve artificial touch perception and hand prostheses by designing systems with bioinspired electronic circuits, known as neuromorphic computing. In some cases, artificial systems have reached even better results than structures that evolution has been shaping for millions of years. However, we know that one of the optimisations that evolution performs is energy consumption which leads to system efficiency. If we want to produce systems that work well and with low energy consumption, looking at biological systems is the most straightforward way to reach such an objective. Following this insight, we may be able to create not only robots that achieve more human behaviour but also prostheses exhibiting a better human-machine integration providing then the user with a more realistic and intuitive experience (e.g. in terms of perception or feedback), which could be crucial to improve the user's abilities and precision.

With this aim, Neutouch was organised as a multidisciplinary project divided into three different overlapping topics that give rise to three different working packages. The Technologies for Touch working package aims to develop novelty materials and neuromorphic sensors that mimic the transduction capabilities of the peripheral nervous system. The Touch for Robotics working package aims to investigate sensorimotor control strategies from event-based sensors and to develop behavioural policies that enhance the extraction of tactile features for tactile classification purposes. The third working package, Touch for Prosthetics, aims to restore sensitivity in missing human limbs through artificial limbs surgically connected to the peripheral nervous systems, understanding of the neural code and the strategies of neuromorphic circuits to adapt the artificial code to interface with the nervous system. The main contributions of this work to the working packages are highlighted in the Venn diagram of Figure 2.

Neutouch was also an environment providing ESRs with several training opportunities and resources. During the project's development, three international summer schools were organised, and soft-skills courses addressing transversal needs for science were provided to the PhD students. Additionally, we were involved in organising events for the Horizon project and contributed to deliverables and contingency plans as part of the ITN responsibilities. Those opportunities provided the students not only with theoretical soft-skills knowledge but also with experience to improve their professional careers in industry and academia.

For further information about the Neutouch project and programmed activities, visit the EU project website.

4 COVID-19 statement

The PhD project in this manuscript started in 2020 when the Covid-19 pandemic outbreak occurred. The measurements taken to stop Covid-19 affected the organization of the Neutouch network, including this project.

Neutouch launched a contingency plan to facilitate the PIs and students in developing their work, minimizing the impact and avoiding significant delays due to travel restrictions, supply delays, and restrictions for physical experimentation. To do so, online meetings and training events were organized to embrace the collaborations between different partners and facilitate the discussion of the main topics that Neutouch was focused on.

Concerning this project, the pandemic impacted the number of home office hours out of contact in real life with members of the same lab in the initial period of the project. Additionally, it has been impacted by a need for more networking and idea exchange opportunities with other researchers in the field that in-person conferences provide. During the first year of the pandemic, the percentage of home office work days may have reached 40% of the total working hours.

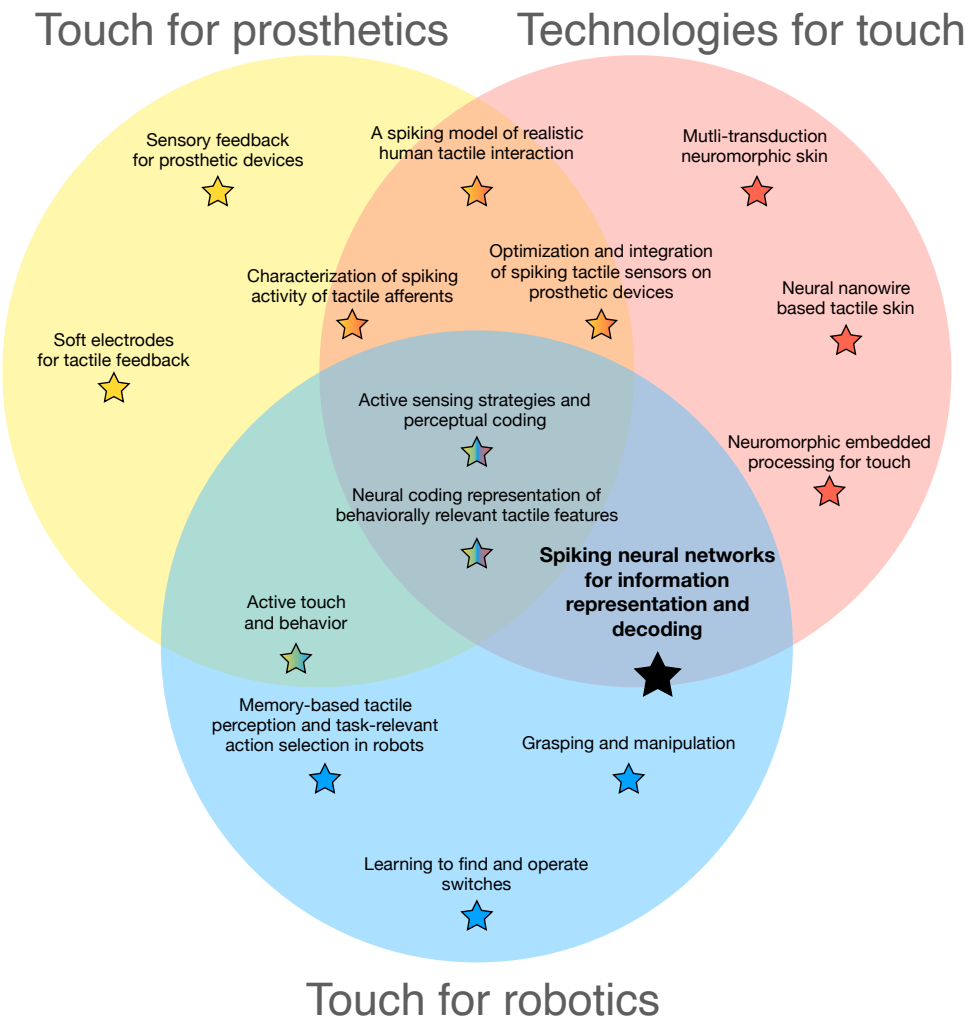


Fig. 2: Neutouch organization in working packages including the contributions of the researchers.

References

[1] S. C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.conb.2010.03.007

[2] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, 2014.

[3] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel *et al.*, "2022 roadmap on neuromorphic computing and engineering," *Neuromorphic Computing and Engineering*, vol. 2, no. 2, p. 022501, 2022.

[4] S. Sheik, M. Pfeiffer, F. Stefanini, and G. Indiveri, "Spatio-temporal spike pattern classification in neuromorphic systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8064 LNAI, pp. 262–273, 2013.

[5] C. Toumey, "Less is moore," *Nature nanotechnology*, vol. 11, no. 1, pp. 2–3, 2016.

[6] Y. LeCun, "1.1 deep learning hardware: past, present, and future," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 12–19.

[7] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 1393–13 696, 2020.

[8] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[9] D. Ivanov, A. Chezhegov, M. Kiselev, A. Grunin, and D. Larionov, "Neuromorphic artificial intelligence systems," *Frontiers in Neuroscience*, vol. 16, 2022.

[10] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S. C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, no. MAY, pp. 1–23, 2011.

[11] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[12] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[13] M. Davies, "Benchmarks for progress in neuromorphic computing," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 386–388, 2019. [Online]. Available: http://dx.doi.org/10.1038/s42256-019-0097-1

[14] A. Vanarse, A. Osseiran, and A. Rassau, "A review of current neuromorphic approaches for vision, auditory, and olfactory sensors," *Frontiers in Neuroscience*, vol. 10, no. MAR, pp. 1–6, 2016.

[15] C. Bartolozzi, L. Natale, F. Nori, and G. Metta, "Robots with a sense of touch," *Nature materials*, vol. 15, no. 9, pp. 921–925, 2016.

[16] S. Sundaram, "How to improve robotic touch," *Science*, vol. 370, no. 6518, pp. 768–769, 2020.

[17] H. P. Saal, B. P. Delhaye, B. C. Rayhaun, and S. J. Bensmaia, "Simulating tactile signals from the whole hand with millisecond precision," *Proceedings of the National Academy of Sciences*, vol. 114, no. 28, pp. E5693–E5702, 2017.

[18] M. M. Iskarous and N. V. Thakor, "E-Skins: Biomimetic Sensing and Encoding for Upper Limb Prostheses," *Proceedings of the IEEE*, vol. 107, no. 10, pp. 2052–2064, 2019.

[19] G. Corniani and H. P. Saal, "Tactile innervation densities across the whole body," *Journal of Neurophysiology*, vol. 124, no. 4, pp. 1229–1240, 2020.

[20] M. H. Tayarani-Najaran and M. Schmuker, "Event-Based Sensing and Signal Processing in the Visual, Auditory, and Olfactory Domain: A Review," *Frontiers in Neural Circuits*, vol. 15, no. May, pp. 1–31, 2021.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] N. Kasabov, N. M. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. G. Doborjeh, N. Murli, R. Hartono, J. I. Espinosa-Ramos, L. Zhou, F. B. Alvi, G. Wang, D. Taylor, V. Feigin, S. Gulyaev, M. Mahmoud, Z. G. Hou, and J. Yang, "Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications," *Neural Networks*, vol. 78, pp. 1–14, 2016.

[23] C. Mead and M. Ismail, *Analog VLSI implementation of neural systems*. Springer Science & Business Media, 1989, vol. 80.

[24] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[25] D. S. M. Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswa, "Amillion spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 7812, pp. 668–673, 2014.

[26] M. Davies, T.-h. Lin, C.-k. Lin, S. Mccoy, Y.-h. Weng, A. Wild, and H. Wang, "Loihi : A Neuromorphic Manycore Processor with On-Chip Learning," no. February, 2018.

[27] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE transactions on computers*, vol. 62, no. 12, pp. 2454–2467, 2012.

[28] S. Moradi and G. Indiveri, "An event-based neural network architecture with an asynchronous programmable synaptic memory," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 1, pp. 98–107, 2014.

[29] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *2007 IEEE international symposium on circuits and systems*. IEEE, 2007, pp. 845–848.

[30] M. Mahowald and C. Mead, "The silicon retina," *An Analog VLSI System for Stereoscopic Vision*, pp. 4–65, 1994.

[31] P. Lichtsteiner and T. Delbruck, "A 64x64 aer logarithmic temporal derivative silicon retina," in *Research in Microelectronics and Electronics, 2005 PhD*, vol. 2. IEEE, 2005, pp. 202–205.

[32] K. A. Zaghloul and K. Boahen, "A silicon retina that reproduces signals in the optic nerve," *Journal of neural engineering*, vol. 3, no. 4, p. 257, 2006.

[33] J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco, "A spatial contrast retina with on-chip calibration for neuromorphic spike-based aer vision systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 7, pp. 1444–1458, 2007.

[34] V. Chan, S.-C. Liu, and A. van Schaik, "Aer ear: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.

[35] B. Wen and K. Boahen, "A silicon cochlea with active coupling," *IEEE transactions on biomedical circuits and systems*, vol. 3, no. 6, pp. 444–455, 2009.

[36] S. Mandal, S. M. Zhak, and R. Sarpeshkar, "A bio-inspired active radio-frequency silicon cochlea," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 6, pp. 1814–1828, 2009.

[37] T. Wan, S. Ma, F. Liao, L. Fan, and Y. Chai, "Neuromorphic sensory computing," *Science China Information Sciences*, vol. 65, no. 4, pp. 1–14, 2022.

[38] F. Sun, Q. Lu, M. Hao, Y. Wu, Y. Li, L. Liu, L. Li, Y. Wang, and T. Zhang, "An artificial neuromorphic somatosensory system with spatio-temporal tactile perception and feedback functions," *npj Flexible Electronics*, vol. 6, no. 1, 2022.

[39] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[40] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition," *Neural Networks*, vol. 41, no. 1995, pp. 188–201, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2012.11.014

[41] N. K. Kasabov, "NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2014.01.006

[42] R. C. Decharms and A. Zador, "Neural representation and the cortical code," *Annual review of neuroscience*, vol. 23, no. 1, pp. 613–647, 2000.

[43] R. Quian Quiroga and S. Panzeri, "Extracting information from neuronal populations: information theory and decoding approaches," *Nat. Rev. Neurosci.*, vol. 10, no. 3, pp. 173–185, Mar. 2009.

[44] S. Panzeri, J. H. Macke, J. Gross, and C. Kayser, "Neural population coding: Combining insights from microscopic and mass signals," *Trends in Cognitive Sciences*, vol. 19, no. 3, pp. 162–172, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.tics.2015.01.002

[45] S. Nigam, S. Pojoga, and V. Dragoi, "Synergistic Coding of Visual Information in Columnar Networks," *Neuron*, vol. 104, no. 2, pp. 402–411.e4, 2019. [Online]. Available: https://doi.org/10.1016/j.neuron.2019.07.006

[46] N. Shahidi, A. R. Andrei, M. Hu, and V. Dragoi, "High-order coordination of cortical spiking activity modulates perceptual accuracy," *Nature Neuroscience*, vol. 22, no. 7, pp. 1148–1158, 2019.

[47] L. Carrillo-Reid, S. Han, W. Yang, A. Akrouh, and R. Yuste, "Controlling Visually Guided Behavior by Holographic Recalling of Cortical Ensembles," *Cell*, vol. 178, no. 2, pp. 447–457.e5, 2019. [Online]. Available: https://doi.org/10.1016/j.cell.2019.05.045

[48] O. I. Rumyantsev, J. A. Lecoq, O. Hernandez, Y. Zhang, J. Savall, R. Chrapkiewicz, J. Li, H. Zeng, S. Ganguli, and M. J. Schnitzer, "Fundamental bounds on the fidelity of sensory cortical coding," *Nature*, vol. 580, no. 7801, pp. 100–105, 2020. [Online]. Available: http://dx.doi.org/10.1038/s41586-020-2130-2

[49] S. Panzeri, E. Janotte, A. Pequeño-Zurro, J. Bonato, and C. Bartolozzi, “Constraints on the design of neuromorphic circuits set by the properties of neural population codes,” *Neuromorphic Computing and Engineering*, vol. 3, no. 1, p. 012001, jan 2023. [Online]. Available: https://dx.doi.org/10.1088/2634-4386/acaf9c

Braille Letter Reading: A Benchmark for Spatio-Temporal Pattern Recognition on Neuromorphic Hardware

Müller-Cleve SF, Fra V, Khacef L, **Pequeño-Zurro A**, Klepatsch D, Forno E, Ivanovich DG, Rastogi S, Urgese G, Zenke F and Bartolozzi C (2022) Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware. Front. Neurosci. 16:951164. https://doi.org/10.3389/fnins.2022.951164

Keywords: spatio-temporal pattern recognition, braille reading, tactile sensing, event-based encoding, neuromorphic hardware, spiking neural networks, benchmarking.

Candidate's contribution to the paper

SM-C had the initial idea and recorded the dataset. **AP-Z** and SM-C performed the frame-to-event conversion. VF, EF, and DK performed the NNI implementation. VF, DK, and **AP-Z** ran the optimization. VF and SM-C implemented the performance evaluation. **AP-Z**, DK, SM-C, VF, and DI investigated standard classifiers and carried out the encoding analysis. **AP-Z**, SM-C, and SR investigated linear SVM and obtained PCA plots. DK acquired GPU measurements. LK deployed the SNNs on Loihi and acquired the measurements. LK, GU, FZ, and CB supervised the work. All authors contributed to the manuscript writing and approved the submitted version.

Abstract

Spatio-temporal pattern recognition is a fundamental ability of the brain which is required for numerous real-world activities. Recent deep learning approaches have reached outstanding accuracies in such tasks, but their implementation on conventional embedded solutions is still very computationally and energy expensive. Tactile sensing in robotic applications is a representative example where real-time processing and energy efficiency are required. Following a brain-inspired computing approach, we propose a new benchmark for spatio-temporal tactile pattern recognition at the edge through Braille letter reading. We recorded a new Braille letters dataset based on the capacitive tactile sensors of the iCub robot's fingertip. We then investigated the importance of spatial and temporal information as well as the impact of event-based encoding on spike-based computation. Afterward, we trained and compared feedforward and recurrent Spiking Neural Networks (SNNs) offline using Backpropagation Through Time (BPTT) with surrogate gradients, then we deployed them on the Intel Loihi neuromorphic chip for fast and efficient inference. We compared our approach to standard classifiers, in particular to the Long Short-Term Memory (LSTM) deployed on the embedded NVIDIA Jetson GPU, in terms of classification accuracy, power, and energy consumption together with computational delay. Our results show that the LSTM reaches ∼97% of accuracy, outperforming the recurrent SNN by ∼17% when using continuous frame-based data instead of event-based inputs. However, the recurrent SNN on Loihi with event-based inputs is ∼500 times more energy-efficient than the LSTM on Jetson, requiring a total power of only ∼30 mW. This work proposes a new benchmark for tactile sensing and highlights the challenges and opportunities of event-based encoding, neuromorphic hardware, and spike-based computing for spatio-temporal pattern recognition at the edge.

Keywords: spatio-temporal pattern recognition, Braille reading, tactile sensing, event-based encoding, neuromorphic hardware, spiking neural networks, benchmarking.

1 Introduction

Touch, or tactile perception, is a critical component of sensorimotor activity [50]. Uniquely among the senses, in many situations deliberate action by the subject is required to experience tactile feedback: this is known as *active touch* [51]. Active use of touch is important for blind or visually impaired subjects, as visual perception may need to be aided or replaced by tactile and auditory information [52]. One example of active touch is the reading of Braille letters, where fingers slide over lines of characters that are typeset into surfaces as a set of 1-6 embossed dots arranged in a 2×3 matrix. In Braille reading, characters are read sequentially, unlike print reading, where entire words or groups of words can be perceived simultaneously by the eye. However, expert users can significantly speed up their Braille reading as they learn to identify various lexical, perceptual, and contextual clues [53], achieving optimal reading speeds of at most 80-120 words per minute [54], which is about half the average silent reading rate for adults in English [55].

The sequential and time-dependent nature of Braille reading makes it an excellent benchmark for machine learning applications involving time-varying signals since Braille text can be represented by sequential data acquired by moving tactile sensors. Optical classification applications for Braille have been developed both in machine learning [56, 57] and deep learning [58, 59]. However, the image-based approach for Braille recognition requires good quality Braille samples and accurate preprocessing. Braille documents are characterized by the lack of any contrast in color between the text and the background and may require specific lighting and camera settings to produce accurate results [60]. On the other hand, automatic Braille reading relying on tactile sensors requires adding tactile sensors to robots that can slide their sensors over surfaces. The high number of sensors necessary to obtain acceptable performance adds a high overhead in terms of area, power consumption, and communication latency.

A possible solution to these problems is the inherent data encoding capabilities and sparse transmission of neuromorphic event-driven sensing [61]. Event-driven sensors only transmit signals when a change has been detected in their sensory space, reducing communication and processing costs [62]. The event-driven domain [63] features binary, time-discrete events, avoiding the continuous polling of sensor readouts. This is especially desirable in the domain of

touch, because tactile perception is naturally sparse from a temporal and spatial perspective, with only local correlation, e.g. multiple local neighbors of receptors, or sensors, are activated by the same event. Tactile events are perceived for a limited time, as long as the stimulus is applied, and in a localized part, or patch, of the sensor. When no stimulus is present, the tactile system can be considered at rest.

While other event-driven neuromorphic sensors such as the Dynamic Vision Sensor (DVS) [64] and the silicon cochlea [65] have attracted much interest from researchers, leading to specialized data pipelines and standardized benchmarks such as the DVS gesture recognition dataset and TIDigits [66] dataset, there have been comparatively few such developments in the field of touch. Among the scarce number of examples of tactile classifiers exploiting tactile neuromorphic sensors [67, 68], See et al. proposed the Spiking Tactile MNIST (ST-MNIST) dataset of handwritten digits [69] obtained by writing on a neuromorphic tactile sensor array. Consequently, the information in the spike patterns is mostly spatial, and a feedforward Convolutional Neural Network (CNN) reaches the best accuracy when summing up all the spikes to get a "tactile image". Bologna et al. proposed a neuroengineering framework for robotic applications, including spatio-temporal event coding, probabilistic decoding, and closed-loop motion policy adaptation for active touch. The system was benchmarked on Braille, proving effective modulation of fingertip kinematics depending on character complexity. Classification of the signals recorded by a fingertip sensor mounted on a robotic arm was performed using a subset of 7 Braille characters, achieving a recognition rate of (89 ± 5.3) % [70, 71].

In this paper, we propose a development path applicable to neuromorphic tasks in the tactile domain. The proposed method, while developed and tested on tactile output obtained from capacitive sensors, can be generalized to a whole class of time-dependent data such as audio streams, inertial sensor outputs, and temperature or voltage monitoring, to name just a few. Given the inherently time-dependent nature of its information content, we selected the Braille reading problem as a benchmark, for which we designed an end-to-end event-based neuromorphic classification procedure. We acquired a reliable dataset, applied a widely adopted and well know encoding technique such as the one provided by a sigma-delta modulator, and finally performed classification relying on a neuro-inspired approach, using Feed-Forward SNN (FFSNN) and Recurrent Spiking Neural Network (RSNN) models, both of them implemented in software and

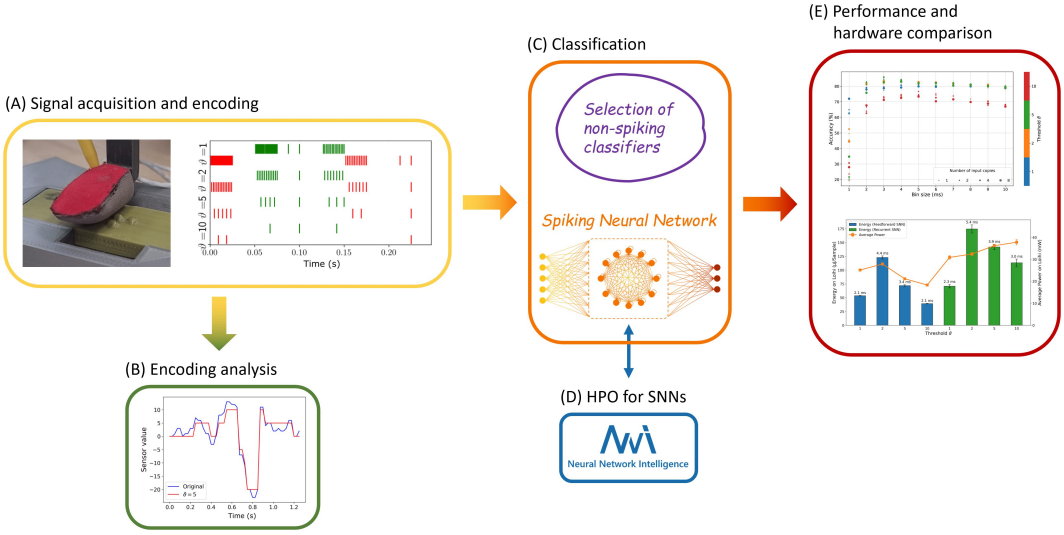


Fig. 3: The workflow is summarized in five steps. Dataset acquisition and signal encoding (a) with analysis of information content and reconstruction loss (b). Different non-spiking classifiers are identified and employed to produce references for the proposed RSNN (c), with the latter undergoing a hyperparameter optimization (d). Finally, performances are evaluated, accounting for different metrics and hardware implementations.

hardware. For the hardware implementation, we selected the NVIDIA Jetson, a modular embedded GPU platform, and the Intel Loihi, a dedicated neuromorphic chip, and benchmarked them against standard classifiers in terms of classification accuracy, average power usage, energy consumption, and computation delay during inference.

Using a comprehensive analysis accounting for multiple aspects such as the information loss introduced by the encoding technique, the accuracy of the classifiers, and the power consumption on different hardware with different models, we demonstrate that Braille reading can be performed in a highly energy-efficient way by using event-based data and deploying SNN on dedicated neuromorphic hardware.

2 Methods

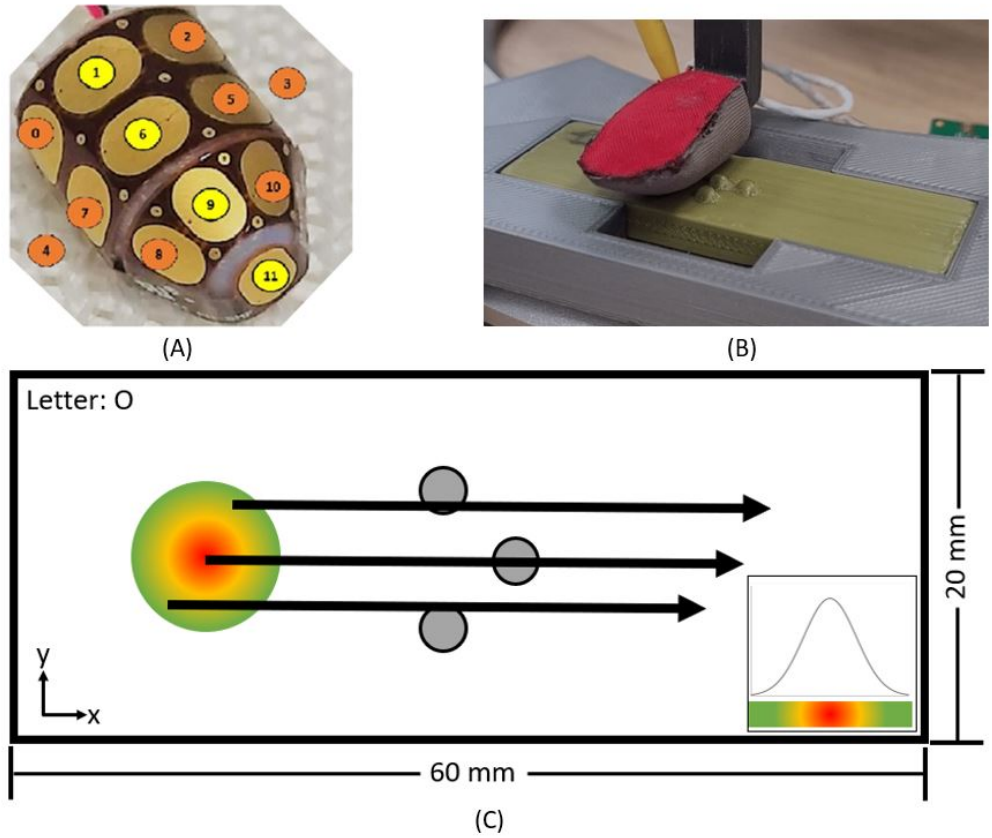


Fig. 4: The inner part of the fingertip is composed of 12 capacitance plates shown in (a), wrapped by a three-layer fabric and slid over the Braille letters with a constant sliding distance and velocity (b). The start position was varied following a Gaussian distribution (c).

2.1 The dataset

The Omega.3 robot was used to slide a sensorized fingertip [72] over 3D printed Braille letters from *'A'* to *'Z'* as well as *'space'* at a controlled speed and position [1].

The size of the Braille letter was chosen to match the spatial distribution of the fingertip so that the full letter can be detected by a single sliding movement. The sliding distance (15.5 mm), the sliding velocity (20 mm/s), and the distance to the flat surface of the plate were held constant. The start position varied following a Gaussian distribution to include spatial-temporal variability between each repetition, illustrated in Fig. 4c. Each letter was recorded 200 times with a sampling frequency of 40 Hz. The capacitance value is encoded in 8-bit, leading to a range from 255-0

[1]The robot is controlled and the sensor response is stored using YARP on a DELL XPS 15 laptop running Ubuntu 20.04 LTS.

with a minimal change of 1, with 255 being in rest and 0 for maximum load. For convenience, the encoding was inverted in software being 0 the resting state, and 255 the maximum load of the sensor capacity.

2.1.1 Event-based encoding

The objective of this work is to explore the potential of an end-to-end neuromorphic system for tactile perception with event-based communication (sensor level), asynchronous processing (hardware level) and spike-based computing (algorithmic level). However, there is to the best of our knowledge no available event-based tactile sensor today. Therefore, we emulate the output of such a sensor by encoding the frame-based data into temporally sparse streams of events (i.e. spikes) using a sigma-delta modulator ($\Sigma\Delta$ modulator) [73]. At threshold (ϑ) crossings, ON or OFF digital events are generated for increase or decrease of pressure, respectively [62], as indicated in Fig. 5a. Each original stream of frames is converted in an offline preprocessing step from a 12-taxel time sequence to 24 binary event-based channels, emulating an event-based tactile sensor.

The maximum re-sampling frequency is given by the precision of the stored 64-bit float variable. Given the sensor encoding regime and the sampling frequency, the precision of the conversion is enough to encode the data with a minimum change in the sensor value of $\vartheta = 1.02\mathrm{E} - 11$ without losing information from the frame-based data. A threshold value $\vartheta = 1$ corresponds to the highest implemented precision and no information loss. Increasing values correspond to increasing sparsity, lower data rate, and improved efficiency at the cost of lower accuracy and information loss, as shown in Fig. 5b.

2.2 Standard classifiers

Non-event-driven approaches such as linear classifiers (e.g. Support-Vector Machine (SVM)), time-series classifiers, and Long Short-Term Memory (LSTM) were used on frame-based signals available in the dataset as baselines for more traditional strategies independent of neuromorphic, event-based approach. Additionally, LSTM was used for event-based data as a benchmark for SNN, too.

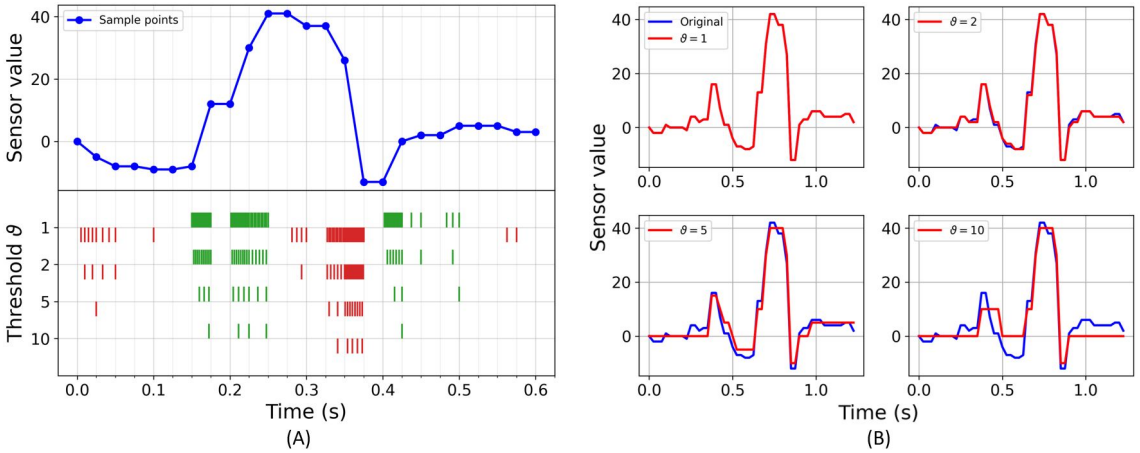


Fig. 5: Event-based encoding and reconstruction of a sample: (a) Sensor reading sequence of a sample letter along with spikes generated using sigma-delta modulation. The upper part represents 600 ms of a sequence of sensor readings of a single taxel during sliding. The bottom part shows the generated events for the ON (green color) and OFF (red color) channels for increasing threshold values, leading to decreasing numbers of events. (b) Reconstructed sequence from event-based data compared with the original sequence for a full letter sequence. Each plot represents the reconstruction with a different threshold of the same frame-based sequence. Increasing thresholds increase the compression, but also increase the reconstruction error.

2.2.1 Time-series classifiers for frame-based data

We used standard time-series classifiers proven to work with time-variant datasets: Fully Convolutional Network (FCN) [74], Residual Neural Network (ResNet) [74, 75], Encoder [76], Time-CNN (TCNN) [77], and Inception [78], available as implementation on GitHub [78, 79]. Please find more detailed information in the Supplementary Material 1.1

While these networks were specifically designed for time series, they are still based on standard feedforward structures (e.g. fully-connected layers, convolutional layers). Their lack of internal memory and recurrence requires that a whole time series is presented to the network all at once, instead of only the data from one timestep. Consequently, during inference, a buffer with the length of a time-series needs to be filled to perform classification. In case of multiple overlapping time windows, even more, buffers are required. This leads to memory overhead and classification delay compared to networks with a recurrent structure.

2.2.2 Long short-term memory

Given the time-dependent characteristic of the dataset, we implemented a recurrent neural network. This type of network, in contrast to feedforward networks, incorporates an internal loop

that allows temporal information to persist, and therefore is the natural choice for sequential data, but suffers from vanishing gradients for sequences characterized by long-term dependencies. Such problem is avoided by using LSTM architectures [80], where the cell state C_t holds long-term information. Additionally, LSTM can add or remove information from the cell state by *gates*.

The architecture chosen for the Braille dataset consists of a single layer LSTM with 228 hidden nodes, followed by a regular fully-connected layer of 228×27 output neurons that performs the classification, giving a total number of 225,975 trainable parameters. The choice was made to have a number of trainable parameters as close as possible to that of the best performing RSNN, obtained through a two-step Hyperparameters Optimization (HPO) procedure as described in the following, for a fair comparison. The method for calculating the number of trainable parameters is reported in Supplementary Material 1.2, Equ. S.1.

2.3 Spiking neural networks for event-based data

We designed a two-layer RSNN adopted from [81] and [82] to perform classification on the dataset encoded as an event-stream with four different thresholds, to achieve a quantitative comparison of the different possible strategies suitable to effectively deal with time-based Braille reading signals. We used the current-based (CUBA) Leaky Integrate and Firing Neuron Model (LIF) neuron model written in continuous form as

$$\tau_{mem}\frac{\mathrm{d}U_i^{(l)}}{\mathrm{d}t} = -(U_i^{(l)} - U_{rest}) + RI_i^{(l)} \tag{1}$$

with U_i being the membrane potential of neuron i (hidden state) in layer l, U_{rest} being the resting potential, τ_{mem} the membrane time constant, R the input resistance, and I_i being the input current defined as

$$\frac{\mathrm{d}I_i}{\mathrm{d}t} = \frac{i_I(t)}{\tau_{syn}} + \sum_j W_{ij}S_j^{(0)}(t) + \sum_j V_{ij}S_j^{(1)}(t) \tag{2}$$

with τ_{syn} being the synapse decay time constants, $S_j(l)$ the spike train of the jth neuron at the lth layer, W_{ij} being the forward, and V_{ij} the recurrent connection's weights.

$$I_i^{(l)}(t) = \alpha I_i^{(l)}(t-1) + \sum_j W_{ij} \cdot S_j(t) \tag{3}$$

$$U_i^{(l)}(t) = (\beta U_i^{(l)}(t-1) + I_i^{(l)}(t)) \cdot (1.0 - U_{reset}) \tag{4}$$

with $\beta = \exp(\frac{-Time\_bin\_size}{\tau_{mem}})$ being the voltage decay constant, $\alpha = \exp(\frac{-Time\_bin\_size}{\tau_{syn}})$ the current decay constant, and $I_i^{(l)}$ the synaptic input current from neuron i in layer l multiplied by the input resistance $R = 1\Omega$ for convenience and U_{reset} the reset after eliciting an event.

The error is propagated throughout the entire network, unrolled in time, using Backpropagation Through Time (BPTT). To perform supervised learning the feedforward and recurrent weight matrices W_{ij} and V_{ij} change following a given loss $\mathcal{L}$

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial \mathcal{L}}{\partial W_{ij}} \quad \text{and} \quad V_{ij} \leftarrow V_{ij} - \eta \frac{\partial \mathcal{L}}{\partial V_{ij}} \tag{5}$$

with the learning rate η. To use a binary step function $\Theta(x)$ in the forward pass (inference), whose derivative is zero everywhere except the zero crossing where it becomes infinite, we use the partial derivative (the gradient) of a fast sigmoid function $\sigma(x)$

$$\sigma(U_i^{(l)}) = \frac{u_I^{(l)}}{1 + \lambda |U_i^{(l)}|} \tag{6}$$

in the backward pass (training), the surrogate gradient, and prevent vanishing issues. Whereas $\Theta(x)$ is invariant to multiplicative re-scaling, $\sigma(x)$ needs the introduction of the scale parameter λ being part of the hyperparameter optimization.

To compute the gradients we are using the capabilities to over-loading the derivative of spiking nonlinearity with a differential function in custom PyTorch [82, 83].

For the loss, we apply the cross entropy to the active readout layer $l = L$. For data with N_{batch} samples and N_{class} classes it is formalized as

$$\mathcal{L} = -\frac{1}{N_{batch}} \sum_{s=1}^{N_{batch}} 1(i = y_2) \cdot \log \left\{ \frac{\exp\left(\sum_{n=1}^{T} S_i^{(l)}[n] \right)}{\sum_{i=1}^{N_{class}} \exp\left(\sum_{n=1}^{T} S_i^{(l)}[n] \right)} \right\} \tag{7}$$

whereas n stands for the time step. At last we have to define the $\mathcal{L}_1$ and $\mathcal{L}_2$ regularization loss function.

$$\mathcal{L}_1 = \frac{S_L}{N_{batch} + N} \sum_{s=1}^{N_{batch}} \sum_{i=1}^{N} \max\left\{ 0, \frac{1}{T} \sum_{n=1}^{T} S_i^{(l)}[n] - \theta_l \right\} \tag{8}$$

representing a per neuron lower threshold spike count regularization with strength s_l and threshold θ_l, and

$$\mathcal{L}_2 = \frac{S_U}{N_{batch}} \sum_{s=1}^{N_{batch}} \left[\max\left\{ 0, \frac{1}{N} \sum_{i=1}^{N} \sum_{n=1}^{T} S_i^{(l)}[n] - \theta_u \right\} \right]^2 \tag{9}$$

being an upper threshold mean population spike count regularization with strength s_u and threshold θ_u. Finally, the total loss is summarized by

$$\mathcal{L}_{\text{tot}} = \mathcal{L} + \mu_1 \mathcal{L}_1 + \mu_2 \mathcal{L}_2 \tag{10}$$

with μ as a scaling factor and minimized using the Adamax optimizer [84].

To implement and simulate SNN based on this model in PyTorch, we need to account for a time binning step for the input event stream. Although aiming to work with asynchronous and sparse event-based data, fixed frame lengths had to be defined to properly simulate algorithmic time steps in the domain of clock-driven, conventional hardware like CPUs and GPUs. Time binning was performed by subdividing the time extracted from the signal recordings (T_{rec}) into T chunks, with T defined as $T = \text{int}(T_{rec}/time\_bin\_size)$ and the quantity $time\_bin\_size$ introduced as an additional hyperparameter of the HPO. Then, by iterating over the encoded signal with a stride equal to $time\_bin\_size$, a value of 1 was assigned whenever at least one spike was found, otherwise a 0. The winning neuron in the output layer is the one with the highest spike count.

2.4 Hyperparameter optimization

For each event stream produced from the original frame-based signal by applying a specific threshold value, a tailored RSNN was obtained by adapting the parameter optimization procedure introduced in [85]. The HPO was performed by means of the *Anneal* algorithm in the Neural Network Intelligence (NNI) toolkit, using the parameters listed in Tab. 1, over 600 trials. To partially mitigate the impact of local minima [86], two evenly spaced random reinitializations of the tuner were performed during each experiment. An 80/20 train-test split was used and all trials were composed of 300 training epochs with intermediate results, for both training and test, at the end of each epoch. Test accuracy was defined as the optimization objective of the HPO experiments, and its highest value was extracted at the end of each trial. The choice of selecting test accuracy as a reference value to be optimized was taken to account for possible overfitting.

Following the annealing-based procedure, we performed a further exploration of a portion of the initial search space through a grid search on the two most relevant hyperparameters from the energy consumption perspective, namely $time\_bin\_size$ and nb_input_copies since they determine the number of operations that need to be computed per inference.

As the outcome of such a two-step HPO procedure, an optimized network for each threshold value used in the sigma-delta encoding was obtained. All of these RSNNs were composed of a recurrent, fully connected hidden layer containing 450 LIF neurons and an output layer composed of 28 LIF neurons. The number of output neurons was defined to account for an extra class, in addition to the 27 defined by the letters, suitable to identify, given future possible online implementations, edge cases such as missing contact between the fingertip and the letters. The input layer was instead part of the optimization, with its number of input neurons defined as $2 \cdot n\_taxels \cdot nb\_input\_copies$ where 2 covers the event polarity, n_taxels was given by the 12 sensors in the robotic fingertip, and nb_input_copies was to be optimized. A batch size of 128 and a learning rate $\eta = 0.0015$ were adopted for all the networks.

2.5 Hardware implementation

Beyond the algorithmic evaluation, we determined key performance metrics that are relevant to real-world deployment by implementing the networks on different hardware platforms. These

Table 1: Description of the hyperparameters included in the search space for the HPO procedure.

Hyperparameter	Description
scale, λ	Steepness of surrogate gradient
time_bin_size	Time binning of the encoded input
nb_input_copies	Copies of the encoded signals provided to the input layer
tau_mem, τ_{mem}	Decay time constant of the membrane
tau_ratio	Ratio between the membrane and the synapse (τ_{syn}) decay time constants
fwd_weight_scale	Scaling factor for weight initialization of the forward connections (W_{ij})
weight_scale_factor	Scaling factor for weight initialization of the recurrent connections (V_{ij})
reg_neurons, μ_1	Scaling factor for the regularization on the number of spikes per neuron
reg_spikes, μ_2	Scaling factor for the regularization on the total number of spikes

metrics covered power usage, energy consumption, and computational delay, which allowed us to conclude the deployment feasibility in real-world scenarios. Considering the platform-related factors of high integration and availability, and our ultimate goal of deploying the algorithms in a real-world environment on robots, we targeted the NVIDIA Jetson Xavier NX, a commercially off-the-shelf available computing platform equipped with a System-on-Chip (SoC) that integrates a CPU and GPU, and the Intel Loihi, a neuromorphic processor dedicated to accelerating SNN.

2.5.1 NVIDIA Jetson Xavier NX

The NVIDIA Jetson is a product family of compact and embedded computation platforms mainly targeted toward edge AI. The Xavier NX is the most powerful model among its compact 260-pin SO-DIMM modules. Despite being a general-purpose platform, it is similar to common machine learning workstations in terms of architecture and software. We used it to run all algorithms and compare the different standard time-series classifiers, as well as evaluate the differences between conventional algorithms and event-based algorithms on off-the-shelf hardware. Furthermore, the inference metrics give an outlook on what performance can be expected during deployment with the same hardware.

Execution time was measured by using the native functionality offered by the Linux Operating System (OS). Power usage was measured by utilizing the module's onboard INA3221 power monitor, polling the system's main power rail with a fixed interval of 50 ms. The power monitor

also measures a CPU/GPU and a SoC power rail. But due to the lack of public information on what components exactly these rails supply, and also due to a productive application requiring a full system instead of just single core components, the main power rail was chosen for comparison.

The general procedure for measuring the performance metrics was performed as follows: initially, the whole dataset was loaded into memory, followed by the loading of the model and its trained weights. Next, a warmup of the system was performed by letting the model predict the whole dataset in batches of 64 for six times. Its purpose was to fill the caches and to avoid additional library load times during the actual inference. Afterward, the recordings of the execution time and power values were started, which was immediately followed by the inference. Like in the warmup, the whole dataset was predicted six times, but with the difference that a batch size of 1 was used to simulate how the system would behave in a real-world scenario where single samples are predicted consecutively. For our SNN the number of samples during warmup and inference was reduced to 750 each due to timing constraints. Finally, after the inference was done, the recordings were stopped and the following metrics were evaluated:

- Inference time (i.e. computational delay) per sample, which is the total inference time divided by the number of samples processed.
- Minimum, Maximum, and Average power usage over total inference time as well as per sample.
- Total and per sample energy consumption. The total energy is calculated by multiplying and accumulating each power measurement with a polling interval of 50 ms. Energy per sample is given by dividing the total energy by the number of samples processed.

Furthermore, to get more significant results, the above procedure was repeated three times per network to eliminate possible outliers and also performed for each available power mode on the Jetson, whereas the results reported are from power mode 4.

2.5.2 Intel Loihi

Intel's Loihi [87] is a fully digital neuromorphic research processor. Each Loihi chip hosts 128 neuron cores, where every neural core can run up to 1,024 CUBA LIF neurons by

time-multiplexing. The Loihi neuron's equations for the current and voltage compartments are

$$I_i(t) = I_i(t-1) \cdot (2^{12} - \delta_i^I) \cdot 2^{-12} + 2^6 \cdot \sum_j w_{ij} \cdot s_j(t) \tag{11}$$

$$U_i(t) = U_i(t-1) \cdot (2^{12} - \delta_i^U) \cdot 2^{-12} + I_i(t) \tag{12}$$

where t is the algorithmic time step, $I_i(t)$ and $U_i(t)$ are the current and voltage of neuron i, δ_i^I and δ_i^U are the current and voltage decay constants, w_{ij} is the synaptic weight from neuron j to i and $s_j(t)$ is the spike state (0 or 1) of neuron j.

Each Loihi neuron core supports arbitrary connection topologies as long as the capacities of the in-core memories for storing axons and synapses are not exceeded. The neuron cores are parallel and distributed with local on-chip SRAMs to store the network state and configurations. The neuron cores are fully asynchronous, performing synaptic accumulation only when there is an input event, which highly benefits from the spatio-temporal sparsity of event-based sensors and encoding. The algorithmic time step in the entire Loihi system is maintained by a distributed handshaking mechanism called *barrier synchronization*. In addition, each Loihi chip has 3 synchronous embedded x86 cores also taking part in the barrier synchronization. The x86 cores run C code and are used to monitor and interact with the SNN running on the neuron cores, handling data IO between the on-chip asynchronous neuron cores and off-chip devices, and optionally synchronizing the algorithmic time steps duration (in physical time) to integrate the chip with a sensor.

We developed a solution to deploy the trained networks from our PyTorch implementation to Loihi (PyTorch2Loihi). First, we export the neurons' hyper-parameters and the trained synaptic weights from PyTorch in an HDF5 file by taking into account the Loihi hardware specifications and constraints as follows:

- **Current and voltage decays constants:** we calculate the Loihi decay constants δ^I and δ^U from the PyTorch time constants τ_{syn} and τ_{mem} respectively, with

$$\delta = \text{int}\left(2^{-12} - 2^{-12} \cdot e^{\frac{-Time\_bin\_size}{\tau}}\right) \tag{13}$$

- **Synaptic weights and neuron threshold:** Loihi supports up to 8-bit fixed point weights. To minimize the effect of quantization, we quantize the weights from PyTorch training into 256 states and adjust the weight scaling factor and threshold scaling factor to have the same overall effect of an input spike. The weight scaling factor w_{scale} and the weight quantization scheme is described by

$$w_{scale} = \text{int}\left(\frac{256}{\max|w|}\right) \tag{14}$$

$$w_{Loihi} = \text{quantize}\left(w, \text{step} = 2\right) \cdot w_{scale} \tag{15}$$

$$\theta_{Loihi} = 2^6 \cdot \theta \cdot w_{scale} \tag{16}$$

After the network is deployed, we run the inference with the event-based tactile data and first quantify the classification accuracy, then, measure the energy consumption and computation delay. The test inference was made by injecting the input events of all samples of the test set in a continuous flow, where samples are separated by a blank time of 100 algorithmic time steps where the neurons' currents and voltages decay to zero. The output spikes are gathered throughout the duration of the inference, and then the classification accuracy is calculated offline.

Loihi system boards include voltage regulators and power telemetry which can be used to measure the total power consumption of the Loihi chip while a model is running. The power measurements can be combined with timing information recorded by the on-chip x86 cores during model operation to estimate energy consumption. NxSDK exposes a high-level user interface for measuring power, energy, and timing when a workload is running. We used the interface to benchmark the performance of our SNN models on Loihi.

3 Results

3.1 Encoding analysis

To characterize the event-based datasets, we reconstructed the temporal sequences out of the event streams and compared the results with the original frame-based signal. Additionally, we performed the same analysis by taking into account the time binning step used to prepare the data for clock-driven computation as described in 2.3. Results are summarized in Tab. 2, where the mean number of events, the compression ratio γ with respect to the encoded data at $\vartheta = 1$ and the reconstruction Mean Squared Error (MSE) values ε are reported both, before and after the binning step for each threshold.

3.1.1 Signal reconstruction before time binning

From the event stream, the signal was reconstructed, starting from zero, by increasing or decreasing, for every event, according to the polarity ON or OFF, by an amount equal to the threshold used in the encoding. Fig. 5b shows the reconstruction values for one sample and taxel at different threshold values. The compression ratio γ is defined as the number of events at $\vartheta = 1$ divided by the number of events at each threshold value. The reconstruction error ε is the MSE between the original sequence and the reconstructed frame-based sequence for each of the event-based datasets.

The analysis of the reconstructed frame-based signal revealed that with increasing thresholds the number of events dramatically decreases, increasing the reconstruction error. However, the compression ratio γ increases with a higher rate than the reconstruction error ε, showing a sparsity gain of the event-based dataset at the cost of information content.

3.1.2 Signal reconstruction after time binning

Running SNN in PyTorch requires the introduction of time bins. To quantify the impact of the time binning on the different encoded datasets, we counted the total number of events given in each dataset after the time binning. The total number of events for a given encoding threshold is always higher than the total number of events for a given lower encoding threshold, regardless

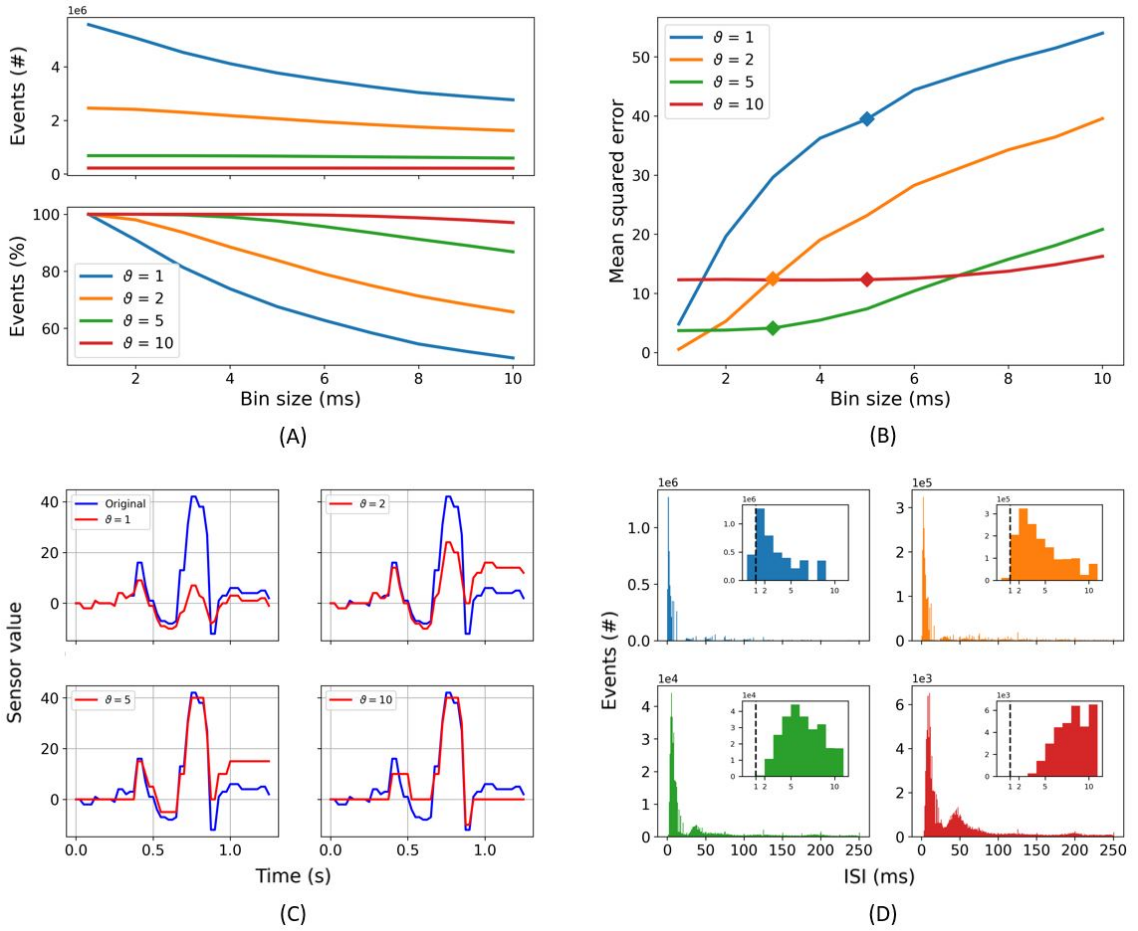


Fig. 6: Spike encoding: (a) Total number of events counted in the whole dataset in dependence of the selected threshold and $time\_bin\_size$ is shown in the top panel, while the relative amount of events found in the dataset relative to $time\_bin\_size = 1$ is reported in the bottom one. Increasing the encoding threshold reduces the number of events significantly, whereas encoding with lower thresholds is much more affected by increasing $time\_bin\_size$. The amount of events lost is 50.37 % for $\vartheta = 1$, 34.25 % for $\vartheta = 2$, 13.17 % for $\vartheta = 5$, and only 2.92 % for $\vartheta = 10$. (b) MSE values of signal reconstruction after time binning as a function of the $time\_bin\_size$ for each encoding threshold. The $time\_bin\_size$ resulting from the HPO and grid search used to preprocess the event stream are highlighted by the markers. (c) Shows the reconstruction of the frame-based signal from the event stream for all given thresholds after time binning with a bin size of 5 ms. (d) Number of events as a function of the Inter-Spike Interval (ISI) with fixed $time\_bin\_size$ as reported in Tab. 2, with the same colour coding as in (a) and (b). The insets show the detail at ISI values equal to the $time\_bin\_size$ used in this work, highlighting the minimum temporal resolution of 1 ms with the vertical dashed line. Spikes below 1 ms in percentage of total number of spikes: 8.22 % for $\vartheta = 1$, 0.45 % for $\vartheta = 2$, 0 % for $\vartheta = 5$ and $\vartheta = 10$.

Table 2: Characterisation of event-based encoding for each of the generated datasets at different threshold values. Compression ratio γ is defined as the number of events at perfect encoding ($\vartheta = 1$) divided by the number of events at each higher threshold value. Values of the reconstruction error ε are calculated per reconstructed frame by MSE. Mean events are calculated per sample. All $time\_bin\_size$, introduced to prepare the data for clock-driven computation, follow the results reported in 3.4 from the two-step HPO procedure.

	Before time binning			**After time binning**			
Threshold (ϑ)	**Events**	**Comp. ratio** (γ)	**MSE** (ε)	**Bin size** (ms)	**Events**	**Comp. ratio** (γ)	**MSE** (ε)
1	87.6	1	0	**5**	58.1	1.5	39.5
2	38.0	2.3	0.4	**3**	35.5	2.5	12.5
5	10.5	8.3	3.7	**3**	10.5	8.3	4.1
10	3.4	25.7	12.3	**5**	3.4	25.7	12.3

of the time binning, as shown in the top panel of Fig. 6a. The higher the encoding threshold, the lower the impact of the time binning. For the encoding threshold $\vartheta = 1$, the total number of events counted in the dataset for a $time\_bin\_size$ equal to 1 ms is halved with increasing $time\_bin\_size$. For the encoding threshold $\vartheta = 2$, we still have a loss of 35 %, whereas for higher encoding thresholds $\vartheta \geq 5$ the impact of the time binning decreases close to or below 10 % and most of the events are perceived, as shown in the bottom panel of Fig. 6a.

The same reconstruction of the frame-based signal from the event stream as described in 3.1.1 was performed for every possible $time\_bin\_size$, included in our HPO and grid search, of the event stream for each threshold value. The reconstruction error ε depends, on the one hand, as discussed above, on the encoding threshold, and on the other hand on the introduced time binning. Results are shown in Fig. 6b with markers at the $time\_bin\_size$ selected after the HPO and grid search to run the SNN for each encoding threshold. The smallest reported reconstruction error ε is 0.55 for threshold $\vartheta = 2$ and 1 ms $time\_bin\_size$. We see a great increase in the reconstruction error ε with increasing $time\_bin\_size$. That increase is even more drastic for threshold $\vartheta = 1$ starting at a higher reconstruction error ε of 4.82. The higher reconstruction error ε can be explained by the loss of events when multiple of them fall into a single time bin, leading to an increase in the reconstruction error by introducing an accumulating offset, as shown in Fig. 6c. Additionally, the discriminative power of amplitudes is lost, leading to similar amplitude for small and high changes after the reconstruction. A further analysis unveiled, that 8.22 % of the ISI for threshold $\vartheta = 1$ are below 1 ms which is the smallest $time\_bin\_size$ used, whereas

for $\vartheta = 2$ only 0.45 % are below 1 ms, as summarized in Fig. 6d. For all higher thresholds ($\vartheta > 2$) no ISI are below 1 ms. The higher the threshold, the higher the majority of ISI due to the increasing sparseness. The impact of time binning has decreasing impact, but the error introduced by the higher encoding thresholds is becoming more relevant. Overall, the higher thresholds are more resilient to the impact of $time\_bin\_size$, but less able to reflect the temporal dynamics.

3.2 Standard classifiers for frame-based data

3.2.1 Linear classifier

The recorded dataset encodes information in spatio-temporal patterns. Before this dataset is used for pattern recognition, it is important to know whether only spatial information or both spatial and temporal information in this dataset needs to be considered for character recognition. To investigate the significance of spatial and temporal information for the Braille letter classification task, we applied SVM with a linear kernel. To eliminate only the temporal information from the data by keeping the spatial information intact, we computed the mean of all frames with respect to time for each channel. In this way, we got one data point for each channel which is time independent and each channel represents the spatial location of each sensor, referred to as 'time collapsed data'. Data keeping both, spatial and temporal information intact, is referred to as 'raw data'. We applied a one-vs-rest multiclass classifier on the raw and the time-collapsed data for five cross-validated splits. We achieved (86.7 $\pm$ 0.77) % and (72.3 $\pm$ 1) % accuracy for raw and time-collapsed data respectively. To further investigate the temporal nature of the data we iteratively increased the number of frames taken into account from the first to the total number of frames in the frame-based data, in every iteration the data has been reduced to the first 12 principle components found by Principal Component Analysis (PCA) to always consider the same dimensionality of the predictors in the classifier.

In Fig. 7a the results of this procedure are shown, with a clear increase in the accuracy with three significant phases at 0.5 s and 0.8 s before it finally saturates after 1s at (86.7 $\pm$ 0.77) %. We can identify the three phases in the sliding procedure, namely: the first contact with the dot pattern between 0.1 s and 0.5 s, the first contact with the second row of the pattern between 0.5 s and

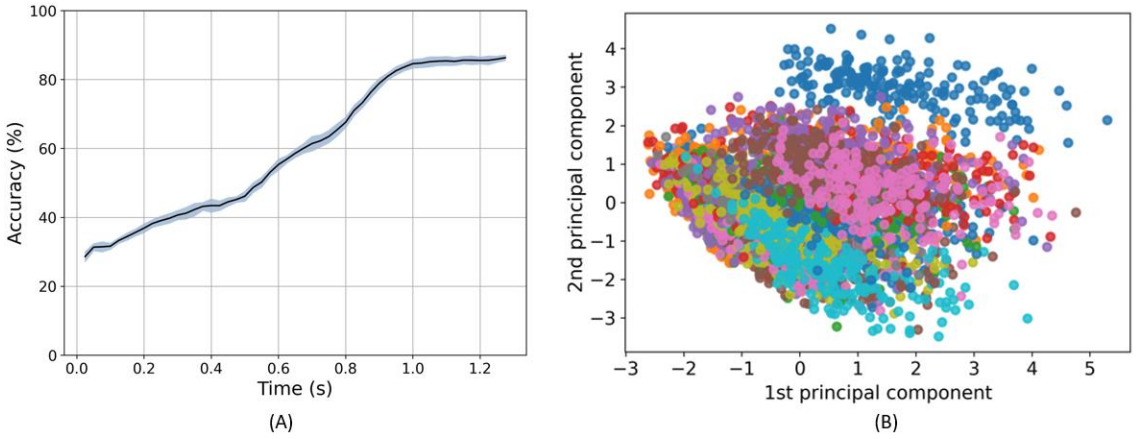


Fig. 7: (a) Dependency of SVM performance in regard to the first 12 principle components extracted using PCA provided with an increasing number of frames. An increase in the number of frames results in an increase in the performance close to saturation in 1s at (86.7 $\pm$ 0.77) %. (b) Dimensionality reduction (PCA) with 2 components applied to the spatio-temporal sequences of the frame-based dataset. Each of the colors in the visualization represents a category (letter) in the dataset.

0.8 s, and the end of the pattern after 1 s, by comparing Fig. 4c and Fig. 7a.

Similarly, for the event-based dataset, we investigated the discrimination power of the spatial components by removing the temporal dimension of the dataset. In this case, we built the classifier with predictors as the summation of all the events for each taxel and channel (ON/OFF) resulting in 24 predictors per letter in the dataset. Fig. 7b shows the first two principal components after applying PCA to the spatial predictors displaying an overlapping in data categories. The result of the trained classifier drops from (58.94$\pm$1.15) % for the event-based data with all time bins as predictors to (47.7$\pm$1.48) % for the predictors that only account for the sum of spatial information. This analysis confirms the intuition that the temporal information of the signal is important for the character discrimination tasks.

3.2.2 Time-series classifier and LSTM

We trained the time-series classifiers and the LSTM network with an 80/20 train-test split for 300 epochs and averaged the results over three runs per network. The resulting test accuracy as well as the network's respective number of trainable parameters are shown in Fig. 8.

ResNet, Inception and FCN performed at comparable level. They all achieved 100 % in training accuracy and reached comparable test accuracy of (98.2 $\pm$ 0.2) %, (97.8 $\pm$ 0.2) % and

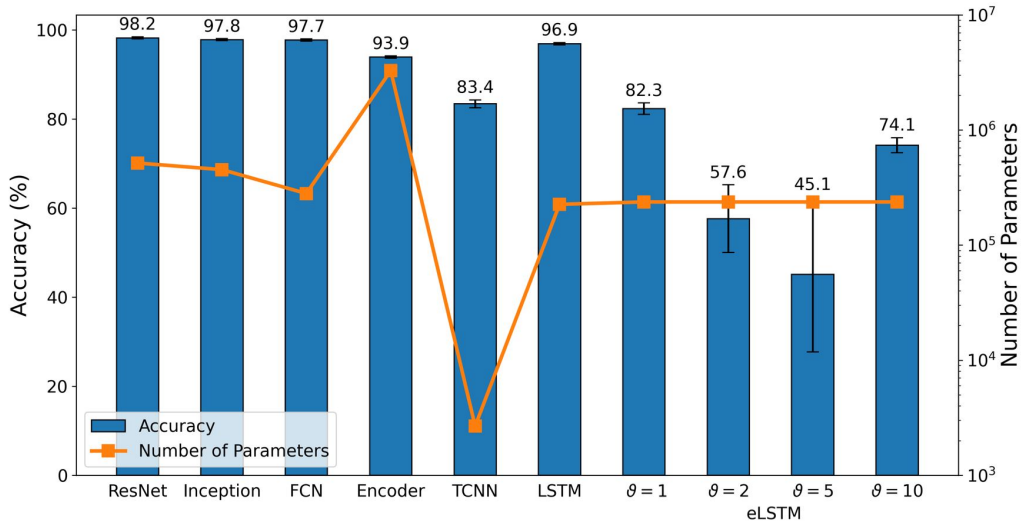


Fig. 8: Test accuracy and number of trainable parameters of standard classifiers after training for 300 epochs and averaging over three runs. eLSTM refers to LSTM with event-based input.

(97.7 ± 0.3) %, respectively. Similarly, their number of parameters turns out to be comparable too, with ResNet $(516k)$ and Inception $(452k)$ close to each other and FCN $(280k)$ slightly smaller but in the same order of magnitude.

The Encoder architecture reached a training accuracy of (99.8 ± 0.1) % but the drop in test accuracy was greater, resulting in (93.9 ± 0.2) %. This deviation is an indication of overfitting and can possibly be explained by the fact that the network's number of parameters is one order of magnitude higher than the one of the former mentioned networks. This high amount of parameters can lead to a training outcome where the network memorizes the training data and conversely generalizes worse than it would with fewer parameters.

LSTM is the only recurrent architecture in our selection. It performs sequential processing from a streaming input without buffering the data. This reduces the delay and memory footprint which is important for embedded systems. Compared to the previous networks, it achieved again a training accuracy of 100 % and the test accuracy did not drop as significantly as for the Encoder and resulted in (96.9 ± 0.3) %, suggesting less overfitting.

TCNN performed worst among all selected networks and only achieved (88.5 ± 0.8) % and (83.4 ± 1.3) % for training and test accuracy, respectively. However, it uses a greatly reduced number of parameters, containing only 2,673, which is two orders of magnitude lower than

ResNet, Inception, FCN and LSTM, and even three orders of magnitude lower than the Encoder network.

3.3 LSTM with event-based input (eLSTM)

As a benchmark for the results provided by the optimized RSNN, LSTM was adopted for event-based data, using the same architecture as described in Section 2.2.2, the same train-test split as described in 3.2.2 and the same data preprocessing as for the RSNN, meaning sigma-delta encoding and time binning. Results are reported in Fig. 8. A clear impact of the encoding threshold on the eLSTM can be observed, with $\vartheta = 1$ providing the best accuracy values. Compared to the optimized RSNN, while similar results are found when using $\vartheta = 1$, significantly lower performances are achieved with $\vartheta = 2$ and $\vartheta = 5$, which also show a drastically increasing standard deviation. For $\vartheta = 10$, results are again similar to those reported for the RSNN, but a higher standard deviation is observed in this case as well. A similar behaviour across the different threshold values can be observed in Fig. 10 as well, where $\vartheta = 2$ and $\vartheta = 5$ show worse performances, in terms of inference time, compared to $\vartheta = 1$ and $\vartheta = 10$. One possible explanation is the same $time\_bin\_size$ for $\vartheta = 1$ and $\vartheta = 10$ with 5 ms, compared to 3 ms for the other encoding thresholds. This leads to $3/5$ the number of time steps to compute, which the LSTM seems to benefit from.

3.4 Spiking neural networks

During the two-step HPO procedure, the main objective for the optimization was the classification accuracy, but we also monitored the Time-to-Classify (TTC) and the power consumption, with the latter separately reported in Section 3.5. In the perspective of an online implementation of the proposed RSNN, an informative figure of merit to be accounted for is the minimum temporal length of the input needed for successful classification. To this aim, we defined TTC as the portion of the signal needed for successful classification with respect to the full acquisition time of the Braille letter, which was fixed to 1.35 s, due to the fixed sliding speed. Additionally, in order to account for possible corner cases in the online implementation, like 'no contact' situations, we included one extra class leading to 28 classes in total.

Table 3: Optimized values of the hyperparameters, for each encoding threshold, following grid search.

	Threshold (ϑ)			
	1	**2**	**5**	**10**
scale	5	15	10	10
time_bin_size (ms)	5	3	3	5
nb_input_copies	2	8	4	2
tau_mem (ms)	60	50	70	70
tau_ratio	10	10	10	10
fwd_weight_scale	1	1	1.5	4
weight_scale_factor	1e-2	2e-2	3.5e-2	1.5e-2
reg_spikes	4e-3	1.5e-3	1e-3	1.5e-3
reg_neurons	1e-6	0	0	0

The parameter space after the NNI optimization, as well as after the grid search, for each threshold, shows no significant trends. The complex interaction of different parameters leads to a variety of local optima resulting in comparable test accuracy. Looking only at the trials with the best classification accuracy for different encoding thresholds, reported in Tab. 3, gives the same picture. Only the forward weight scale, fwd_weight_scale, seems to be constantly increasing with increasing thresholds. The membrane potential time constant τ_{mem} has only slight variations, again with no clear trend, and the tau_ratio, describing the relation of τ_{mem} and τ_{syn}, is constant. Seeing similar membrane time constants indicates their dependencies on the spatial-temporal properties of the data despite the encoding threshold. A constant membrane-to-synapse time constant ratio in contrast shows to be an optimal relation between neuron and synapse dynamics for this task.

Fig. 9 shows a summary of the RSNN classification accuracy after grid search optimization, while in Supplementary Material Fig. S2 the hyperparameters exploration during the first step of the HPO procedure is reported. From all the explored combinations of $time\_bin\_size$ and nb_input_copies for all the encoding thresholds employed, the best configuration in terms of accuracy turned out to be the one adopting an encoding threshold of $\vartheta = 5$, a $time\_bin\_size$ of 3 ms and 4 input copies. Nevertheless, such RSNN configuration did not provide the strongest reliability in terms of repeatability. As shown in Fig. 9b, the standard deviation of the provided results is larger than the one observed for other configurations. Fig. 9b shows that the best configuration found for an encoding threshold $\vartheta = 2$ resulted in a mean accuracy as high as the

one for an encoding threshold $\vartheta = 5$ but with a significantly reduced standard deviation: (80.9 $\pm$ 0.3) % test accuracy compared to (80.9 $\pm$ 1.9) %.

Looking at the accuracy performances from this twofold perspective, it is hence possible to identify as the best configuration the one employing the encoding threshold $\vartheta = 2$ with a $time\_bin\_size$ of 3 ms and a number of copies equal to 8. The overall performance of the highest threshold ($\vartheta = 10$) is the worst. The analysis of accuracy results provided an additional insight also, reported in both Fig. 9a and Fig. 9b, highlighting that the encoding threshold and $time\_bin\_size$ can have a significant impact. Particularly, they induce a similar behavior of test accuracy: after an initial growth leading to a maximum, a further increase of one of them produces a deterioration of the classification performance. In contrast to the findings regarding the preservation of events concerning the $time\_bin\_size$ for different thresholds discussed in 3.1, the accuracy for higher encoding thresholds decreases most for higher $time\_bin\_size$. Comparing the network performance of the RSNN and the FFSNN, shown in Fig. 9b, we observe a constant decrease for the FFSNN with increasing thresholds, but an almost constant performance for the RSNN up to $\vartheta = 10$, where it initially starts to drop off.

Targeting an online hardware implementation, next to classification performance, the energy efficiency needs to be considered too. From this point of view, the encoding threshold of 1 is the most promising candidate using a small number of input copies and a greater $time\_bin\_size$, leading to a significantly lower energy footprint at comparable performance. Regardless of the hyperparameters, the TTC is equal throughout all conditions and the whole time series is needed for the best classification performance, similar to the findings using the linear SVM, reported in 3.2.1. To validate the use of the extra class, we also compared these results with an implementation based on 27 classes only. Such analysis revealed that similar results are achieved in both cases, thus showing that the additional 28th class does not have a detrimental impact on the classification performances.

3.5 Hardware implementation

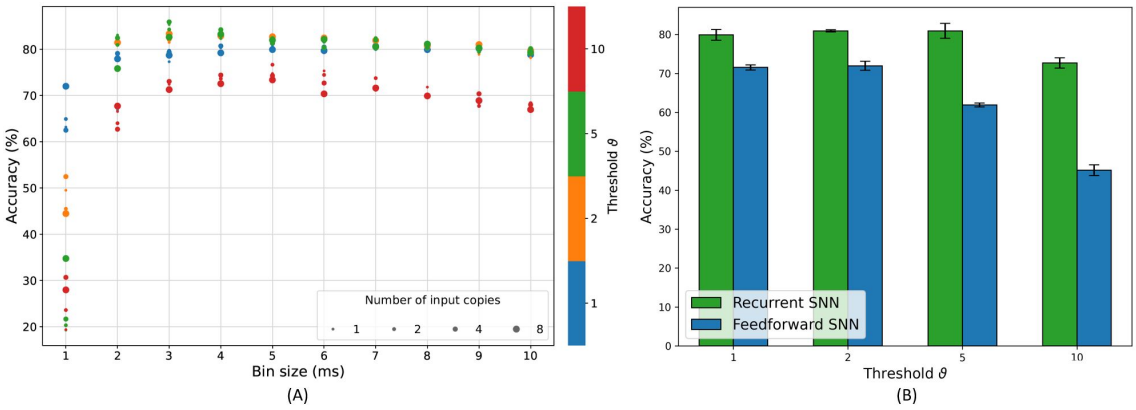


Fig. 9: Summary of the accuracy performances of RSNN and FFSNN resulting from the grid search exploration in the two-step HPO procedure. (a) Best test accuracy results achieved with the RSNN for all the combinations of $time\_bin\_size$ and nb_input_copies. (b) Mean and standard deviation of the accuracy results of both the FFSNN and the RSNN with the best parameters for each encoding threshold.

3.5.1 NVIDIA Jetson embedded GPU

The energy consumption, average power usage, and inference time of the standard classifiers running on the NVIDIA Jetson[2] are shown in Fig. 10. We consider energy consumption the main metric as it includes both power usage and inference time per sample. Average power usage gives insight into what power budget would be required to achieve certain inference times. While energy and power consumption are important metrics for battery-powered applications, inference time is crucial in applications with real-time constraints.

When comparing these results with the parameter counts in Fig. 8, some similarities can be found. ResNet, Inception, and FCN, which are comparable in terms of accuracy and parameter count, had nearly the same average power usage during inference. This means that their energy consumption is directly proportional to their inference time. In the case of Inception and its parameter count, we expected the energy consumption to lie between ResNet and FCN, but it exceeded both. While parameter count is not directly related to computational complexity, another explanation for this observation could be that Inception uses operations that are less optimized or not accelerated by the GPU. The similarities continue with the Encoder, where average power usage jumped and the parameter count increased by one order of magnitude, compared to the former three networks. Judging only by the parameter count, we expected energy consumption to be even higher. The

[2]Energy and timing measurements for the Jetson were obtained on an NVIDIA Jetson Xavier NX Developer Kit running the NVIDIA JetPack 4.6.1 SDK, containing Ubuntu 18.04.6 LTS, L4T 32.7.1, TensorRT 8.2.1, and CUDA 10.2

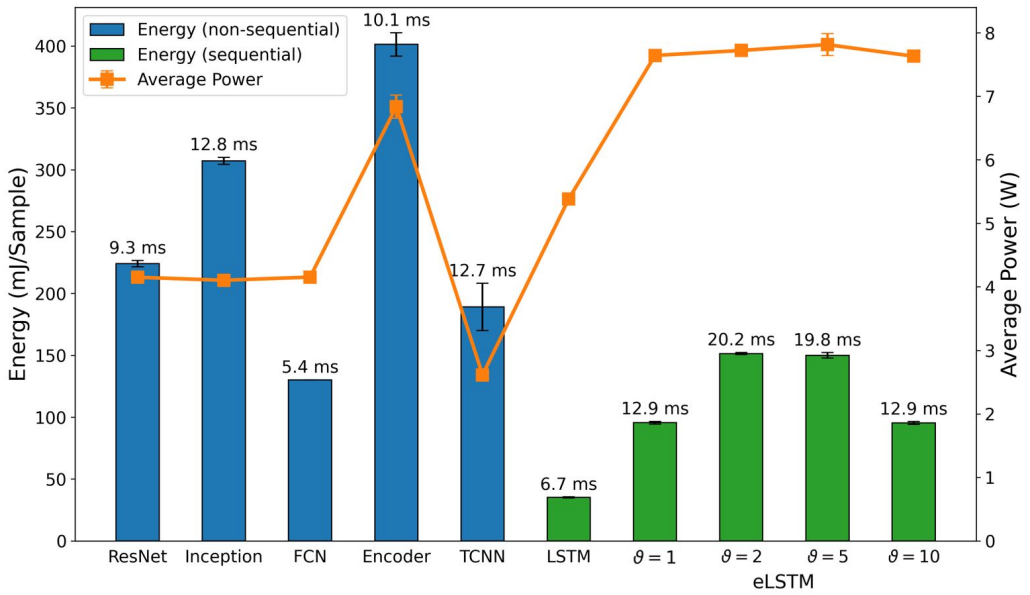


Fig. 10: Comparison of inference metrics from the standard classifiers for frame-based data in terms of energy consumption and average power usage as measured on an NVIDIA Jetson Xavier NX. eLSTM refers to LSTM with event-based input. The label on top of each bar shows the inference time per sample on the corresponding network.

difference, especially compared to Inception, is not as great as expected, though. Considering the higher average power usage, it had better GPU utilization and therefore could benefit from an overall higher acceleration. Inference time also supports this assumption, as it was lower on the Encoder than on Inception.

Amongst all standard classifiers, the results for the LSTM with frame-based input are the most notable. Despite being a sequential network, which we expected to be slower compared to the standard classifiers due to its iterative and recurrent nature, it achieved the fastest inference time and lowest energy consumption. But this came at the cost of the second highest power consumption. Similar to the Encoder, this suggests that the network can benefit from a high GPU utilization or general acceleration of internal operations.

For the LSTM with the event-based input, further referred to as eLSTM, we found that the energy per sample and the inference time reflect the number of time steps processed with a ratio of 5 to 3 which is given by the time binning applied for each encoding threshold. Similar $time\_bin\_size$ for $\vartheta = 1$ and $\vartheta = 10$ with 5 ms resulting in 270 time steps and for $\vartheta = 2$ and $\vartheta = 5$ with 3 ms resulting in 450 time steps have been used. Interestingly this ratio does not hold when compared to the frame-based signal with 54 timesteps. The reason for that might be the nature of the data

with float numbers for the frame-based data and integer numbers for the event stream. The power consumption for all eLSTMs is nearly the same.

Lastly, the TCNN performed poorly when putting it alongside the other networks. It had, on average, a similar energy consumption but a high inference time, despite having by far the least amount of parameters of the shown networks. In general, it seems that this specific architecture is not very well suited for solving the problem at hand.

For the SNN and RSNN, we expected results on the NVIDIA Jetson to be solely proportional to the parameters $time\_bin\_size$ and nb_input_copies, and whether the feedforward or recurrent architecture was used. These three factors mainly define the number of operations to be calculated during inference. Conversely, we assumed that the threshold does not affect performance as the implementation on general-purpose computers does not take advantage of the temporal sparsity in the data. Our measurements generally support these assumptions and are shown in Fig. 11. The average power usage of the feedforward and recurrent architectures across their thresholds were very close, deviating less than 0.7 % for the former, and less than 0.2 % for the latter from their respective means. This implies a constant utilization of computational resources as well as the energy consumption being directly proportional to the inference time for each architecture. Thresholds $\vartheta = 1$ and $\vartheta = 10$ as well as thresholds $\vartheta = 2$ and $\vartheta = 5$ consumed roughly the same amount of energy per inference. Tab. 3 shows that both threshold pairs presumably depend on $time\_bin\_size$ and nb_input_copies. The only exception is nb_input_copies for threshold $\vartheta = 2$ and $\vartheta = 5$, which is 8 and 4, respectively. This brings us to the conclusion that nb_input_copies does not have a meaningful impact on energy consumption and inference time in real-life scenarios, whereas $time\_bin\_size$ and the type of architecture are the main factors for the computational load. A comparison between the SNN and the eLSTMs shows an increased inference time of 20×, even though both compute the same number of time steps, indicating the lack of computational optimization for the SNN computation. The energy per sample is 10× higher and the average power consumption 1.5×.

In general, when comparing the absolute numbers of Fig. 11 with the standard classifiers in Fig. 10, our implementations of the FFSNN and RSNN have a clear disadvantage at energy consumption and inference time when being run on a GPU accelerated device. The most efficient

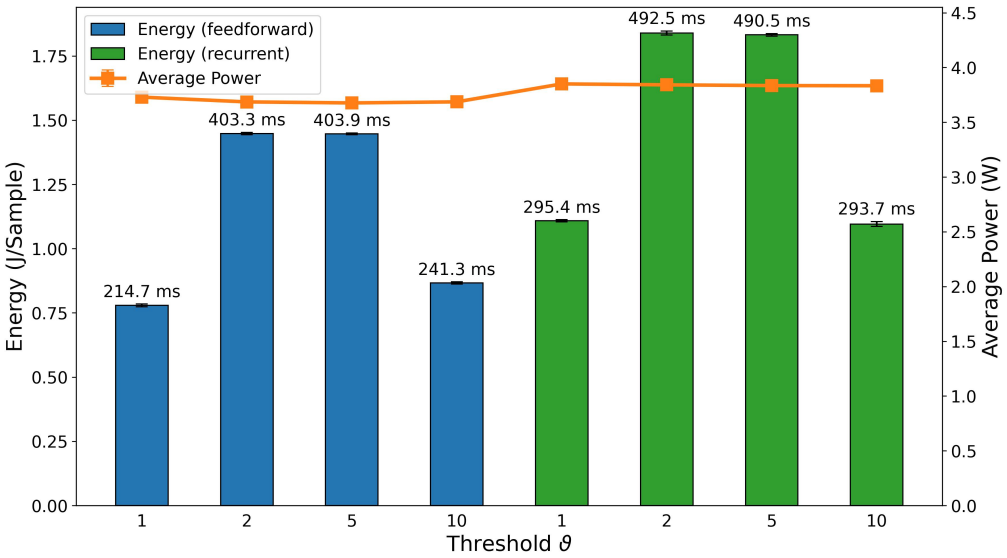


Fig. 11: Comparison of inference metrics for all spiking neural networks in terms of energy consumption and average power usage as measured on an NVIDIA Jetson Xavier NX. The label on top of each bar shows the inference time per sample on the corresponding network.

SNN consumed $\sim$88 % more energy than the least efficient standard classifier, and the fastest spiking network took 16.8$\times$ longer for one inference than the slowest standard classifier. These numbers clearly show the need for dedicated neuromorphic hardware to implement event-based algorithms.

3.5.2 Intel Loihi neuromorphic chip

The overall trend of accuracy in Loihi for the SNN, shown in Fig. 12 follows the trend of accuracy on the PyTorch simulations and Jetson inference shown in Fig. 9b. Nevertheless, there is a loss in accuracy of a few percent (e.g. -1.58 % for the RSNN with encoding threshold $\vartheta = 1$, which is due to the PyTorch training procedure without accounting for the Loihi hardware constraints, in particular, the 8-bit fixed point weights implementation. The loss varies depending on the PyTorch weights distribution.

Then, we compared the hardware efficiency of the recurrent and FFSNN in terms of delay (i.e. execution time), power and energy consumption. Before discussing the results, it is important to specify the neural cores mapping we have used which does not affect the accuracy but does affect the hardware efficiency. Loihi offers flexibility in how to map the network neurons into the neural

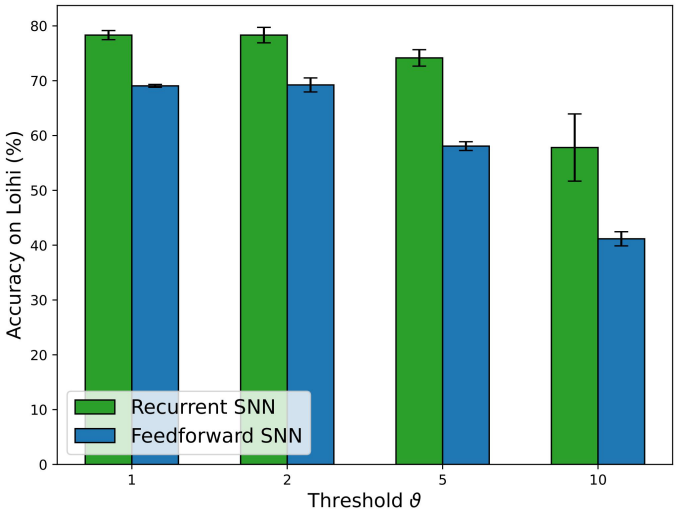


Fig. 12: Comparison of the accuracy results for FFSNN and RSNN on Loihi with the best parameters found by the two-stage HPO for each encoding threshold.

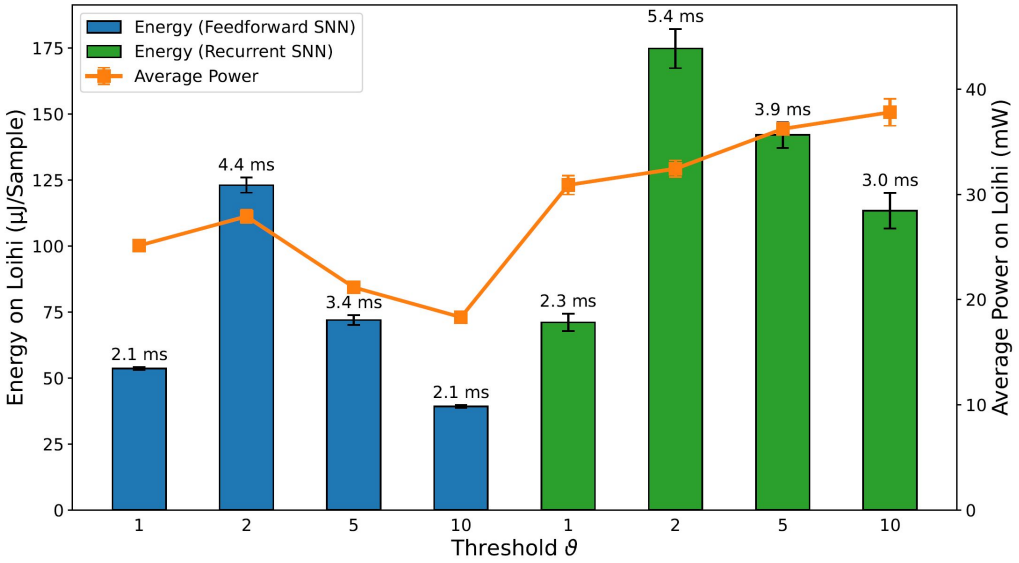


Fig. 13: Comparison of inference metrics for all trained spiking neural networks in terms of energy consumption and average power usage as measured on Loihi. The label on top of each bar shows the inference time per sample on the corresponding network.

cores, constrained by the number of cores in a chip as well as the number of input axons, synapses, neurons, and output axons in a neural core. The goal is then to find a good trade-off between parallelism (i.e. using more neural cores with fewer neurons per core) and time-multiplexing (i.e. using fewer neural cores with more neurons per core), to balance the neural core's power, the mesh routing power and algorithmic time step duration to get an optimal configuration for the application requirements. We found, that for the specific typologies, the more cores we use the more power and energy we consume without a significant impact on the delay. We, therefore, used the minimum number of cores considering all the hardware constraints which are 8 cores for all the trained networks.

The delay, power and energy consumption of the different deployed SNN on Loihi[3] are shown in Fig. 13. For the sake of simplicity, we refer to energy consumption as the main metric, as it includes both power and delay. As expected, FFSNN consumes less energy than RSNN, for the same thresholds. This is due to the overhead of memory and computation from the recurrent synaptic connections. FFSNN and RSNN follow a similar trend for the different thresholds: networks with thresholds $\vartheta = 2$ and $\vartheta = 5$ consume more energy because they have smaller bin sizes and therefore more time bins (i.e. algorithmic time steps) per sample (450) compared to networks with thresholds $\vartheta = 1$ and $\vartheta = 10$ (270), and they have more input copies as shown in Tab. 3. Networks with threshold $\vartheta = 2$ consume more than networks with threshold $\vartheta = 5$, mainly because they have more input copies (8 vs. 5). Nevertheless, the FFSNN with threshold $\vartheta = 1$ consumes more than the FFSNN with threshold $\vartheta = 10$, while the RSNN with threshold $\vartheta = 1$ consumes less than the RSNN with threshold $\vartheta = 10$. Even though in both cases the networks with threshold $\vartheta = 1$ have more events in the input and fewer events in the hidden layer than the networks with threshold $\vartheta = 10$, shown in the Supplementary Material Tab. S2, the impact of the hidden layer events is different, because every event in the FFSNN hidden layers gets transmitted to the 28 output neurons while every event in the RSNN hidden layers gets transmitted to both the 28 output neurons and the 450 hidden neurons. Therefore, the gain obtained in the input layer for the RSNN with threshold $\vartheta = 10$ is lost with the recurrent topology which increases the number of synaptic operations. Finally, while the Jetson GPU is

[3]Energy and timing measurements were obtained on Nahuku 32 board ncl-ext-ghrd-01 with an Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00 GHz and 4GB RAM running Ubuntu 20.04.4 LTS and NxSDK v1.0.0

Table 4: Results summary from RSNN on Loihi and RSNN, eLSTM and LSTM on Jetson for accuracy, total power, energy per sample, delay and energy-delay product. The number of trainable parameter (i.e. synaptic weights) are similar between the RSNN (236,700), the LSTM (225,975) and the eLSTM (236,919). Event-based inputs are encoded with threshold $\vartheta = 1$. Comparisons with respect to RSNN on Loihi are evaluated as differences for the accuracy and as ratios for all the other quantities.

	Results summary				**Comparison with RSNN on Loihi**		
Network	**RSNN**	**RSNN**	**eLSTM**	**LSTM**	**RSNN**	**eLSTM**	**LSTM**
Hardware	Loihi	Jetson	Jetson	Jetson	Jetson	Jetson	Jetson
Input	Events	Events	Events	Frames	Events	Events	Frames
Accuracy (%)	78.32	79.90	82.31	96.92	+1.58	+3.99	+18.60
Total power (mW)	31	3,851	7,642	5,385	124×	247×	174×
Total energy per sample (μJ)	71	1,108,695	96,000	35,212	15,615×	1,352×	496×
Delay per sample (ms)	2.3	295.3	12.9	6.7	172×	5.6×	2.9×
Energy-delay product (μJ·s)	0.16	327,398	1,238	236	2,046,237×	7,738×	1,475×

mostly sensitive to the number of bins as shown in Fig. 11, Loihi is also sensitive to the number of input copies and the spatio-temporal sparsity of the spikes and synaptic operations in the network.

After the comparison of the different deployed SNN on Loihi, taking into account the accuracy, power, energy, and delay, we conclude that the RSNN with encoding threshold $\vartheta = 1$ is the best option. For simplicity, we will refer to it as the RSNN for the rest of this section.

Tab. 4 compares RSNN in Loihi, the RSNN on Jetson, the LSTM and the eLSTM on Jetson. The RSNN on Loihi loses 1.58 % in accuracy compared to the RSNN on Jetson, mainly due to the quantization that was done after training. It further loses 17 % compared to the LSTM on Jetson, but only 3 % compared to the eLSTM. A specific LSTM architecture with the same number of parameters as in the RSNN has been used and trained for 300 epochs. However, the RSNN on Loihi show several orders of magnitude gains in hardware efficiency. First, compared to the RSNN on Jetson, it is 124× more power-efficient and 172× faster, which makes it four orders of magnitude (15,615×) more energy-efficient. It clearly shows that SNN are particularly inefficient when implemented on standard GPU hardware. It should nevertheless be highlighted that the delay or execution time of the RSNN on Jetson still satisfies the real-time constraint imposed by the sensor which has a sampling frequency of 40 Hz (i.e. a maximum algorithmic

time step duration of 25 ms). Even though the average execution time of the RSNN on Jetson is relatively long (295.38 ms), it is still lower than the total duration of each sample (1,350 ms). It is to note that this delay can increase in the real-world setting when adding off-chip communication with the robot. Second, compared to the LSTM on Jetson, the RSNN on Loihi is more than 170× more power-efficient and exhibits a 2.9× longer average execution time, which makes it three orders of magnitude (1,475×) more energy-efficient.

Finally, compared to the standard eLSTM classifier on the Jetson GPU using the same event data, the neuromorphic approach with the Loihi chip and RSNN is about 4 % less accurate but two orders of magnitude (247×) more power-efficient, reducing the total power from 7.642 W to about 31 mW. Furthermore, as a consequence of the reduced execution time, from 12.9 ms down to 2.3 ms, the neuromorphic approach introduces a gain in energy efficiency of 1,352× and a gain in energy-delay product of 7,738×. This goes in line with recent results of SNN on Loihi compared to standard algorithms and hardware, where the best performing workloads on Loihi make use of highly recurrent networks [88]. Furthermore, we can expect an even higher gain in energy efficiency when using the RSNN on Loihi in the real-world environment, because it exploits the spatio-temporal sparsity of the event-driven encoding. Therefore, if the robot is not moving its finger, no event is transmitted to the Loihi chip, drastically reducing the dynamic power that represents about 20 mW out of the total 31 mW. Instead, the Jetson GPU would always process the redundant frames coming from the sensor. Our study shows how event-driven encoding, neuromorphic hardware, and SNN are put together to improve the overall efficiency of tactile pattern recognition, emphasizing the importance of the neuromorphic approach for embedded applications with a continuous stream of data.

4 Conclusions

The initial analysis of the frame-based data using a linear classifier demonstrates that the information in our dataset is encoded in both the spatial and the temporal domains. The results show a decrease in the accuracy of linear classifiers when no time dimension is accounted for in both frame- and event-based data, thus motivating the use of architectures that are capable to learn spatio-temporal patterns from the data. Although linear classifiers provide a high accuracy

when all time bins are taken into account as predictors, it is not the desired approach to learning spatio-temporal patterns since they cannot be applied online where data needs to be gathered in real-time instead of being already available. This is in contrast with the sequential learning approaches and spike-based algorithms explored in this manuscript.

The data encoding analysis presents the trade-off between information content from the original frame-based data and the sparsity of the event stream. The original frame-based data is inherently redundant since the information content decreases slower than the compression ratio of events revealed by comparing the datasets at different threshold levels. Next to the impact of the encoding threshold, the analysis has shown that time binning needs to be considered as an important influencing factor. Due to the need to create vectors with sparse event representation to run on GPUs, the smallest $time\_bin\_size$ creates the lowest boundary in which the temporal dynamics of the event stream can be correctly represented. If ISI fall below the $time\_bin\_size$ information content and temporal dynamics suffer and information is lost. Based on the implementation results, the network may not reflect the sparsity in the input event stream at every network layer. An optimized network for task performance showed an increase in the number of events and quantity of energy consumption of the total architecture although the dataset presented to the input layer has a compression ratio bigger than 1 in terms of events. It is true, that the number of time bins has a significant impact on the power consumption, but comparing threshold $\vartheta = 1$ with $\vartheta = 10$ as well as $\vartheta = 2$ with $\vartheta = 5$ for the RSNN, which have the same $time\_bin\_size$ of 5 ms and 3 ms respectively, the higher threshold has in both cases a higher power consumption as highlighted in our hardware implementation on Loihi shown in Fig. 13, which can only be explained by a higher number of spikes transmitted in the network as reported in the Supplementary Material Tab. S2. Therefore, encoding schemes that minimize the number of events in the input may result in higher energy consumption in the system as a whole.

We further found that the spiking neuron current and voltage time constants that were optimized independently for each encoding threshold are similar because they are correlated to the inherent temporal dynamics of the input event stream rather than the encoding threshold or binning time window. The HPO did not settle at a global optimum for any of the encoding thresholds, which underlines the highly complex interaction of the included parameters, leading to many locally optimal solutions. Looking at the results of the follow-up grid search reveals, that all

$time\_bin\_size$ greater than 2 ms lead to a similar trend showing a slight decrease in classification accuracies for increasing $time\_bin\_size$. Relating the moderate decrease of accuracy with the increase in energy and power saving makes the selection of higher $time\_bin\_size$ for further robotic implementation preferable.

Our findings regarding the implementation of the SNN on the NVIDIA Jetson show that it is capable of fulfilling the constraints of real-time performance, which is defined here as the inference time of any network is lower than the recording time of one sample. The increase of the inference time between non-spiking and spiking architectures is substantial, though. It ranges from $\sim 16\times$ (Inception vs. FFSNN with threshold $\vartheta = 1$) to $\sim 91\times$ (FCN vs. RSNN with threshold $\vartheta = 2$). As a consequence, the energy consumption also rose by a significant margin which ranges from $\sim 2\times$ (Encoder vs. FFSNN with threshold $\vartheta = 1$) to $\sim 40\times$ (LSTM vs. RSNN with threshold $\vartheta = 2$). These numbers show the clear drawback of our implementation on conventional hardware. For a meaningful deployment, it would either need a more optimized implementation that is better accelerated by GPUs, or dedicated hardware that can take advantage of the characteristics of the spiking domain, like temporal sparsity.

We have been able to demonstrate the possibility of efficiently performing time series classification by exclusively using event-based encoding and asynchronous event-based computation on neuromorphic hardware. Our deployed RSNN can discriminate between 27 classes of Braille letters with 78.32 % accuracy, using only 450 recurrently connected hidden units and consuming a total amount of 31 mW on the Intel Loihi neuromorphic chip. This is yet not sufficient to report a competitive classification performance compared to standard classifiers or other results achieved with SNN on different tasks. The encoding analysis revealed that too much information is lost using sigma-delta encoding and that hinders the network from further performance improvements. Nevertheless, comparing these findings to a LSTM on the NVIDIA Jetson embedded GPU yields a gain in power-efficiency of $250\times$ for the RSNN with threshold $\vartheta = 2$ on Loihi. On the one hand, these results show the challenges of spike-based computing compared to standard algorithms in terms of accuracy; on the other hand, they highlight the opportunities of the neuromorphic approach with event-based communication and asynchronous processing in terms of power/energy efficiency and delay, especially for mobile robots or highly energy-constrained fields of applications. In addition to the neuromorphic computation while

performing the task, event-based encoding is important in this context. Event-based systems can be considered at rest while no significant change occurs and with that, the power consumption during that time is extremely low. Nevertheless, the system can react to changes immediately, due to its asynchronous nature.

The LSTM with the frame-based data-stream outperforms the RSNN in accuracy by 17 %, whereas the eLSTM achieves comparable performance to the the RSNN, highlighting the need for better event-based encoding techniques. For example, graded spikes supported by Loihi 2 [89] could reduce the information loss by adding the magnitude of change to the event (when it exceeds a threshold), keeping both event-based communication and precise information. Furthermore, models mimicking the biological skin with its wide range of neuronal responses like the slow and fast adaptive receptors [90] can be beneficial in terms of information extracted from the stimuli using multiple parallel pathways. Regarding the SNN, several mechanisms can be explored to improve the pattern recognition performance of the RSNN. First, by only applying a single recurrent hidden layer, we limited the trainable parameters in our network and with that, its learning capabilities. Future investigations might use multiple hidden layers to increase the spatio-temporal encoding power. Second, at the neuron level, we used homogeneous time constants for all neurons. Learning the time constants along with the synaptic weights to adapt the different neurons' temporal dynamics has been shown to be beneficial for the classification performance [91]. Third, recent works suggest the need for more powerful recurrent units for spiking neurons to bridge the gap with LSTM and other formal (i.e. non-spiking) recurrent neuron models [92, 93]. Finally, on another note, in contrast to this offline benchmarking methodology, an online classification could be performed by implementing a winner-take-all network [94] as the output layer or by using the recent attentive RSNN with a decision-making circuit [95], enabling the network's prediction readout at each time step.

Overall, we presented a tactile spatio-temporal dataset suitable for benchmarking event-based encoding schemes and neuromorphic algorithms. We compared an event-based encoding and spike-based learning algorithm with standard machine learning approaches and implemented a classification system into different hardware platforms showing the advantages of the end-to-end neuromorphic approach for tactile pattern recognition. Nevertheless, our findings are not limited to tactile data, because spatio-temporal information is present in all sensory modalities when

processed in a streaming fashion at the edge, and our approach can be applied to these types of workloads. Building full neuromorphic systems that inherently sense, process and communicate their output in an event-based manner provide great potential in terms of energy efficiency and scalability for embedded and *embodied* sensory-motor systems that interact in real-time with the real-world environment. Despite the energy benefits of neuromorphic hardware, neuromorphic algorithmic space is still being explored, therefore, not as mature as standard ANN methods. We envision that this problem and the performance metrics we used will serve as a benchmark to drive progress in the field.

Data Availability Statement

The code for encoding, training, inference, and exporting the HDF5 file from PyTorch is open source and available in a GitHub repository. The code for importing the HDF5 into Loihi is shared within the Intel Neuromorphic Research Community (INRC). The dataset is open access and uploaded in Zenodo under the DOI 10.5281/zenodo.7050094 [96].

5 Supplementary material

Energy consumption approximation from SNN simulation

While the energy consumption from the hardware implementations can be directly measured, in simulation it can be approximated by determining the number of spikes in each layer of the network. In Tab. 5 we show the total number of spikes normalized per letter in the dataset. The spikes have been calculated in simulation with the parameters and weights pre-trained according to the Tab. 3. Similar to the profile of energy consumption, as seen in the hardware implementation results, an increase in the threshold of the dataset transformation, that initially decreases the number of spikes in the input layer, does not correspond with a decrease in the number of spikes in the total network.

Table 5: Number of spikes in each of the layers of the network trained with parameters obtained from the optimization. Spikes calculated as the inference of the whole dataset. Percentages as ratio of spikes with the equivalent in threshold 1 as reference.

	Delta coding threshold			
Recurrent	**1**	**2**	**5**	**10**
IN layer	698	426 (61%)	126 (18%)	41 (6%)
MID Layer	1230	2165 (176%)	2775 (225%)	2591 (210%)
OUT LAYER	371	301 (81%)	508 (137%)	257 (69%)
Total	2298	2892 (126%)	3409 (148%)	2889 (126%)
Feedforward	**1**	**2**	**5**	**10**
IN layer	698	426 (61%)	126 (18%)	41 (6%)
MID layer	1029	4881 (474%)	4917 (478%)	2817 (274%)
OUT layer	496	507 (102%)	833 (168%)	235 (47%)
Total	2223	5815 (262%)	5876 (264%)	3092 (139%)

Support-Vector Machine

We applied a SVM with a linear kernel. The SVM was provided with raw sensor readings in the first place. In a pre-processing step, we first subtracted the sensor recordings from 255 to invert its range and make an increasing value represent increasing pressure on the sensor. We then normalised the resulting values by the maximum value out of all sensors and all trials. As last step, the sample-based values for all 12 sensors were concatenated and split in a training and test dataset, using 80% and 20% of the samples respectively. To validate the impact of the temporal nature, the SVM was evaluated with a continuously increasing number of samples, starting with

1 sample. Each iteration included 1 more sample until all samples were included. Each iteration was trained and tested 50 times.

Time-Series Classification

More details regarding the architectures used for time-series classification:

FCN: Mainly a CNN but without local pooling layers, which keeps the length of the time series unchanged. The last Fully-Connected (FC) layer is replaced by a Global Average Pooling (GAP) layer, helping to identify which of the time series contributed the most to the classification and reducing the number of parameters. The network contains 3 convolutional blocks with a stride of 1, each performing three operations: 1) convolution, 2) batch normalization, and 3) ReLU activation function. The result of the third convolutional block is averaged over the whole time dimension. The first convolution contains 128 filters with a filter length of 8, the second contains 256 filters with a filter length of 5, and the last contains 128 filters with a filter length of 3.

ResNet: The network is composed of 11 layers where the first 9 are convolutional layers, of which every 3 layers are forming a residual block. Short-cuts between these residual blocks are the main characteristic and difference to FCN. These blocks are followed by a GAP layer, averaging the time series across the time dimension, and a final softmax layer, containing neurons equal to the number of classes in the dataset. The convolutions in the residual blocks have a filter length fixed to 64 with batch normalization followed by ReLU as activation function. The filter length in the residual blocks is 8, 5, 3 respectively.

Encoder: A hybrid deep CNN inspired by FCN, whereas the GAP layer is replaced by an attention layer. The first three layers are convolutional layers. The first has 128 filters with a length of 5, the second 256 filters with a length of 11, and the third 512 filters with a length of 21. Each convolution is followed by an instance normalization operation and fed to a Parametric Rectified Linear Unit (PReLU). The output of the PReLU is followed by a dropout operation with a rate of 0.2 and a final max pooling of length 2. Finally, after the attention layer, a softmax classifier with the number of neurons equal to the classes in the dataset is used.

TCNN: The TCNN differs in 3 main points from the previously described networks. First, instead of using the cross-entropy loss function, MSE is used, leading to a FC final layer with sigmoid

as activation function. Second, local average pooling instead of local max pooling is used and one convolution is applied to each dimension of the Multivariante Time Series (MTS). Third, the final classifier is FC directly to the second convolution. Two convolutional layers with filter lengths 6 and 12 are followed by a local average pooling operation with a length of 3 and sigmoid as activation function. The convolutions are applied on all dimensions. The final FC layer has neurons equal to the number of classes in the dataset and sigmoid as activation function. Further, the output of the second convolution is directly fed as input to the FC layer.

Inception: The network is composed of two residual blocks, each containing three inception modules. Each inception module contains three main parts. The first part is a bottleneck layer, which performs convolution to represent the input according to the user defined size, the default value is 32, to reduce the input dimensionality. The second part are three parallel sliding filters with increasing length of 10, 20 and 40, which are applied to the output of the previous bottleneck layer. The third part is a max pooling layer, performed on the input of the inception module, which is followed by a bottleneck layer. The output of the three filters together with the this bottleneck layer are concatenated and form the output of a module. The two residual blocks are followed by a GAP layer and a final FC layer with neurons equal to the number of classes in the dataset.

Parameter counting (LSTM)

The calculation of the number of parameters in the RSNN was made according to

$$nb\_parameters = nb\_channel \times nb\_input\_copies \times nb\_hidden + nb\_hidden^2 + nb\_hidden \times nb\_output$$

with $nb\_channel = 12$, $nb\_input\_copies = 2$, $nb\_hidden = 450$, and $nb\_output = 27$.

The LSTM recurrent neural network is comprised of four basic gates that take as input the concatenation of the input feature vector $x_{(t-n)}$ and the previous hidden state, and can be conceptually summarized as

1. *Input gate*: it decides which new values are going to be incorporated to the cell.
2. *Forget gate*: it takes the input x_t, and the previous hidden state h_{t-1} and delivers a number

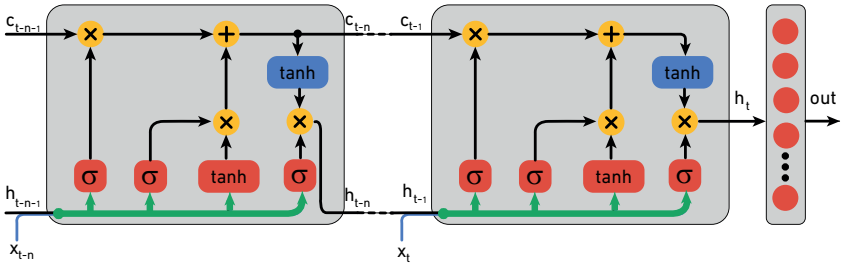


Fig. 14: Implemented single layer LSTM with a fully connected of size $h \times 27$. The green line represents the concatenation operation, the red blocks represent weighted operations followed by the activation indicated in each label, the yellow are element-wise operations, and the blue are $tanh$ activation functions.

between 0 and 1 for each cell state C_{t-1} indicating how much past information should be remembered.

3. *Estimated cell state*: New candidate vector created also from the input x_t and previous hidden state h_{t-1}, that is combined with the information coming from the input gate, to produce the new cell state C_t.

4. *Output gate*: it filters the new cell state and combines it with a filtered version of the input and last hidden state h_{t-1} to produce the new hidden state h_t.

Each gate has the same number of trainable parameters in the form of weight W and bias b, and can be calculated by hand as

$$\text{parameters per gate} = \underbrace{h_{size} \times (h_{size} + x_{size})}_{\text{number of } W_i} + \underbrace{h_{size}}_{\text{number of } b_i}$$

where $(h_{size} + x_{size})$ is the resulting dimension of the concatenation between the input feature x_t and previous hidden state h_{t-1}. The input feature x_t results from replicating i_{copies} times the values of the 12 taxels in each time step, resulting in a total parameter count of

$$\text{total number of parameters} = 4\,[h_{size} \times (12 \times nb\_input\_copies + h_{size}) + h_{size}] + nb\_output \times h_{size}$$

In this case, we aim to have the same number of parameters as the best performing RSNN for

Table 6: LSTM Parameter counting. threshold 1

Threshold $\vartheta = 1$

scale	time bin size	nb input copies	tau mem (ms)	tau ratio	fwd weight scale	weight scale factor	reg spikes	reg neurons	accuracy (%)
5	8	4	0.06	10	1	0.01	0.004	1E-06	79.61
5	5	1	0.08	10	1	0.04	0.0025	4E-06	78.20
5	5	4	0.08	10	1	0.04	0.003	4E-06	78.09
40	7	8	0.08	10	1	0.035	0.003	0.0	77.70
5	5	4	0.08	10	1	0.04	0.0025	4E-06	77.69
5	5	4	0.08	10	1	0.04	0.0025	3E-06	77.42
5	5	4	0.08	10	1	0.045	0.0025	3E-06	77.30
35	7	8	0.09	10	1	0.035	0.0025	0	77.22
5	8	2	0.1	10	1	0.045	0.003	6E-06	77.16
40	7	2	0.08	10	1	0.045	0.003	0	77.11

Table 7: LSTM Parameter counting. threshold 2

Threshold $\vartheta = 2$

scale	time bin size	nb input copies	tau mem (ms)	tau ratio	fwd weight scale	weight scale factor	reg spikes	reg neurons	accuracy (%)
15	8	8	0.05	10	1	0.02	0.0015	0	81.37
15	8	4	0.06	10	1	0.025	0.001	0	81.05
25	5	1	0.08	10	1.5	0.025	0	0	81.03
10	9	8	0.05	10	1	0.02	0.0015	0	80.59
5	7	2	0.05	10	2	0.025	0.0015	1E-06	79.92
5	10	2	0.04	2	2.5	0.03	0.001	0	79.64
30	5	4	0.08	10	1.5	0.02	0.001	0	79.59
30	5	4	0.08	10	1.5	0.02	0.0005	0	79.40
15	9	8	0.05	10	1.5	0.02	0.002	0	79.35
15	3	1	0.05	10	1	0.015	0.003	1E-06	79.34

comparison purposes, leading to 236,892 parameters in total with 228 hidden units.

Table 8: LSTM Parameter counting. threshold 5

Threshold $\vartheta = 5$

scale	**time bin size**	**nb input copies**	**tau mem** (ms)	**tau ratio**	**fwd weight scale**	**weight scale factor**	**reg spikes**	**reg neurons**	**accuracy** (%)
10	3	1	0.07	10	1.5	0.035	0.001	0	85.62
10	4	2	0.07	10	1	0.04	0.002	0	85.40
10	3	2	0.07	10	1.5	0.035	0.001	0	84.64
10	3	4	0.07	10	1.5	0.035	0.001	0	84.61
10	4	2	0.06	5	1.5	0.04	0.0015	0	84.49
10	3	2	0.07	10	1.5	0.035	0.0015	0	84.46
10	4	2	0.08	10	1.5	0.04	0.0025	0	84.29
10	4	2	0.07	10	1.5	0.035	0.0005	0	84.21
10	3	1	0.07	10	1.5	0.035	0.0015	0	84.02
10	4	2	0.08	10	1.5	0.04	0.002	0	83.94

Table 9: LSTM Parameter counting. threshold 10

Threshold $\vartheta = 10$

scale	**time bin size**	**nb input copies**	**tau mem** (ms)	**tau ratio**	**fwd weight scale**	**weight scale factor**	**reg spikes**	**reg neurons**	**accuracy** (%)
10	6	1	0.07	10	4	0.015	0.0015	0	75.30
35	7	8	0.06	2	2	0.03	0.001	0	73.60
10	5	2	0.06	10	3.5	0.045	0.0025	0	73.23
10	5	4	0.05	2	3.5	0.045	0.0035	0	72.83
10	5	4	0.06	10	3.5	0.045	0.0025	0	72.56
10	5	1	0.06	10	3.5	0.05	0.0025	0	72.43
10	5	1	0.05	10	3.5	0.05	0.003	0	72.43
10	5	1	0.06	10	3.5	0.045	0.002	0	72.33
10	8	2	0.07	2	1	0.015	0.0025	1E-06	72.32
10	5	1	0.05	2	3.5	0.045	0.0035	0	72.23

References

[50] R. Romo and E. Salinas, "Touch and go: decision-making mechanisms in somatosensation," *Annual review of neuroscience*, vol. 24, no. 1, pp. 107–137, 2001.

[51] T. J. Prescott, M. E. Diamond, and A. M. Wing, "Active touch sensing," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 366, no. 1581, pp. 2989–2995, 2011. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2011.0167

[52] P. Bach-y-Rita, "Tactile sensory substitution studies," *Annals of the New York Academy of Sciences*, vol. 1013, 2004.

[53] N. Martiniello and W. Wittich, "The association between tactile, motor and cognitive capacities and braille reading performance: a scoping review of primary evidence to advance research on braille and aging," *Disability and Rehabilitation*, pp. 1–15, 2020.

[54] L. Bola, K. Siuda-Krzywicka, M. Paplińska, E. Sumera, P. Hańczur, and M. Szwed, "Braille in the sighted: Teaching tactile reading to sighted adults," *PloS one*, vol. 11, no. 5, p. e0155394, 2016.

[55] M. Brysbaert, "How many words do we read per minute? a review and meta-analysis of reading rate," *Journal of Memory and Language*, vol. 109, p. 104047, 2019.

[56] H. Kawabe, S. Seto, H. Nambo, and Y. Shimomura, "Experimental study on scanning of degraded braille books for recognition of dots by machine learning," in *International Conference on Management Science and Engineering Management*. Springer, 2019, pp. 322–334.

[57] J. Li and X. Yan, "Optical braille character recognition with support-vector machine classifier," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 12. IEEE, 2010, pp. V12–219.

[58] B.-M. Hsu, "Braille recognition for reducing asymmetric communication between the blind and non-blind," *Symmetry*, vol. 12, no. 7, p. 1069, 2020.

[59] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi, and S. J. Kwon, "Deep learning scheme for character prediction with position-free touch screen-based braille input method," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, pp. 1–24, 2020.

[60] T. Li, X. Zeng, and S. Xu, "A deep learning method for braille recognition," in *2014 International Conference on Computational Intelligence and Communication Networks*. IEEE, 2014, pp. 1092–1095.

[61] C. Bartolozzi, L. Natale, F. Nori, and G. Metta, "Robots with a sense of touch," *Nature materials*, vol. 15, no. 9, pp. 921–925, 2016.

[62] C. Bartolozzi, P. M. Ros, F. Diotalevi, N. Jamali, L. Natale, M. Crepaldi, and D. Demarchi, "Event-driven encoding of off-the-shelf tactile sensors for compression and latency optimisation for robotic skin," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 166–173.

[63] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[64] J. Conradt, R. Berner, M. Cook, and T. Delbruck, "An embedded aer dynamic vision sensor for low-latency pole balancing," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2009, pp. 780–785.

[65] V. Chan, S.-C. Liu, and A. van Schaik, "Aer ear: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.

[66] R. G. Leonard and G. Doddington, "Tidigits speech corpus," *Texas Instruments, Inc*, 1993.

[67] K. E. Friedl, A. R. Voelker, A. Peer, and C. Eliasmith, "Human-inspired neurorobotic system for classifying surface textures by touch," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 516–523, 2016.

[68] U. B. Rongala, A. Mazzoni, and C. M. Oddo, "Neuromorphic artificial touch for categorization of naturalistic textures," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 4, pp. 819–829, 2015.

[69] H. See, B. Lim, S. Li, H. Yao, W. Cheng, H. Soh, and B. C. K. Tee, "ST-MNIST - the spiking tactile MNIST neuromorphic dataset," *CoRR*, vol. abs/2005.04319, 2020. [Online]. Available: https://arxiv.org/abs/2005.04319

[70] L. Bologna, J. Pinoteau, J. Passot, J. Garrido, J. Vogel, E. R. Vidal, and A. Arleo, "A closed-loop neurobotic system for fine touch sensing," *Journal of neural engineering*, vol. 10, no. 4, p. 046019, 2013.

[71] J. Pinoteau, L. L. Bologna, J. A. Garrido, and A. Arleo, "A closed-loop neurorobotic system for investigating braille-reading finger kinematics," in *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*. Springer, 2012, pp. 407–418.

[72] N. Jamali, M. Maggiali, F. Giovanniniand, G. Metta, and L. Natale, "A new design of a fingertip for the icub hand," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[73] J. M. de la Rosa, "Sigma-delta modulators: Tutorial overview, design guide, and state-of-the-art survey," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 1, pp. 1–21, 2011.

[74] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1578–1585.

[75] Y. Geng and X. Luo, "Cost-sensitive convolution based neural networks for imbalanced time-series classification," *CoRR*, vol. abs/1801.04396, 2018. [Online]. Available: http://arxiv.org/abs/1801.04396

[76] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series," *CoRR*, vol. abs/1805.03908, 2018. [Online]. Available: http://arxiv.org/abs/1805.03908

[77] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[78] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 1, pp. 162–169, 11 2020. [Online]. Available: https://doi.org/10.1007/s10618-020-00710-y

[79] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, p. 917–963, Mar 2019. [Online]. Available: http://dx.doi.org/10.1007/s10618-019-00619-1

[80] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[81] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2020.

[82] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Computation*, vol. 33, no. 4, pp. 899–925, Mar 2021. [Online]. Available: https://doi.org/10.1162/neco_a_01367

[83] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, 2019.

[84] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS Autodiff Workshop*. Conference on Neural Information Processing System, 2017, pp. 1–4.

[85] V. Fra, E. Forno, R. Pignari, T. C. Stewart, E. Macii, and G. Urgese, "Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications," *Neuromorphic Computing and Engineering*, vol. 2, no. 1, p. 014006, Feb. 2022. [Online]. Available: https://doi.org/10.1088/2634-4386/ac4c38

[86] E. Forno, A. Acquaviva, Y. Kobayashi, E. Macii, and G. Urgese, "A Parallel Hardware Architecture For Quantum Annealing Algorithm Acceleration," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, vol. 2018-Octob. IEEE, 2018, pp. 31–36.

[87] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[88] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[89] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, "Efficient neuromorphic signal processing with loihi 2," 2021. [Online]. Available: https://arxiv.org/abs/2111.03746

[90] V. E. Abraira and D. D. Ginty, "The sensory neurons of touch," *Neuron*, vol. 79, 2013.

[91] N. Perez-Nieves, V. Leung, P. Dragotti, and D. Goodman, "Neural heterogeneity promotes robust learning," *Nature Communications*, vol. 12, p. 5791, 10 2021.

[92] W. He, Y. Wu, L. Deng, G. Li, H. Wang, Y. Tian, W. Ding, W. Wang, and Y. Xie, "Comparing snns and rnns on neuromorphic vision datasets: Similarities and differences," *Neural networks : the official journal of the International Neural Network Society*, vol. 132, pp. 108–120, 2020.

[93] F. Paredes-Vallés, J. J. Hagenaars, and G. C. de Croon, "Self-supervised learning of event-based optical flow with spiking neural networks," in *NeurIPS*, 2021.

[94] Y. Chen, "Mechanisms of winner-take-all and group selection in neuronal spiking networks," *Frontiers in Computational Neuroscience*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2017.00020

[95] B. Yin, Q. Guo, F. Corradi, and S. Bohte, "Attentive decision-making and dynamic resetting of continual running srnns for end-to-end streaming keyword spotting," in *Proceedings of the International Conference on Neuromorphic Systems 2022*, ser. ICONS '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3546790.3546795

[96] S. F. Müller-Cleve, "Tactile braille letters dataset," Distributed by Zenodo https://doi.org/10.5281/zenodo.6556273, May 2022.

Towards efficient keyword spotting using spike-based time difference encoders

Pequeno-Zurro A., Khacef L., Panzeri S., Chicca E. “Towards efficient keyword spotting using spike-based time difference encoders” [In Preparation].

Keywords: *neuromorphic computing, spiking neural networks, time difference encoder, spatio-temporal pattern recognition, energy-efficiency, keyword spotting.*

Candidate's contribution to the paper

Conceptualization: **Alejandro Pequeno-Zurro**, Lyes Khacef, Stefano Panzeri, Elisabetta Chicca

Methodology: **Alejandro Pequeno-Zurro**, Lyes Khacef, Stefano Panzeri, Elisabetta Chicca

Software: **Alejandro Pequeno-Zurro**

Validation: **Alejandro Pequeno-Zurro**

Formal analysis: **Alejandro Pequeno-Zurro**, Lyes Khacef, Stefano Panzeri, Elisabetta Chicca

Investigation: **Alejandro Pequeno-Zurro**

Data Curation: **Alejandro Pequeno-Zurro**, Lyes Khacef

Writing (original draft): **Alejandro Pequeno-Zurro**, Lyes Khacef

Writing (review and editing): **Alejandro Pequeno-Zurro**, Lyes Khacef, Stefano Panzeri

Visualization: **Alejandro Pequeno-Zurro**

Supervision: Lyes Khacef, Stefano Panzeri, Elisabetta Chicca

Abstract

Keyword spotting in edge devices is becoming increasingly crucial as voice-activated assistants are widely used. However, its deployment is often limited by the extremely low-power constraints of the target embedded systems. Here we explore the performance in keyword spotting of the Time Difference Encoder (TDE), a recent model of Current-Based Leaky Integrate-and-Fire (CuBa-LIF) neurons with rich temporal dynamics, in order to perform efficient keyword spotting with neuromorphic processors. The TDE has two inputs and three leaky compartments, and its output encodes the time difference between its two inputs' spikes in a given order. We use the TIdigits dataset of spoken digits with a formant decomposition and rate-based encoding into spikes and compare three architectures of spiking neural networks (SNNs) to learn and classify the spatio-temporal signals. The SNNs are made of three layers with the same input and output layers and a hidden layer composed of either (1) feedforward TDEs, (2) feedforward CuBa-LIFs or (3) recurrent CuBa-LIFs. We first show that the spike patterns of the spoken digits have a large amount of information in the temporal domain, reinforcing the importance of better exploiting time for such a task. We then train the three SNNs with the same number of synaptic weights to quantify and compare their performance based on the classification accuracy and the number of synaptic operations. Our results show that the accuracy of the feedforward TDE network (89%) is higher than the feedforward CuBa-LIF network (71%) and close to the recurrent CuBa-LIF network (91%). However, although they have the same number of synapses, the feedforward TDE-based network performs 92% less synaptic operations than the recurrent CuBa-LIF network. In addition, the training of the TDE network is highly interpretable in terms of the frequency and timescale features of the spoken keywords in the dataset. Our findings support the claim that the TDE is a promising spiking neuron model to implement in neuromorphic hardware for scalable event-driven processing of spatio-temporal patterns.

Keywords: neuromorphic computing, spiking neural networks, time difference encoder, spatio-temporal pattern recognition, energy-efficiency, keyword spotting.

1 Introduction

Keyword spotting is a representative task of the always-on, real-time computation at the device's edge. Edge devices such as smart home solutions, smartphones and other wearables are becoming ubiquitous in our daily lives. This trend pushes intelligence into tiny embedded systems that are highly constrained in energy consumption and need to interact in real-time with their-environment [97, 98]. This combination of low-power and low-latency requirements limits the deployment of keyword spotting systems on conventional hardware, such as CPU and GPU architectures. More generally, AI progress with current algorithms and hardware limitations is predicted to become technically unsustainable [99, 100], in particular, when targetting constrained embedded systems [101]. Neuromorphic hardware and systems present an alternative to traditional silicon circuits by harnessing the efficiency and computational power of the brain.

Spiking neural network(SNN) models implemented in neuromorphic hardware present an efficient alternative to building AI systems [102–104]. SNNs are considered the third generation of neural networks [105] characterized by two main properties: (1) asynchronous computing, which exploits the sparsity of event-driven sensors, and (2) stateful compartments which may have inherent temporal dynamics in the form of leakage. Hence, compared with conventional Artificial Neural Networks (ANNs), SNNs exploit more efficiently the temporal information of the data [106]. Applied to cortical signals, especially auditory stimuli, this property of SNN models is particularly promising given the evident importance of spike timing in neural codes for encoding and behavioural discrimination [107–109]. SNNs have gotten traction in the last five years with the emergence of neuromorphic hardware platforms such as IBM TrueNorth [110] and Intel Loihi [111], effectively demonstrating the low-latency and low-power potential in several applications such as multimodal visual-EMG hand gesture recognition [112], tactile sensing [113] and keyword spotting [114].

In the context of audio signal processing, formant frequencies have been historically important for determining the phonetic content of speech sounds [115]. Authors typically have used Hidden Markov Model (HMM) with a preprocessing of the Mel Frequency Cepstrum Coefficients (MFCCs) for speech recognition systems [116–118]. However, several authors have proved that using formants as initial preprocessing of the audio signal improves the performance

of speech recognition algorithms [119–122]. We propose using novel SNN models as a more efficient and scalable system implementation for classifying the spatio-temporal patterns found in spoken words.

Regarding sensory neurons in the brain, several works highlight the importance of spike timing in the neural code and the use of time-related computations for sensory discrimination tasks. Neural populations encode information in the temporal codes within the time pattern, latency, interspike intervals, or phase of firing [123]. While there is still unknown how the cortical circuits exploit this information at the computational level, these codes require a reference signal to decode the time difference whether it is aligned to a mechanical onset (latency code to stimulus onset), a different spike train from a population of neurons (stereotyped neurons), or network oscillations [124]. This type of encoding mechanism has been found across the main sensory modalities (visual, auditory, somatosensory).

The proposed TDE model [125, 126] is a particular Current-Based Leaky Integrate-and-Fire (CUBA-LIF) neuron with an extra synaptic compartment and fixed input directional connectivity. The spiking rate of the cell's output is proportional to the time difference of the input's spike trains. The output is therefore maximised with highly correlated spike trains in its input. The neuron dynamics and the cell's spatial selectivity make it especially suitable to exploit the spatio-temporal structure of the data. Therefore, it makes it suitable for processing spatio-temporal features from all sensory modalities such as vision [125–127], audition [121, 128], touch [129], and olfaction. Related cortical circuits that inspire this type of implementation can be found in animals that range from insects to mammals, e.g. visual optic flow circuits [130–132] and the auditory thalamocortical processing [133, 134].

The contributions of this work are outlined here. We apply the TDE neuron previously introduced in the visual sensory domain into speech recognition. The initial analysis of the dataset shows a non-negligible amount of information in the time pattern of the encoded spikes. We train our SNNs using supervised learning methods and compare the performance of our three architectures with TDE neurons against CUBA-LIF neuron models. Next, we present our results where we show a similar performance between TDE neuron models and recurrent network of CUBA-LIF cells whose number of neurons is balanced based on the synaptic weights. While the performance

is similar in pattern recognition, there are several advantages found when using the TDE network architecture in comparison with the recurrent CUBA-LIF. We present the advantages found in terms of efficiency with 92% reduction of synaptic operations, scalability with highly sparse signal representations and high interpretability of the results, and investigate these claims in the context of classifying spatio-temporal patterns in neuromorphic systems.

2 Materials and methods

The pipeline of the proposed SNN architecture is represented in Figure 15. Spoken digits are preprocessed through a formant decomposition method to extract the evolution in time of the frequencies with the highest energy of the signal, which corresponds with the resonance frequencies of the vocal tract. The frequency space is represented in the spatial dimension of the network's input, and the amplitude of the formants represents the evolution in time of the energy per frequency. The spatio-temporal features are then encoded and processed in a three-layer SNN for spoken keyword classification. The layers consist of a first encoding layer (L0) that encodes the amplitude of the formants into spike trains, a hidden layer (L1), and an output layer (L2) in which every neuron output represents a keyword class. The keyword class is decoded from the most active neuron in layer L2. We use a spoken digits dataset, apply speech decomposition tools to extract the formants and compare network performance with different neuron models in the hidden layer L1. All the networks are trained to maximize the accuracy in the classification task according to supervised learning methods applied to SNNs. In the following sections, we dissect the dataset, network architecture comparison, and training methods for classifying spoken digits with SNNs.

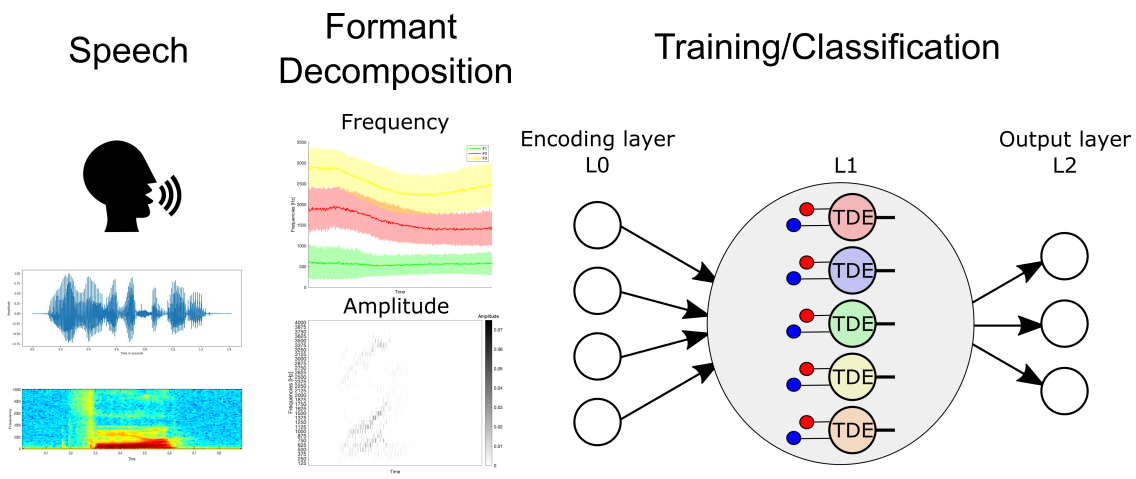


Fig. 15: System pipeline. From speech generation, we extract offline formant decomposition in frequency, and the amplitude of the three formants with the biggest energy. Frequency is then represented as spatial information in the number of input neurons and amplitude of the formants is encoded in L0 into spikes. The network's parameters are optimised for performance in the classification task.

2.1 Dataset and spike-based encoding

To test the performance of the SSN in keyword spotting, we use a popular speech recognition dataset, TIdigits [135], that consists of speech utterances from 225 adult subjects from 11 classes (10 digits plus 'oh' the alternative utterance for 0). The dataset has been sampled at 20 Khz for a total of 450 repetitions per class. The spatio-temporal sequences of words last approximately 400ms and are presented at random initialisation times. We built the dataset with a fixed duration of 1500ms keeping this random vocalisation in place. Each of the keywords has been preprocessed into the main formants of the words as keyword features for the classification. The formants are defined in speech science as the acoustic resonances of the vocal tract and correspond with the main frequency components of the signal. Previous works used these features for word discrimination and speech recognition, where it is estimated that three formants are sufficient to discriminate vocal and semi-vocal sounds [115, 122]. In this work, we use the sinewave speech analysis toolbox in Matlab [136] to extract the main three formants for each keyword. The formant frequencies have associated amplitudes that determine the energy of each formant. The sequence in time of the formant frequency and amplitude are the spatio-temporal patterns that characterise each keyword class. For our application, we quantify the frequency spectrum into 32 equally spaced frequency bands mapping the typical frequency range in human speech(0-4 Khz) into channels with a width of 125Hz. Each of these channels codifies the

maximum amplitude of the three main formants into a temporal sequence of frequency amplitudes. Since the frequency channels have a bandwidth, there is the possibility of a coincidence of 2 formants within the same input channel.

The first layer of the classification network L0, composed of CUBA-LIF neurons, converts the input sequence of amplitudes into spikes through the current of the neurons. Each of the 32 channels is connected to a single neuron for the spike conversion. Therefore, every cell in L0 represents a frequency band in the formant space of the keywords, creating a spatio-temporal spike code in which space represents the frequency domain and time the evolution of the formants for each of the keywords. Figure 16 illustrates this conversion for 2 sample keywords of the words "oh" and "three". The left side of the figure represents the evolution in time of the frequency channels and their respective associated amplitude. Once the sequence of frequency amplitudes passes through the encoding layer of the network L0, the formants are transformed into spikes, as seen on the right side of the figure. This conversion is homogeneous for all frequencies with the same parameters for the neurons representing the frequency bands. This encoding layer, as well as the decoding layer L2, consists of the same type, number and cell parameters for all the networks compared in this work. The use of formant frequencies reduces the amount of information in the frequency domain that receives the encoding layer to the most relevant features in speech recognition aiming for a more sparse representation in the encoding layer of our network.

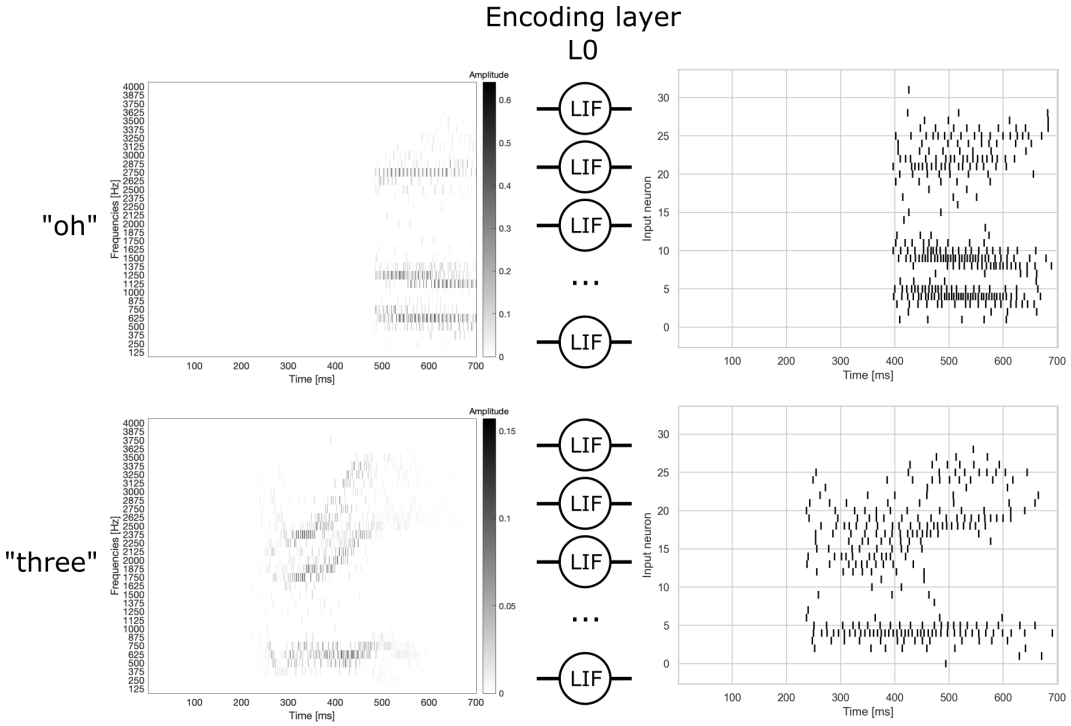


Fig. 16: Example of formants transformed into spikes for 2 sample keywords. Spoken keywords are decomposed in formant frequency bands and transformed into spikes in the encoding layer of the networks. Left side, sequence in time of frequency amplitude. Right side, the transformation of amplitude into spikes through the current of the CUBA-LIF neuron.

2.2 Networks architectures

We built different SNNs based on the TDE neuron model and the CUBA-LIF neuron model, as seen in Figure17. While network layers L0 and L2 are formed with CUBA-LIF neurons for all the architectures, our work shows how different models in the hidden layer L1 affect the performance, efficiency and interpretability of the classification. We can conceptualize layer L0 as an encoding layer since the formant's amplitude from the speech preprocessing is converted into a train of spikes assigning each of the neurons in L0 a frequency band. On the other end of the network, every neuron in layer L2 represents one keyword class in the classification problem. The class is selected based on the neuron with higher activity at its output. For all the networks in this comparative study, layer L1 is fully connected to layer L2, and the weight parameter (W_2) represents the matrix of connectivity between cells in L1 and output classes in L2. This parameter is also trained for all the networks in this work. The network types benchmarked are defined in the text as the TDE network whose L1 layers are made of TDE neurons, the Leaky

Integrat-and-Fire with recurrency (LIFrec) network with CUBA-LIF recurrent cells in the L1 layer, and the LIF network that is composed of CUBA-LIF cells in the hidden layer L1.

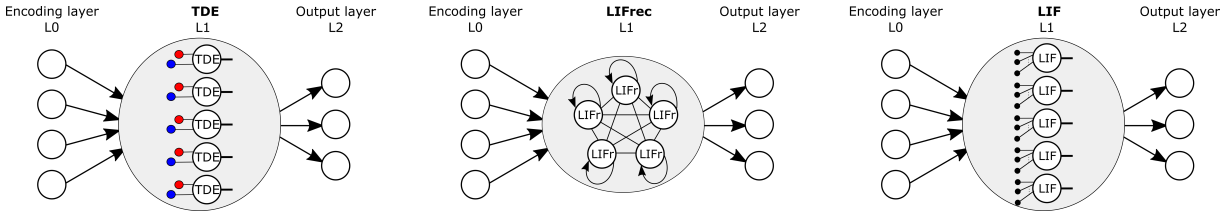


Fig. 17: Network architectures are compared in the manuscript. Networks present the same number of neurons and the same type of model in layers L0 and L2. They also have the same number of synapses and a similar number of trained parameters for comparison.

2.2.1 Current-Based Leaky Integrate-and-Fire (CuBa-LIF)

The CUBA-LIF model is adopted from previous work ([113, 137, 138]). This model, core of the comparative study, is described based on its membrane potential of the neuron i in the layer l with $U_i^{(l)}$ as,

$$\tau_{mem}\frac{\mathrm{d}U_i^{(l)}}{\mathrm{d}t} = -(U_i^{(l)} - U_{rest}) + RI_i^{(l)} \tag{17}$$

with U_{rest} as membrane potential, τ_{mem} as the membrane time constant that defines the time dynamics of the cell, R as resistance, and $I_i^{(l)}$ the input current of the cell. Similarly, the $I_i^{(l)}$ is described as the combination of inputs from the previous layer $l-1$ accounting for the weights as:

$$\tau_{syn}\frac{\mathrm{d}I_i}{\mathrm{d}t} = -I_i(t) + \sum_j W_{ij}S_j^{(l-1)}(t) + \sum_j V_{ij}S_j^{(l)}(t) \tag{18}$$

with τ_{syn} the synaptic time decay of the input, $\sum_j W_{ij}S_j^{(l-1)}(t)$ describes the combination of the input contributions S_j of the jth presynaptic neuron combined with the correspondent weight W_{ij}. Moreover, the contribution $\sum_j V_{ij}S_j^{(l)}(t)$ only applies when we use recurrency in our neuron model as what we described in the text as recurrent LIF architecture or the abbreviation LIFrec. Discretizing the previous equations for the modelling into:

$$I_i^{(l)}(t) = \alpha I_i^{(l)}(t-1) + \sum_j W_{ij}\cdot S_j^{l-1}(t) \tag{19}$$

$$U_i^{(l)}(t) = (\beta U_i^{(l)}(t-1) + I_i^{(l)}(t)) \cdot (1.0 - U_{reset}) \tag{20}$$

with $\beta = \exp(\frac{-dt}{\tau_{mem}})$ as the voltage decay constant of the membrane, $\alpha = \exp(\frac{-dt}{\tau_{syn}})$ the current decay constant at synaptic state. We use for convenience an input resistance $R = 1\Omega$ and U_{reset} the voltage to reset the membrane after evoking a spike event.

2.2.2 Time Difference Encoder (TDE)

The TDE is a processing cell model bio-inspired from the optical flow neural circuitry in biological organisms and used previously as a spiking Elementary Motion Detector (sEMD) [125]. This model has been applied in the spiking domain for visual motion detection and obstacle avoidance [126, 127].

The TDE model consists of a pre-synaptic compartment with two inputs connected to a LIF neuron, as seen in Figure 18. The spike rate in the output of the neuron is inversely proportional to the time difference between the two inputs of the cell in a directional matter. A spike in the facilitatory input (red trace) elicits a trace in the Gain compartment characterised by the time constant τ_g. When a spike arrives at the trigger input (blue trace), the Excitatory PostSynaptic Current (EPSC) of the cell is increased by the value of the gain at that time. The resulting EPSC presents the same dynamics of the CUBA-LIF neuron model in which a cell membrane integrates the current of the EPSC trace to generate spikes in its output. The resulting equations are as follows:

$$\tau_{syn}\frac{\mathrm{d}I_i}{\mathrm{d}t} = -I_i(t) + \sum_j G_i(t) S_{j,ithrig}(t) \tag{21}$$

$$\tau_{gain}\frac{\mathrm{d}G_i}{\mathrm{d}t} = -G_i(t) + \sum_j W_{j,i} S_{j,ifac}(t) \tag{22}$$

where compared with the CUBA-LIF model, the current $I_i(t)$ has a time dependency in the weight $G_i(t)$ that multiplies the thrigger connection. This synaptic compartment, defined as $gain$ computes the spikes in the facilitatory connection with a leak governed by the time constant τ_g. Similar to the previous discretization, the resulting equations are as follows:

$$I_i(t) = \alpha I_i(t-1) + \sum_j G_i(t) \cdot S_{j,thrig}(t) \tag{23}$$

$$G_i(t) = \gamma G_i(t-1) + \sum_j W_{i,j} \cdot S_{j,fac}(t) \tag{24}$$

where γ represents the exponential decay $\gamma = \exp(\frac{-dt}{\tau_g})$ with τ_g as an extra parameter of the cell which modulates the time effect of the correlation between the two inputs. $W_{i,j}$ represents an additional weight assigned to the facilitatory spike that we will keep at 1 in all cells for simplification. Therefore, the TDE aggregates a pre-synaptic compartment in which the gain of the trigger connection is modulated by the temporal trace of the facilitatory input, as represented in the figure. τ_g governs the leakage of the gain compartment assigned to the facilitatory input. It is a key parameter of the cell since it controls the effect of the gain and the effective time distance between the spike trains between facilitatory and triggering inputs. This will be trained for each of the cells in the TDE layer since every cell has a different pair of connections from the encoding layer L0. A low value of the time constant τ_g filters long-time differences between spikes in the cell's input. While in a cell with a high value of τ_g, a long time distance between spikes at its input elicits neural response in the synaptic trace. Training this parameter, we are selecting the time difference dynamic per pair of frequencies that maximises network performance.

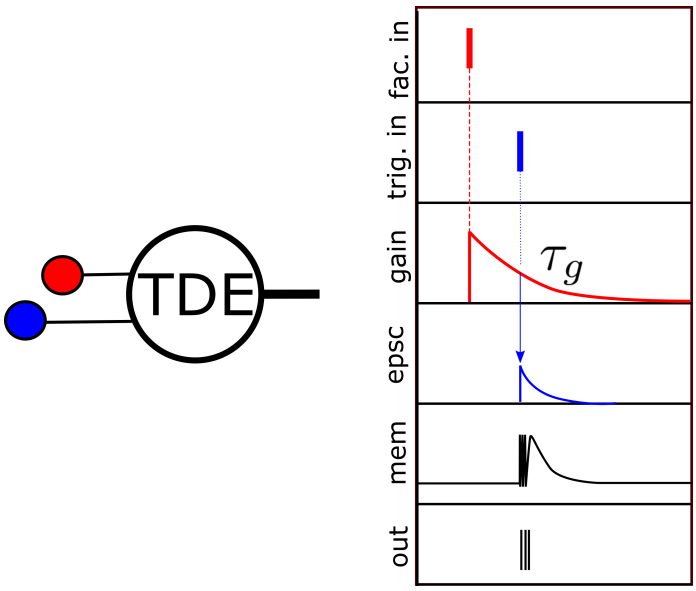


Fig. 18: Time Difference Encoder (TDE) in the presence of input spikes. τ_g parameter governs the effect of the distance between facilitatory and trigger spikes.

2.3 Offline training with back-propagation through time

We use SNNs for keyword classification and optimize the network parameters with supervised learning methods. We use BPTT with surrogate gradient descent that uses a surrogate function in the backwards propagation of the network loss $\mathcal{L}$, bypassing the spike discontinuity that represents an issue with applying the derivative of the signal. This method has been proven to be highly effective in training SNNs [138–140]. Following previous works, we exploit the differentiation capabilities by overloading the derivative of the spiking nonlinearity with a differentiable fast sigmoid function with λ as the scale parameter of the slope:

$$\sigma(U_i^{(l)}) = \frac{U_i^{(l)}}{1 + \lambda |U_i^{(l)}|} \tag{25}$$

In order to apply the backpropagation method to the train of spikes, we first use the cross entropy loss function over the active number of spikes in the output layer (L2 in the network architecture), whose each individual cell represents a keyword class. Formalizing the function for a given batch number(N_b) and number of classes(N_c) for the loss function as:

$$\mathcal{L} = -\frac{1}{N_b}\sum_{s=1}^{N_b} \mathbb{1}(i = y_s) \cdot \log\left\{ \frac{\exp\left(\sum_{n=1}^{T} S_i^{(l)}[n]\right)}{\sum_{i=1}^{N_c} \exp\left(\sum_{n=1}^{T} S_i^{(l)}[n]\right)} \right\} \tag{26}$$

then we propagate the loss function to the parameters of the network as:

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial \mathcal{L}}{\partial W_{ij}} \quad , \quad \tau_g \leftarrow \tau_g - \rho \frac{\partial \mathcal{L}}{\partial \tau_g} \tag{27}$$

where W_{ij} describes the training of the weights and the τ_g the parameter of the TDE cells for the TDE network as described in Section 2.2.2. The parameters η and ρ define the learning rate. Then, we used an ADAptive Moment estimation (ADAM) optimizer algorithm to update the parameters based on the gradient functions with dropout and weight decay. These two methods are typically used in neural networks to improve the speed in the generalization of the training and avoid overfitting. For the rest of the hyperparameters of the network, we empirically select, based on performance, a range of values based on previously reported network hyperparameters shown in Table 10. Using typical machine learning methods, we report here the accuracy in the decoding of the testing dataset after training.

Table 10: Description of the hyperparameters used in the supervised learning procedure. Hyperparameters tuned from exploration based on previous works in training SNNs from spatio-temporal patterns with biological constraints [113].

Hyperparameter	Description	Value
scale, λ	Steepness of surrogate gradient	5
time_bin_size	Time binning of the encoded input	0.015 [s]
sim_step_size, dt	Time between steps of the simulation	0.015 [s]
tau_mem, τ_{mem}	Decay time constant of the membrane	0.002 [s]
tau_syn, τ_{syn}	Decay time constant of the synapse	0.008 [s]
learning_rate, μ, ρ	learning rate	0.0015
weight_dec	weight decay of the optimizer in every step	0.0001
p_drop	Probability of training dropout	0.1

2.4 Comparison of the performance of different network models

We train three different networks to compare the results' performance, efficiency and interpretability. The networks are represented in Figure 17 as TDE, LIFrec and LIF networks. All networks explored consist of three layers. Input layer L0 is defined as the encoding layer since it

converts the energy of the formant frequencies into spikes and consists of 32 input neurons as described in Section 2.1. The hidden layer L1 consists of different neuron models based on the description in Section 2.2. The output layer L2 is a fully connected layer with a parameter to optimise defined as W_2. The keyword class is decoded as the cell with higher spike activity in L2, which is formed by 11 cells, one cell for each of the classes in the dataset.

With the aim of comparing the networks, we first balance the number of cells in the hidden layer L1 based on the total number of connections. Accounting with the fixed number of cells in L0 and L2 for all networks (N_{L0} as 32 frequency bands, N_{L2} as 11 keyword classes), the number of connections is described in function of the number of cells in L1 (N_{L1}) as:

$$\begin{aligned} TDE \quad & Conn = N_{L1} \cdot 2 + N_{L1} \cdot N_{L2} \\ LIFrec \quad & Conn = N_{L1} \cdot N_{L0} + N_{L1} \cdot N_{L1} + N_{L1} \cdot N_{L2} \\ LIF \quad & Conn = N_{L1} \cdot N_{L0} + N_{L1} \cdot N_{L2} \end{aligned} \tag{28}$$

Then, we adjust the number of cells (N_{L1}) based on the same number of connections for the three networks. The same number of connections means that the three networks have the same synaptic weight capacity, which serves as a proxy of the memory footprint used by a hardware implementation. The number of connections is defined by the selected number of cells in the TDE network that leads the comparative study.

We compare the networks in terms of keyword classification performance, energy efficiency measured as the number of synaptic operations per decoded keyword, and interpretability of the computations. When comparing performance in keyword spotting, we train each network type according to its defined parameters time constant τ_g, weight per layer W_1, W_2 and recurrent weight W_{11}. In the TDE network the parameters are τ_g and W_2, in LIFrec W_1, W_{11}, W_2, and in LIF W_1, W_2. For the training procedure, we randomly divided the dataset between train and test, keeping the same number of samples per class. For each epoch, the training loop iterates through all the training datasets, optimizing the parameters to minimize the loss function and calculating the classification accuracy of the non-trained test dataset over a fixed number of epochs. The reported performance in the accuracy comparison results from averaging the 25 best testing accuracy results across the training epochs. This method reduces the random variability of the

testing accuracy.

In terms of efficiency, the energy expenditure is typically approximated by the sparsity level in the decoding of all the spatio-temporal sequences. Targetting a neuromorphic implementation of SNNs, we propose to consider two event-driven asynchronous computations at the level of the neuron cell that increases the dynamic energy consumption of the system: (1) the operation of integrating an input spike (i.e. add to the membrane potential), and (2) the operation of firing an output spike (i.e. reset of the membrane potential). Therefore, we calculate the number of Synaptic Operations (SynOps) per layer as described in Equation 29:

$$SynOps = \sum_j S_{in,j} + \sum_j S_{out,j} \tag{29}$$

Regarding the interpretability of the results, the optimization method is finding the optimal weight values of W_2, which maximizes the classification. Interpreting the obtained results in W_2, we quantify the contribution of the cells in L1 to each of the keyword classes in L2. This analysis is simplified in the TDE network, given its cell connectivity.

3 Results

3.1 Time pattern versus rate

The resulting spiking data from the conversion of formants into spikes present a question about how we can exploit this information for the classification task. Several works in the audio modality have already highlighted that speech, by its nature, conveys a large amount of information in the time domain. Since SNNs have been known to exploit the information about time more efficiently, we investigate these findings in our dataset. For that purpose, we use information theory to quantify the information about rate, removing the time information out of the signal, and about time, measuring the time precision of the elicited spike patterns after encoding. This methodology has been previously used to calculate the precision of the spike trains [141]. These measures are termed I_{rate} and $I_{pattern}$ respectively and are described in the left panel of Figure 19 along with the results of the metric applied to the dataset. In the figure, we

combine as mean and standard deviation the measurements about I_{rate} and $I_{pattern}$ from each of the 32 neurons that encodes the information in all the frequency band of the formants. In the figure, Δt represents the time bin which defines the precision of the time pattern code. An increase in the time bin reduces the length of the pattern for a fixed time of 400ms from the first spike (only applies to this metric). The results in the right side of Figure 19 show that the information about spike timing is even bigger than about the rate when we use a big time window, and it is equivalent in the range of 15 to 20ms. Hence, demonstrating here that the information found about time is relevant to the classification task.

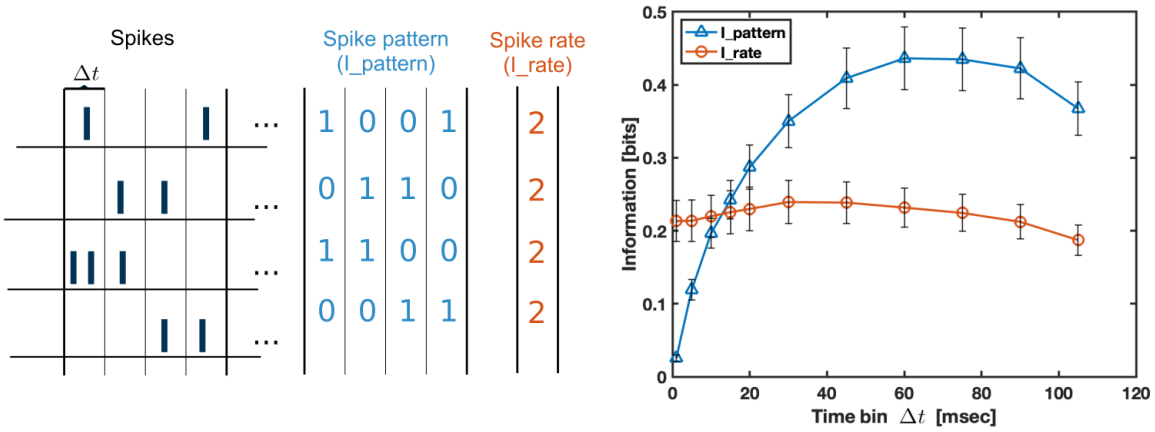


Fig. 19: Information about the spike pattern and spike rate of the encoded spikes in the dataset. The left figure represents the methodology for the calculation of I_pattern versus I_rate from the spike trains. The right figure shows the amount of information about the rate against the pattern of spikes in the encoded dataset. The mean and standard deviation for each of the individual frequency channels that serve as input in the networks.

3.2 Data-driven TDE-based network architecture search

From the cell point of view, TDE neurons present a directionality in its input as expressed in Section 2.2.2. In the proposed network composed of TDE cells, every cell is connected to two different frequency bands where the formants are combined. Following an analysis of the relevant features of the dataset, we can resize the network and guide the initialisation of the cell parameters for the training. In comparison, typical LIF and recurrent LIF architectures seen in the literature present the same connectivity and initialisation with a selection of the cell number as the degree of freedom for the designer. Building TDE networks, we can adapt the network to the dataset presented, building a minimal implementation that maximises energy efficiency and removes extra complexity in the system.

Taking a data-driven approach, we measure the unbiased cross-correlation values to every frequency pair in the spike encoding of the formant amplitudes in L0. With this process, we are able to identify the highest correlated pairs of frequencies connected to the cells in our network. Since the TDE cell behaviour maximises its output as a coincidence detector, we rank the TDE cells according to their cross-correlation values to the expected spike trains. Figure 20 presents the results of this analysis. In the left column, the top heatmap represents the average within the word class and the maximum across classes of the cross-correlation value. The bottom heatmap shows the average of the lags as they correspond to the maximum cross-correlation value expressed in the top figure. This lag value will serve as an initialisation of the time constant τ_g that defines the pre-synaptic connection of each TDE cell.

Given the cross-correlation value, we rank the TDE cells in L1 and redimension the network prioritising the TDE cells with higher cross-correlation values. In the right column of Figure 20, we dimension the network pruning at different levels of cross-correlation values as expressed in the red lines in the top histogram of the figure. We train each of the networks at these levels with the corresponding number of cells and measure the corresponding test accuracy expressed in the bottom figure. It is visually compared the informed pruning of the network with a random pruning that outputs worse performance at all levels. A more compact view of the properties of these networks (number of cells, connections, and accuracy) can be seen in Table 11. From the results, this initial analysis of the dataset leads to reduce 45.5$\%$ in the number of neurons and connections with only a penalty performance of 1.22$\%$ for the network with 540 TDE cells, as seen in the mentioned table. Given its efficiency, we will use this network in the following sections for a more detailed comparison between network types.

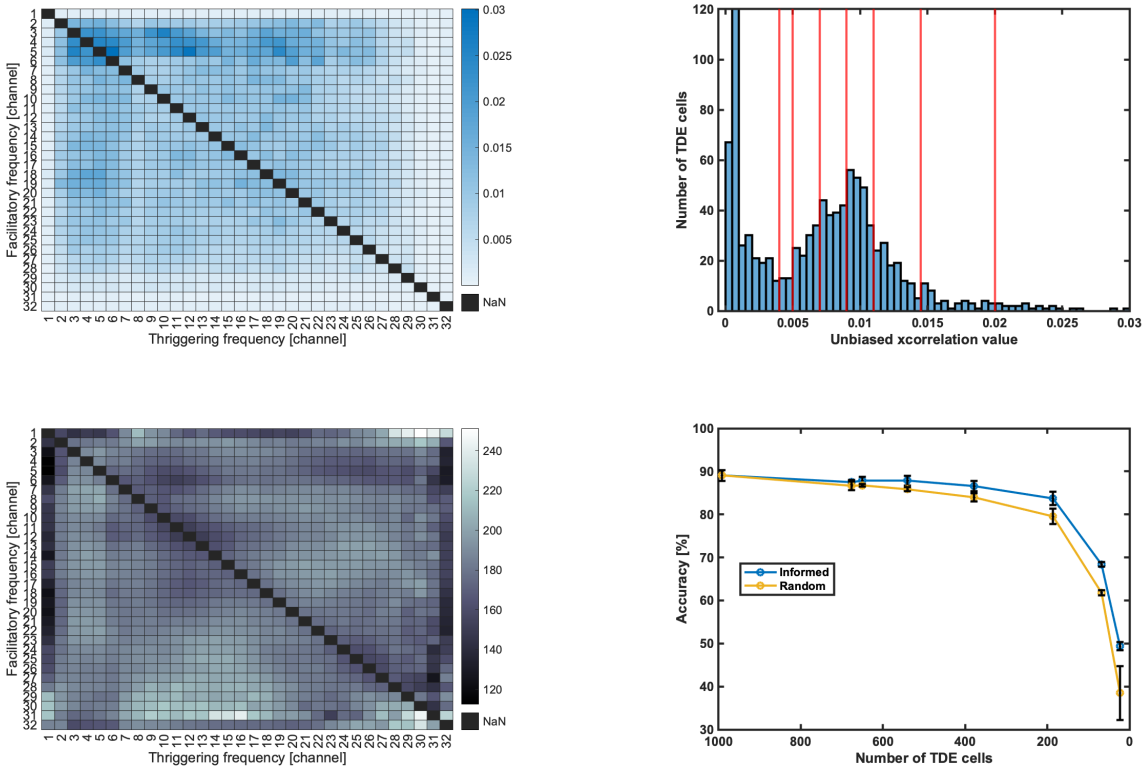


Fig. 20: Cross-Correlation values of pairs of frequencies and TDE cell selection with accuracy results. Top-left: unbiased cross-correlation values for each frequency pair in the dataset calculated as the mean within class and maximum value across class. Bottom-left: average lag between classes which outputs the maximum value of cross-correlation. Top-right: histogram of all possible combinations of TDE cells and associated cross-correlation values of spike input. Red lines defined the pruning levels selected. Bottom-right: accuracies associated with the training of networks at specified pruning levels and correspondent number of TDE cells. Comparative figure between the data-driven informed pruning against a random pruning.

3.3 Classification accuracy and inference efficiency

For the following comparative study, we choose different sizes of TDE networks based on cross-correlation encoded spike trains in L0 from the data as described in Section 3.2. We select three different network sizes in the TDE architecture (992, 540, and 186 cells) and balance the number of neurons of LIFrec and LIF based on the number of connections of the network as explained in Section 2.4.

Balancing the total number of connections instead of comparing the networks based on a number of cells allows us to better approximate the amount of resources that these networks allocate in an

Table 11: Table of cross-correlations, an equivalent number of cells, and comparison between cross-correlation informed and random pruning in terms of network performance for the TDE network. Data-driven pruning of the TDE cells allows us to explore the trade-off between network complexity and performance needed in a hardware implementation. The accuracy values are calculated as the mean and standard deviation of 3 independent learning iterations. The calculated test accuracy is the resultant of the average of the top 25 best test accuracies after 3000 epochs using Backpropagation Through Time (BPTT).

Cross-Corr	**# TDE cells**	**TDE Accs (std) xcorr**	**TDE Accs (std) random**
all	992	89.08 (1.25)	89.08 (1.25)
0.004	676	87.49 (0.47)	86.64 (0.97)
0.005	650	87.86 (0.88)	86.64 (0.27)
0.007	540	87.86 (1.12)	85.83 (0.46)
0.009	378	86.58 (1.22)	83.95 (0.92)
0.011	186	83.7 (1.55)	79.53 (1.77)
0.0145	68	68.36 (0.54)	61.77 (0.63)
0.02	23	49.35 (0.98)	38.47 (6.30)

implementation. Furthermore, it also approximates the complexity between architectures in terms of the number of synaptic weights and trainable parameters. The difference is given due to the fact we trained the τ_g of the TDE cells instead of the connected input weight. The comparison in terms of accuracy for all networks is shown in Table 13. From all the results, the recurrent LIF (LIFrec) network presents the highest accuracy value with 91% in comparison with the 89% of the TDE architecture. These values correspond to the bigger number of cells of the networks tested that correspond to all the frequency pair combinations in the dataset. While all the networks decrease the accuracy when reducing the number of cells, the LIF architecture seems to reach its maximum capability in the range of 160 neurons in L1.

We perform further comparisons with the first level of the network pruning since this network corresponds to a smaller size without a significant decrease in performance. In Figure 21 and Table 12 are presented the comparative results between the three networks in terms of test accuracy, the total number of elicited spikes of all cells of the trained network, and the number of synaptic operations. All the metrics are presented per keyword in an average of the total number of spikes decoding the full dataset. Synaptic operations are calculated as the total operations computed in the cells; we considered the computational events in asynchronous implementations as the ones elicited by a spike in the input of a cell and the elicited spike in their output. This metric combines the sparsity of the network with the connectivity between layers as a single

metric of the computational cost in the network.

We found that, although the TDE network is, on average, 2% lower in accuracy, the number of spikes and the total synaptic operations in the networks are considerably lower for the TDE architecture. In comparison, the TDE network generates 53% fewer spikes and computes 92% less synaptic operations along the network against the LIFrec network. This result is of paramount importance because of the direct impact of the synaptic operations on the energy consumption of the neuromorphic computing system. When comparing different layers of the Synaptic operations, we observe how the considerable number of spikes in the L1 of the LIFrec architecture not only elicit an increase in the number of synaptic operations in L1 (the layer where recurrent cells are) but also in the following layer L2 which is formed by the same number of LIF cells in all three architectures.

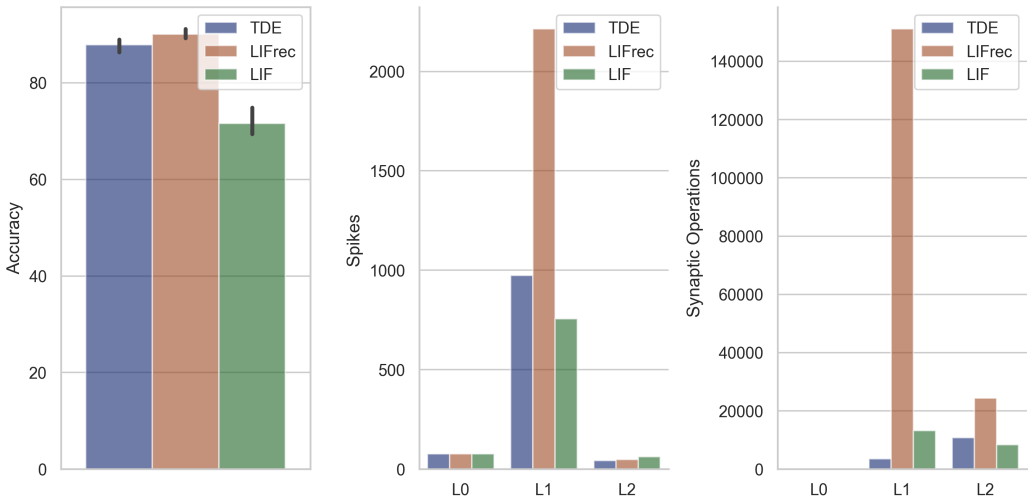


Fig. 21: Comparison of spikes per layer and synaptic operations in decoding the full dataset. Spikes at each layer are calculated as the total number of evoked spikes at each layer for the whole dataset. Spikes in the middle subfigure are defined as the number of spikes in the output of all cells in their respective layer. Synaptic operations account for the connectivity of each of the layers, and every spike evokes an operation in the event-based computation architectures. For instance, spikes in the output of L0 are computed in every single neuron of L1 for the LIF network since it is fully connected, but only a fraction of that is computed in a single cell of L1 for the TDE network.

		TDE		**LIFrec**	**LIF**	
Accuracy (std)		87.86 (1.12)		90.04 (0.8)	71.59 (2.72)	
# cells L1		540		65	163	
Synaptic connections		7020		7020	7009	
Trained parameters		6480		7020	7009	
Spikes		#	Diff	#	#	Diff
Layer	L0	77	0.00%	77	77	0.00%
	L1	975	-55.95%	2213	757	-65.78%
	L2	43	-12.37%	49	63	29.52%
	Total	1095	-53.19%	2339	898	-61.62%
Synaptic Operations		#	Diff	#	#	Diff
Layer	L0	77	0.00%	77	77	0.00%
	L1	3572	-97.64%	151076	13299	-91.20%
	L2	10768	-55.86%	24395	8394	-65.59%
	Total	14417	-91.79%	175548	21771	-87.60%

Table 12: Comparative table of spikes and synaptic operations per keyword averaged over the whole dataset for the tested architectures after training in the decoding of the whole dataset. Diff column describes the percentual difference against LIFrec networ

Table 13: Comparative table of cross-correlations and equivalent network architectures. While the number of cells in the TDE network is fixed (based on the preliminary analysis of the data at different levels, see cross-correlation figure for explanation) the number of cells for LIFrec and LIF is calculated to have the same number of connections. The accuracy values are calculated as the mean and standard deviation of 3 independant learning iterations randonmly initialized. The calculated test accuracy is the resultant of the average of the top 25 best test accuracies after 3000 epochs using Backpropagation Through Time (BPTT) to learn the trainable parameters (which are different for each architecture weights and tau time constant of the TDE cell).

Cross-Corr >=	# TDE cells	Connections	TDE Accs (std)	# LIFrec	LIFrec Accs (std)	# LIF	LIF Accs (std)
all	992	12896	89.08 (1.25)	94	91.11 (1.07)	300	71.46 (0.44)
0.007	540	7020	87.86 (1.12)	65	90.04 (0.80)	163	71.59 (2.72)
0.011	186	2418	83.70 (1.55)	32	83.03 (5.14)	56	68.73 (0.34)

3.4 Training data efficiency

It is known that the performance of neural networks (ANN and SNN) trained with ground truth (i.e. supervised learning) using backpropagation is highly sensitive to the size of the dataset [142, 143]. The size of the training dataset is an important factor, hence the use of techniques to artificially generate bigger datasets injecting noise to improve training performance in what is termed as data augmentation. But it is not always possible to have big datasets, and the computational effort of the training is also a significant factor. Given the dataset presented here, we compare the training efficiency of TDE and recurrent LIF (LIFrec) cells since these two are the ones that present the highest performance in the previous experiments. For that purpose, we reduce the size of the training dataset keeping the test dataset at the same size to examine the inference power of the network against a smaller number of samples per class. The comparative results are shown in Figure 22. From the figure, we appreciate a bigger decreasing slope for the LIFrec than for the TDE architecture, thus expressing a higher sensitivity to the decrease in the training size. The TDE architecture shows no statistical significant change in the performance with a reduction of the training dataset from the 100% available to the 75% while LIFrec shows a statistically significant decrease. The decrease in performance is, in general, bigger in size for the LIFrec architecture showing that TDE architecture, in comparison, is more robust and efficient in the presence of smaller datasets.

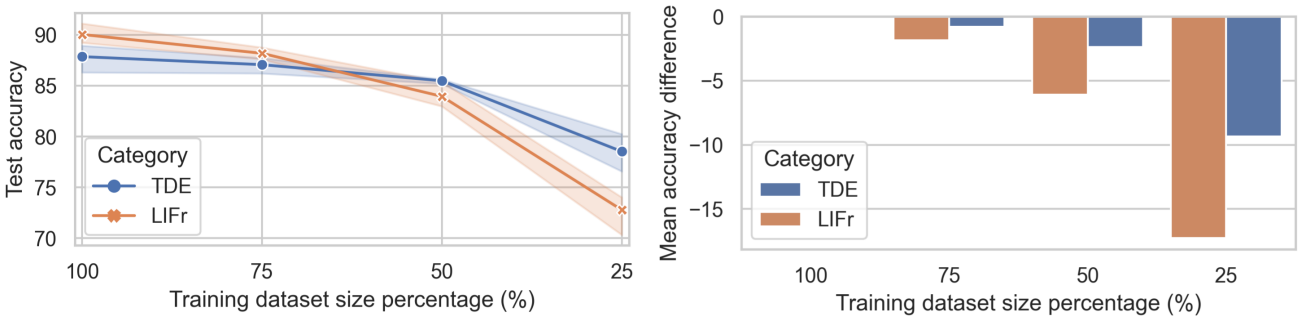


Fig. 22: Comparison TDE versus LIF recurrent in terms of training efficiency. All figures are plotted against a reduction in the training dataset in size percentage. 100% represents the full training dataset available. The left figure compares the decrease in accuracy for both of the models with the highest accuracy with the decrease in the training dataset. The right figure shows the difference in accuracy against the reduction. TDE architecture is more robust to a reduction in the training dataset.

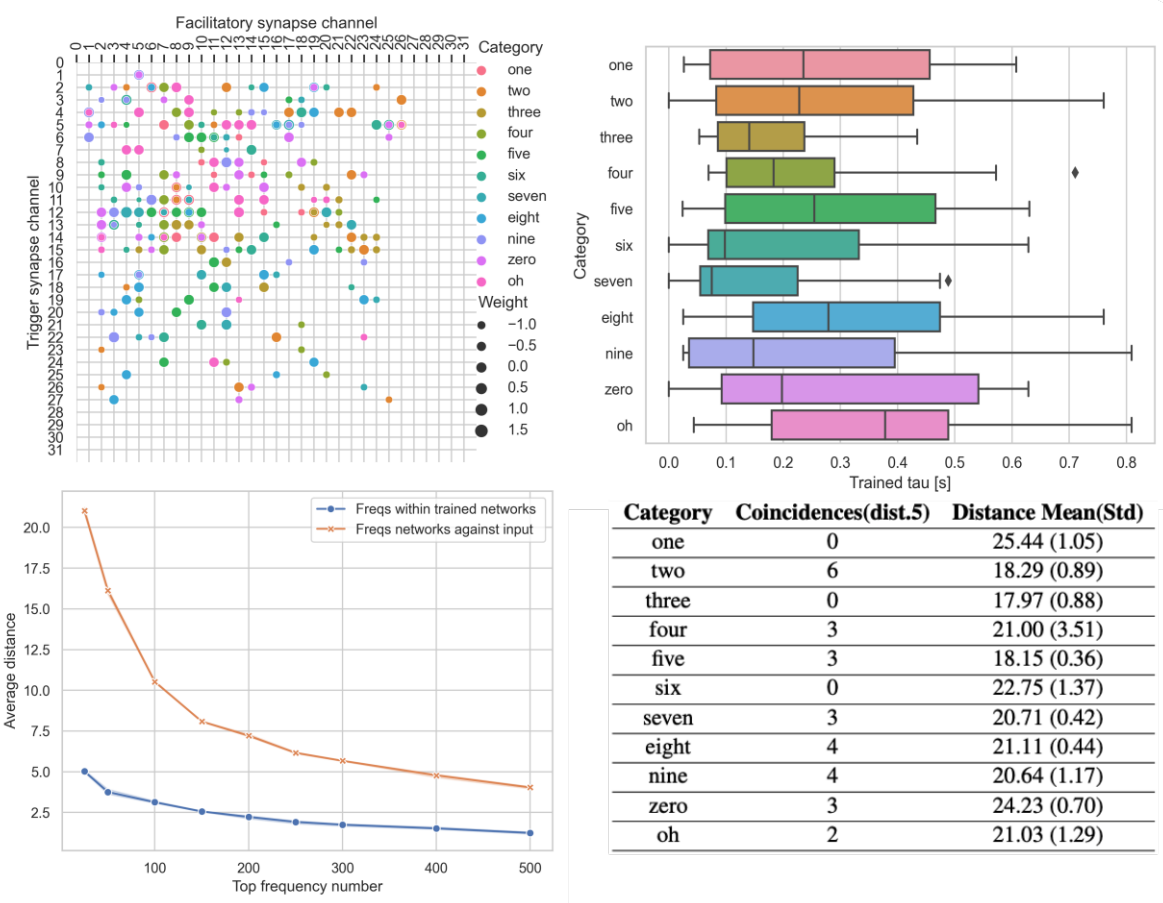
3.5 Interpretation of network parameters

Given the pre-synaptic compartmentalisation of the TDE cells, every cell has distinct connectivity. This connectivity of the TDE cell allows us to further interpret the network trained parameters. We can further analyse the features of the dataset that classify each of the keywords based on the trained parameters of the network. Every cell of the output layer (L2) corresponds to a different class; the trained weights that connect the output of L1 with L2 also define their relation with specific TDE cells. Furthermore, every cell is defined by its frequency pair and time constant τ_g. Hence, linking each keyword class of the dataset with the most relevant frequency pairs at a determined time constant.

Figure 23 it is represented the TDE cells, associated with its frequency pair, for each class selecting the highest weight values of this class in the L2 and the corresponding distribution of time dynamics for each class expressed as the TDE τ_g value. In the left figure, we observe that although all values are concentrated in the first half (frequencies 0 to 1500Hz from the total 3000Hz), there is a small overlap between classes. This fact shows that the trained procedure has found distinct features for each class, creating a local code based on the frequency pair. The right side of the figure represents the distribution of the τ_g time constant of these TDE cells. Following the boxplot figures, we can characterise further the time dynamics in which the frequency pairs features of the keyword maximise its output. Representing here the time lag between spikes in the frequency channels. For instance, a longer time constant here defines, on average, a longer characteristic peak of energy between 2 frequencies, as seen in the word "oh" against a sorter

timescale, as we can observe in the word "seven" in the right figure results.

The bottom-left figure and bottom-right associated table in Figure 23 compares the trained results with the initial data-driven analysis of the system. We compare the pairs of frequencies in the top TDE cells represented in the top-left figure from the training procedure with the most highly cross-correlated pairs of frequencies per class. We sort them based on the proximity between pairs, and we calculate the distance between the individual frequency channels. The results show a small number of coincidences between these values. Therefore, while the input analysis could discriminate between the most highly correlated pairs of frequencies, these highly correlated values are not all necessary. This is more clearly shown in the bottom-left figure that compares the average distance between frequency pairs calculated in different trained networks against a higher average distance displayed by the frequencies trained against the input data analysis.



Category	Coincidences(dist.5)	Distance Mean(Std)
one	0	25.44 (1.05)
two	6	18.29 (0.89)
three	0	17.97 (0.88)
four	3	21.00 (3.51)
five	3	18.15 (0.36)
six	0	22.75 (1.37)
seven	3	20.71 (0.42)
eight	4	21.11 (0.44)
nine	4	20.64 (1.17)
zero	3	24.23 (0.70)
oh	2	21.03 (1.29)

Fig. 23: Frequency pairs and time constant (τ_g) are relevant for the classification. Training of output weights and taus in the TDE architecture with all frequency pairs. Representation of the top 25 biggest weights in absolute value after training the network. In the top-left figure, each TDE cell in the network is represented by a dot in the grid describing its connectivity in the input frequency. Keyword classes with minimum frequency pair overlap. The top-right figure represents the distribution of trained taus per class of the top 25 highest weights in the output layer. We are characterising the temporal dynamics of each keyword in the frequency pair. Bottom-left figure analysis of frequency pair distance between network solution(blue) and network solutions against initial cross-correlation frequency pairs. Bottom-right table of average distance and coincidental frequency pairs between cross-correlation inputs and TDE cells with the top 25 values ranked by weight size and cross-correlation value.

4 Conclusions

In this work, we trained SNNs using supervised learning methods to maximize classification performance with three different architectures. The dataset contains significant information in the temporal pattern of spikes encoding the energy of the main frequencies in the shape of the speech formants. The results show high performance when training TDE and recurrent LIF networks with 89% and 91% accuracy, respectively, highlighting the importance of the temporal

information in this task. In our experiments, the use of CUBA-LIF cells for the feedforward network (i.e. without recurrency) is not performing as well in comparison with the other networks. Indeed, recurrency in LIF networks has been shown to improve performance when datasets contain significant information in the time domain [113, 144, 145]. While the aim of this work focuses on the comparison between neural models for SNN implementations, the hyperparameters have been obtained by exploring a range of values guided by previous insights in [113]. Future work will explore with procedural tuning the most optimized hyperparameters for each of the networks. In the same manner, following the analysis of the amount of information in the time pattern versus the rate, we will explore binning and simulation times in order to exploit the maximal information in the time pattern. These results suggest decreasing the time precision in the binning of the dataset to maximize the time information of the pattern.

Typical sparsity measurements defined the Synaptic Operations (SynOps) as the total number of spikes, using this measurement as a proxy for energy consumption [146, 147]. We calculated the number of synaptic operations considering the number of spikes at the cell's input (accounting for connectivity) plus the elicited spikes. In the comparison, we found this measurement more informative about the energy efficiency properties of the network, independent of its hardware implementation. In our experiments, the networks differ in the hidden layer L1, but the results in terms of SynOps in the L2 (with all CUBA-LIF cells) show, in comparison, an increase in computations in the LIFrec network. Further notice this effect in terms of scalability since increasing the depth of the LIFrec network, or the number of cells, would result in further energy consumption due to increased computations deep down in the network hierarchy. In terms of total efficiency, the TDE cell has an extra synaptic compartment in comparison with LIF cells, which computes an extra synaptic trace (referred to as the Gain in Figure 18), and gives the TDE a richer temporal dynamics which is useful for temporal pattern recognition. However, it has far simpler connectivity with two inputs only. Hence, overall, the TDE has about 92% less synaptic operations than the CUBA-LIF recurrent network. Neuromorphic implementations exploit this gain by using event-driven asynchronous processing in which synaptic operations are only processed in the presence of an event or spike, thus highly reducing the total energy consumption. Nevertheless, our metric is an analytical approximation of the real energy consumption which depends on other operations depending on the particular neuromorphic implementation. In digital

neuromorphic processors, the state of every leaky compartment should be updated at each algorithmic time step, which can induce a significant constant energy consumption [148]. It is possible to optimize the architecture so that the states are only updated in the presence of an input spike. Considering mixed-signal analog-digital neuromorphic circuits as a candidate for the system implementation, there is no algorithmic time step and the leakage in each compartment of the cell is emulated with the physical properties of the devices (e.g. capacitors) [149, 150], resulting in a higher energy-efficiency [151]. The neuromorphic community is pushing the boundary towards that direction with the design of mixed-signal chips such as Dynaps [152]. In these analog-digital neuromorphic circuits, the cost of the complexity of the TDE cell is insignificant in comparison with the efficiency presented in this work, with a 92% reduction in the synaptic operations. Future works will further validate the energy consumption gains of the TDE with the implementation of the networks in neuromorphic hardware platforms. Paving the way forward, the TDE neuron model has already been implemented in Intel's Loihi neuromorphic processor [4].

We also presented other advantages of using TDE neuron cells in an SNN architecture aside from the energy efficiency, i.e. data-driven scaling of the network, training data efficiency, and interpretability of the results. First, we analyse a methodology to prune our network based on the cross-correlation of our dataset input frequencies. The TDE neuron model transforms into a burst of spikes, the time difference between its input. In certain conditions, its behaviour presents a maximum spiking rate as a coincidence detector when two trains of spikes are highly correlated. By design, we built a layer of TDE cells that cover the range of frequency pairs in the input space. Analysing the cross-correlation of the input pairs allows us to identify higher correlated inputs, which will maximise the behaviour of the TDE cell. Starting with the less correlated pairs, we were able to prune almost half of the network cell's very small penalty in the performance of our network (1.22%). This data-driven methodology has proved to be informative in adapting the size of the network. Neuromorphic engineers can resize the network based on the available resources and the interplay between the performance and efficiency of the system. Second, we explore the training efficiency, and the results show a non-statistically decrease in

[4] A. Renner and L. Khacef, "Neuromorphic implementation of TDE in Loihi," https://github.com/intel-nrc-ecosystem/models, 2022.

performance, reducing the training dataset to 75% of its size when using TDE neurons. In comparison with the recurrent LIF, the TDE network has similar optimisation parameters as we calculate the number of cells in $L1$ of each network to balance the synaptic connectivity of the network that also approximates the number of trainable parameters. While the Gain parameter in the TDE cell can also be trained, we found it more relevant and sufficient to optimise τ_g. This parameter also describes the time distance between input spikes that is encoded in its output. The results in training efficiency show the TDE network is more robust to changes in the dataset and generalises faster to optimal solutions.

Finally, the TDE model contributes to increasing the interpretability of the results in our networks which is a common issue in the training of general Neural Networks [153]. The use of TDE cells and their distinctive connectivity allow us to apply direct correspondence between the trained parameters of the network and the key features of the dataset. Every TDE in our network is associated with a pair of frequencies through the input connectivity and with a specific time constant τ_g. The results from the trained weights in the output layer (L2) present a distinctive code of TDE cells (L1) that contributes to each keyword class discrimination. In the same way, we linked the trained τ_g for each TDE to the most representative timescale for the coincidence of amplitude peaks in the frequency spectrum of formants. Therefore, we characterise the features of the dataset in the spatial dimension as a set of frequency pairs for each keyword. Furthermore, the time dimension is the associated timescale of the temporal dynamic of the TDE cells that contribute the most to the classification of each keyword. These distinctive features for classification cannot be obtained with an analysis of the data as the one used for the pruning of the network in light of the comparative results. The features obtained for the classification are, on average, very distant in space to the most highly correlated frequencies of the initial analysis. Although the analysis of the dataset seems insightful to the pruning method, the most informative frequencies of the classification do not correlate with the most cross-correlated input frequencies of the dataset. Cross-correlation of input spike train pairs is informative for network initialisation and performance. However, a maximal coincidence between these trains of spikes is not extracting the most relevant features for the classification task.

The promising results found in this work point towards the computational efficiency of the TDE neuron model for exploiting spatio-temporal patterns of information that are present in all

sensory systems. The future prospect in the neuromorphic field of highly efficient digital chips and mixed-signal analog-digital circuits indicates a new generation of energy-efficient hardware. The combination of network architectures that exploit sensory information with highly efficient hardware has the potential to power a new generation of low-power, low-latency devices with computations at the edge for full event-driven systems.

References

[97] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, oct 2021. [Online]. Available: https://doi.org/10.1145/3469029

[98] N. A. Sulieman, L. Ricciardi Celsi, W. Li, A. Zomaya, and M. Villari, "Edge-oriented computing: A survey on research and use cases," *Energies*, vol. 15, no. 2, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/2/452

[99] N. C. Thompson, K. H. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *CoRR*, vol. abs/2007.05558, 2020. [Online]. Available: https://arxiv.org/abs/2007.05558

[100] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "Deep learning's diminishing returns: The cost of improvement is becoming unsustainable," *IEEE Spectrum*, vol. 58, no. 10, pp. 50–55, 2021.

[101] J. M. Rabaey, M. Verhelst, J. D. Boeck, C. Enz, K. D. Greve, A. M. Ionescu, M. Na, K. P. (editors) with F. Conti, F. Corradi, K. D. Greve, M. D. Ketelaere, B. Dhoedt, M. Hartmann, K. Myny, I. Ocket, W. Philips, W. Verachtert, and D. Verkest, "Ai at the edge - a roadmap," *Technical report by IMEC, KU Leuven, Ghent University, VUB, EPFL, ETH Zurich and UC Berkeley*, 2019.

[102] L. Khacef, N. Abderrahmane, and B. Miramond, "Confronting machine-learning with neuroscience for neuromorphic architectures design," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[103] D. Roy, P. Panda, and K. Roy, "Synthesizing Images From Spatio-Temporal Representations Using Spike-Based Backpropagation," vol. 13, no. June, pp. 1–11, 2019.

[104] D. Ivanov, A. Chezhegov, M. Kiselev, A. Grunin, and D. Larionov, "Neuromorphic artificial intelligence systems," *Frontiers in Neuroscience*, vol. 16, 2022.

[105] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117

[106] S. Y. A. Yarga, J. Rouat, and S. U. N. Wood, "Efficient spike encoding algorithms for neuromorphic speech recognition," vol. 1, no. 1, 2022. [Online]. Available: http://arxiv.org/abs/2207.07073

[107] Z. F. Mainen and T. J. Seinowski, "Reliability of spike timing in neocortical neurons," *Science*, vol. 268, no. 5216, pp. 1503–1506, 1995.

[108] C. Kayser, N. K. Logothetis, and S. Panzeri, "Millisecond encoding precision of auditory cortex neurons," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 39, pp. 16 976–16 981, 2010.

[109] Y. Zuo, H. Safaai, G. Notaro, A. Mazzoni, S. Panzeri, and M. E. Diamond, "Complementary contributions of spike timing and spike rate to perceptual decisions in rat S1 and S2 cortex," *Current Biology*, vol. 25, no. 3, pp. 357–363, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cub.2014.11.065

[110] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[111] M. Davies, T.-h. Lin, C.-k. Lin, S. Mccoy, Y.-h. Weng, A. Wild, and H. Wang, "Loihi : A Neuromorphic Manycore Processor with On-Chip Learning," no. February, 2018.

[112] E. Ceolini, C. Frenkel, S. B. Shrestha, G. Taverni, L. Khacef, M. Payvand, and E. Donati, "Hand-gesture recognition based on emg and event-based camera sensor fusion: A benchmark in neuromorphic computing," *Frontiers in Neuroscience*, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00637

[113] S. F. Müller-Cleve, V. Fra, L. Khacef, A. Pequeño-Zurro, D. Klepatsch, E. Forno, D. G. Ivanovich, S. Rastogi, G. Urgese, F. Zenke, and C. Bartolozzi, "Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware," *Frontiers in Neuroscience*, vol. 16, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2022.951164

[114] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *Proceedings of the 7th Annual Neuro-Inspired Computational Elements Workshop*, ser. NICE '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3320288.3320304

[115] L. Welling and H. Ney, "Formant estimation for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 1, pp. 36–48, 1998.

[116] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[117] E. Wong and S. Sridharan, "Comparison of linear prediction cepstrum coefficients and mel-frequency cepstrum coefficients for language identification," *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, ISIMP 2001*, pp. 95–98, 2001.

[118] Z. Pan, H. Li, J. Wu, and Y. Chua, "An Event-Based Cochlear Filter Temporal Encoding Scheme for Speech Signals," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, 2018.

[119] B. Prica and S. Ilic, "Recognition of vowels in continuous speech by using formants," *Facta universitatis - series: Electronics and Energetics*, vol. 23, no. 3, pp. 379–393, 2010.

[120] M. Stanek and L. Polak, "Algorithms for vowel recognition in fluent speech based on formant positions," *2013 36th International Conference on Telecommunications and Signal Processing, TSP 2013*, pp. 521–525, 2013.

[121] M. Coath, S. Sheik, E. Chicca, G. Indiveri, S. L. Denham, and T. Wennekers, "A robust sound perception model suitable for neuromorphic implementation," *Frontiers in Neuroscience*, vol. 7, no. 8 JAN, pp. 1–10, 2014.

[122] L. Laszko, "Using formant frequencies to word detection in recorded speech," *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016*, vol. 8, pp. 797–801, 2016.

[123] S. Panzeri, N. Brunel, N. K. Logothetis, and C. Kayser, "Sensory neural codes using multiplexed temporal scales," *Trends in Neurosciences*, vol. 33, no. 3, pp. 111–120, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.tins.2009.12.001

[124] S. Panzeri, R. A. Ince, M. E. Diamond, and C. Kayser, "Reading spike timing without a clock: Intrinsic decoding of spike trains," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 369, no. 1637, 2014.

[125] M. B. Milde, O. J. Bertrand, R. Benosmanz, M. Egelhaaf, and E. Chicca, "Bioinspired event-driven collision avoidance algorithm based on optic flow," *Proceedings of 1st International Conference on Event-Based Control, Communication and Signal Processing, EBCCSP 2015*, 2015.

[126] G. D'Angelo, E. Janotte, T. Schoepe, J. O'Keeffe, M. B. Milde, E. Chicca, and C. Bartolozzi, "Event-based eccentric motion detection exploiting time difference encoding," *Frontiers in neuroscience*, vol. 14, p. 451, 2020.

[127] T. Schoepe, E. Janotte, M. B. Milde, O. J. N. Bertrand, and E. Chicca, "Finding the Gap : Neuromorphic Motion Vision in Cluttered Environments," *arXiv*, 2021.

[128] D. Gutierrez-Galan, T. Schoepe, J. P. Dominguez-Morales, A. Jimenez-Fernandez, E. Chicca, and A. Linares-Barranco, "An Event-Based Digital Time Difference Encoder Model Implementation for Neuromorphic Systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 1959–1973, 2022.

[129] M. Mastella and E. Chicca, "A hardware-friendly neuromorphic spiking neural network for frequency detection and fine texture decoding," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[130] V. Hassenstein and W. Reichardt, "System theoretical analysis of time, sequence and sign analysis of the motion perception of the snout-beetle chlorophanus," *Z Naturforsch B*, vol. 11, no. 9–10, pp. 513–524, 1956.

[131] H. B. Barlow and W. R. Levick, "The mechanism of directionally selective units in rabbit's retina." *The Journal of Physiology*, vol. 178, no. 3, pp. 477–504, 1965.

[132] M. S. Maisak, J. Haag, G. Ammer, E. Serbe, M. Meier, A. Leonhardt, T. Schilling, A. Bahl, G. M. Rubin, A. Nern, B. J. Dickson, D. F. Reiff, E. Hopp, and A. Borst, "A directional tuning map of Drosophila elementary motion detectors," *Nature*, vol. 500, no. 7461, pp. 212–216, 2013.

[133] M. Coath, R. Mill, S. L. Denham, and T. Wennekers, "Emergent feature sensitivity in a model of the auditory thalamocortical system," in *From Brains to Systems: Brain-Inspired Cognitive Systems 2010*. Springer, 2011, pp. 7–17.

[134] F. Sandin and M. Nilsson, "Synaptic Delays for Insect-Inspired Temporal Feature Detection in Dynamic Neuromorphic Processors," *Frontiers in Neuroscience*, vol. 14, no. February, 2020.

[135] R. G. Leonard and G. Doddington, "Tidigits speech corpus," *Texas Instruments, Inc*, 1993.

[136] D. Ellis, "Sinewave Speech Analysis/Synthesis in Matlab," http://www.ee.columbia.edu/ln/labrosa/matlab/sws, 2004.

[137] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2020.

[138] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Computation*, vol. 33, no. 4, pp. 899–925, Mar 2021. [Online]. Available: https://doi.org/10.1162/neco_a_01367

[139] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[140] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[141] C. Kayser, M. A. Montemurro, N. K. Logothetis, and S. Panzeri, "Spike-Phase Coding Boosts and Stabilizes Information Carried by Spatial and Temporal Spike Patterns," *Neuron*, vol. 61, no. 4, pp. 597–608, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.neuron.2009.01.008

[142] D. Soekhoe, P. Van Der Putten, and A. Plaat, "On the impact of data set size in transfer learning using deep neural networks," in *Advances in Intelligent Data Analysis XV: 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings 15*. Springer, 2016, pp. 50–60.

[143] A. Alwosheel, S. van Cranenburgh, and C. G. Chorus, "Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis," *Journal of choice modelling*, vol. 28, pp. 167–182, 2018.

[144] N. Perez-Nieves, V. Leung, P. Dragotti, and D. Goodman, "Neural heterogeneity promotes robust learning," *Nature Communications*, vol. 12, p. 5791, 10 2021.

[145] M. S. Bouanane, D. Cherifi, E. Chicca, and L. Khacef, "Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition," 2022. [Online]. Available: https://arxiv.org/abs/2211.07761

[146] P. Blouw and C. Eliasmith, "Event-Driven Signal Processing with Neuromorphic Computing Systems," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 8534–8538, 2020.

[147] M. Sorbaro, Q. Liu, M. Bortone, and S. Sheik, "Optimizing the Energy Consumption of Spiking Neural Networks for Neuromorphic Applications," *Frontiers in Neuroscience*, vol. 14, no. June, 2020.

[148] E. Lemaire, A. Castagnetti, and J. Courtois, "An Analytical Estimation of Spiking Neural Networks Energy Efficiency," *arXiv*, 2022.

[149] G. Indiveri, E. Chicca, and R. J. Douglas, "Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition," *Cognitive Computation*, vol. 1, no. 2, pp. 119–127, 2009. [Online]. Available: http://ncs.ethz.ch/pubs/pdf/Indiveri_etal09.pdf

[150] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, 2014.

[151] A. Joubert, B. Belhadj, O. Temam, and R. Héliot, "Hardware spiking neurons design: Analog or digital?" in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–5.

[152] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps)," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 1, pp. 106–122, 2017.

[153] Y. Zhang, P. Tino, A. Leonardis, and K. Tang, "A Survey on Neural Network Interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021.

Conclusion

Different perspectives have been considered to build full event-based computational systems with neuromorphic technologies, from the sensory acquisition in robotic actuators to circuit design of neuron models in-silico. Over the studies of this work, we explore neuromorphic systems that exploit spatio-temporal patterns from sensory data, building classification systems with spiking neural networks. Our first work explores different schemes to encode time-based to event-based patterns in the tactile domain over a braille classification task. We compare Spiking Neural Networks (SNNs) with the Leaky Integrate-and-Fire (LIF) cell model with standard classifiers in terms of performance and energy consumption implementing the algorithms in Von Neumann and neuromorphic hardware systems. The second main contribution of this thesis explores a classification task of spoken keywords in the auditory domain. In this work, we explore the TDE model cell and benchmark networks built with this type of model against typical LIF model architectures, comparing their results in terms of performance, efficiency and interpretability of the results.

Sensory information is inherently spatio-temporal. In both of our explored tasks in the auditory and tactile domains, we found that the information that belongs to the time dimension in the sequence of encoded values is relevant for task performance. These results overlap with the findings in the encoding of sensory stimuli in the nervous system. The stimulus encoded in the neural population code presents high spatial and time resolution [154]. Most importantly, neuroscience findings correlate the nature of the stimulus with the resolution of the neural population code [155–157]. Therefore, artificial systems should be able to exploit both the spatial and temporal dimensions of the code. In our first study, comparing the performance with standard classifiers after removing the temporal domain, the performance is significantly lower.

Concerning the auditory task, we calculated the amount of information and precision of the temporal and spatial pattern of events resulting in a significant amount of information in the temporal domain. Although spiking neural networks are known to be efficient algorithms to exploit the temporal information of the code, in both sensory-perceptual tasks explored, we clearly show the need for recurrent connectivity to maximise task performance. In both works, the lack of recurrence in the network with LIF neurons outputs less than ten perceptual points difference in classification performance for all datasets. Even the methodological exploration of hyperparameters related to the time dynamics of the cell performed in the tactile work has not closed the gap between these two architectures. We may further explore in future work the cell's time constant value, hypothesising that the values explored in the Hyperparameter Optimization (HPO) procedure did not reach the sufficient range to capture the temporal dynamics of the pattern in the data. In the auditory work, we further explore this outcome with the TDE architecture. The performance of the TDE network is comparable to the recurrent LIF network and significantly better than the non-recurrent LIF network. However, the TDE model does not need recurrent connectivity, which is known to generate less sparse representations, given the constant closed-loop feedback of the recurrent connection. Why is the TDE model capable of exploiting the spatio-temporal information from the temporal sequences more efficiently? The TDE is a cell model that presents an extra synaptic compartment compared to the typical LIF model. Besides the TDE cell being computationally different from the LIF cell, in the TDE network, we train the time constants of the cells individually, associating a different time dynamic to each cell of the network. This result aligns with the works about the positive effect of heterogeneity parameters of cells in the network [158, 159]. Future work will explore further insights into the role of heterogeneous temporal constants in typical LIF models and the TDE computational mechanism that make it more efficient. Therefore, Spiking Neural Networks, especially implementing the TDE cell model, can effectively exploit the task's spatio-temporal information. This capability is particularly relevant for sensory-perceptual discrimination tasks that inherently encode information in the spatio-temporal domain.

A relevant topic for full-scale neuromorphic systems is the encoding of sensory information with sensor devices. Event-based sensors are the most suitable devices to exploit the full capabilities of artificial sensing systems. In this work, we utilise SNNs to exploit the spatio-temporal information

from event-based patterns. Regarding the tactile task, the sensory information is extracted from a sensorized fingertip. The fingertip encodes information in a frame-based manner, sampling at a constant frequency rate of 40 Hz. In our work, we explore different regimes of the delta-coding scheme to transform frame-based information into event-based information via software. In comparison, the event-based sensing approach on the device is entirely asynchronous and data-driven, which leads to highly sparse and higher temporal resolution, desirable capabilities for always-on, edge intelligent devices and sensory systems [160] (e.g., spike-based vision, auditory and olfactory [161–163]). In the auditory work, we proved that the temporal precision of the code carries significant information about the task. In the tactile task, we measure the effect of this conversion with a standard classifier with the LSTM model in both datasets: the frame-based time sequence and the reconstructed sequence of values after the event-based conversion. The results show that LSTM classifiers decreased their performance by measuring the information loss in 15 percent points in terms of accuracy performance (from 97 to 82 per cent). Additionally, to the information lost, we are losing information about the dynamic signal range and temporal resolution of the system without the use of event-sensing devices. In the visual domain, where event-based sensing has yielded more technologies and devices, event-based sensing has achieved up to 1000EPS (events-per-second) per pixel, matching the resolution of the human eye [164, 165]. This increase of information from the stimulus directly impacts the potential of sensory systems that utilise neuromorphic technology. The Spiking Neural Networks in the tactile task output a performance similar to the LSTM models with event-based information, but compared to the latest, they inherently compute event-based information, being more suitable to process information from this type of sensors [166]. Several advantages have been shown in the use of full event-based architectures from sensors to actuators [167, 168], from the amount generated to the data computed in parallel with neuromorphic SNNs. Event-based sensors are low-power, low-latency systems that output sparse data with highly scalable capabilities. These properties seem especially suitable in the tactile domain mimicking the large number of mechanoreceptors in the natural skin with event-based devices. Spiking neural networks, as seen in this work, are adequate to compute the amount of information generated by this data increase. On the contrary, standard classifiers have a current limit on the amount of information to be processed. The current trend with deep learning increases the complexity of their models exponentially with the dimensionality of the input [169].

Future work will explore this capability with fully event-based data acquisition to compare the amount of information, the information gained in the temporal domain, and how spiking neural networks can exploit that information for an increase in performance power.

From an engineering perspective, we are interested in classifying sensory information with the highest possible accuracy and a similar performance to the current standard approaches, given the multiple advantages found in other aspects. In our first work, we stressed the comparison between standard classifiers and spiking neural networks with the LIF based cell model. Although the maximum performance achieved is smaller for the SNN algorithmic approach, there are several advantages that we show concerning the data's sparsity and the system's potential scalability. Furthermore, we need to contextualise the results with the data conversion from frame-based to event-based information. An approach with event-based sensors will prove the full capabilities in terms of performance for a complete comparison. Summarising here the performance power with SNNs from this work, we classify 27 classes of braille letters with an event-based sequence of events from 24 channels sampled with a sensorized fingertip with a 80% performance accuracy. These results have been obtained with a 3-layer Spiking Neural Network with 450 neurons in the hidden layer. In the auditory domain, we classified 11 classes of spoken words with a 3-layer SNN using different cell models in the hidden layer (ranging from 65 to 540 cells) with a maximum performance of 91% of accuracy based on the formant frequency decomposition of the keywords. While the first work aims to compare the classification power with standard approaches, in the second work, we further investigate the capabilities of SNN algorithms in terms of efficiency, interpretability of the results, and the possibility of further implementation of neural models inspired by cortical processing. Several steps are considered from here for future work. First, to explore the inclusion of the TDE model in the somatosensory task, further validating the capabilities of the model to exploit the spatio-temporal structure of the data in the tactile domain. Additionally, the auditory task has yet to be explored methodologically with the HPO approach that it has output high-performance results exploring the high dimensionality of the hyperparameter space in the tactile task's networks. This methodology would give us a closer view of the real potential of the TDE and the network architecture's performance power.

A strong motivation for using SNN architectures is the energy efficiency capabilities. In our

work, we were able to quantify a comparison between standard algorithms running in Von Neumann architectures against neuromorphic hardware implementing SNN. We demonstrate the energy-efficient capabilities of the neuromorphic hardware systems in combination with SNN algorithms. Running an SNN with Loihi neuromorphic chip is 500 times more energy efficient than running an LSTM model with typical GPU acceleration hardware. The total energy power required for the neuromorphic implementation of the braille classification tasks is closed to 30mW. This result is relevant for implementing fully autonomous robotic systems and intelligent devices in which energy expenditure is a vital requirement. Applying this type of implementation may result in a highly significant increase in the device's autonomy. SNNs can further harness improvements by implementing brain-inspired computational models, as we have proved in our second work in the auditory domain. By incorporating the TDE model, we calculated an increase in the energy efficiency capabilities of the system. The number of synaptic operations in the TDE cell is 92% smaller than the typical LIF neuron model with recurrence. Further noting, that this kind of architecture has served as a benchmark in the comparison for the braille task. Implementing this model, we forecast an even further improvement in energy consumption of the entire system. In the keyword spotting task, we measure the amount of synaptic operations as a proxy for the system's energy consumption [170]. Future work in this direction will implement the TDE networks with neuromorphic hardware to further quantify the impact in terms of energy efficiency, similar to the comparative measurements shown in the braille task comparing the results with the measurement based on synaptic operations.

In addition to the energy consumption advantage of neuromorphic architectures, SNN algorithms present further benefits in terms of training efficiency, interpretability, and scalability of the networks. In both of the sensory tasks explored in the thesis, we use shallow network architectures of 3 layers. The most highly performing Artificial Neural Networks (ANNs) in the literature seem to require multiple hidden layers (e.g. Deep Neural Networks) or highly complex neural models that complicate the interpretability of the training results. This effect is visible in the standard networks implemented in the tactile tasks. The LSTM neural model consists of four operations in cascade, including non-linearities; Convolutional Neural Networks (CNN), typically used for image classification, require several layers with multiple functions, including convolution in the space domain, multiplication, and dimensionality reduction at different layers

of the architecture. In comparison, the shallowness and homogeneity of neuron models across layers increase the interpretability of the training results, even though the training procedures used in this work are adapted from typical supervised learning methods applied in ANNs. In that regard, in the auditory task, we explore further the impact of building shallow network architectures with the TDE providing insights into the classification task and the nature of the sensory data. The proposed TDE cell model further improves the interpretability of the results after training due to the nature of its fixed connectivity, which allows us to relate the input layer of the network with the output class of each cell. This effect has proved to have two advantages in the auditory classification task. First, with this methodology, we can select the trade-off of accuracy and resources available in our application. In the manuscript, we prove how to redimension our network with an informed pruning of the number of cells in the hiden layer based on the input correlation of its input. The results of this method inform about the less correlated pairs of input frequencies and, therefore, able to remove up to 50% of cells with a 1% decrease in performance. Following this methodology, we can adapt our network requirements to the dataset building a minimal implementation if necessary. On the second interpretability effect, the training results also inform about the most relevant frequency pairs for classifying each auditory keyword class. After the training, we obtain a series of results that allow us to understand the features of our dataset further, interpreting the features trained in our network. This impacts how to adapt to modifications in the dataset, reduction of keyword classes, and integration of new ones since we can characterise the output class from the input space based on the most relevant cells in the output layer for each class. We prove in the manuscript how stable the results are after training and how we can interpret the frequency space of the input for each classification class. It is a desirable effect since one of the biggest detriments of the use of Deep Neural networks is the lack of interpretability and the specific results of the training which leads to retrain the network when the dataset slightly changes. In terms of scalability, given the shallowness of the networks tested, we can further increase the complexity of the networks expecting an increase in performance. Furthermore, as seen in the auditory tasks, the improvements in cell models in terms of energy and sparsity of the network will be more noticeable as the spikes transmit along multiple layers in the architecture. Future work may explore the increase in the number of layers of the network for the highest possible performance

in neuromorphic datasets and its effect on the system's energy consumption.

In order to build an efficient and optimal neuromorphic system, we discussed the implications of even-based sensors and event-based algorithms. Another relevant aspect is the implementation of the algorithms in neuromorphic hardware and the design of custom neuromorphic hardware circuits. In our first comparison, we implemented the tactile classification algorithms in a digital neuromorphic chip with programmable capabilities (Loihi). This type of hardware provides reprogrammable flexibility, encapsulating multiple cores in the chip (128 in the example) with the power of simulating several hundreds of neurons per core. In this type of implementation, we strongly contributed to proving the high energy efficiency of neuromorphic implementations against typical classifiers built-in standard hardware architectures (GPU acceleration on Jetson). The design of neuromorphic circuits also compromises analog-digital mixed signal circuits for a candidate to further harness energy efficiency to build minimal edge computation devices [171]. In the auditory classification task, we highlighted a strong motivation for using these circuits in neuromorphic applications. The TDE neuron model exploits the input's spatio-temporal structure, showing better efficiency capabilities than typical LIF neuron models. However, we may argue that there is a cost to increasing the computational complexity. In neuromorphic digital hardware, the time step of the neural computations is simulated, resulting in a cost by increasing the computational complexity as predicted. However, in mixed-signal analog-digital circuits, these computations are emulated by physical devices. In this case, the rise in complexity of the TDE is negligent compared with the high energy efficiency shown, supporting the use of this type of neuromorphic circuits. We reinforce the current trend to design mixed-signal analog-digital circuits that harness further capabilities of brain-inspired computational models while providing a highly efficient implementation for artificial systems. Future work may explore the implementation of TDE networks in circuits [172], building comparisons not only with traditional systems but also with different neuromorphic architectures in order to explore the effect of increasing computational complexity in terms of energy consumption and latency of the system.

This work has explored the capabilities of SNNs in the context of sensing neuromorphic systems. For that purpose, we utilised supervised learning methods with backpropagation of error signals, which was already proven as a successful methodology for classification tasks. However,

learning algorithms for training SNNs is a very active topic in the neuromorphic field. While exploring learning algorithms is outside of the scope of this work, future work will explore the use of different learning algorithms with a focus on building online learning adaptive systems.

Our contribution in this work highlights the promising results of neuromorphic systems from the side of spike-based algorithms and especially to build full neuromorphic systems that sense and process information—showing the energy efficiency and scalability required to drive the new generation of intelligent systems that can interact and adapt in real-time their models with the real-world around them.

References

[154] S. Panzeri, E. Janotte, A. Pequeño-Zurro, J. Bonato, and C. Bartolozzi, "Constraints on the design of neuromorphic circuits set by the properties of neural population codes," *Neuromorphic Computing and Engineering*, vol. 3, no. 1, p. 012001, jan 2023. [Online]. Available: https://dx.doi.org/10.1088/2634-4386/acaf9c

[155] J. D. Victor, "How the brain uses time to represent and process visual information," *Brain Research*, vol. 886, no. 1-2, pp. 33–46, 2000.

[156] D. A. Butts, C. Weng, J. Jin, C. I. Yeh, N. A. Lesica, J. M. Alonso, and G. B. Stanley, "Temporal precision in the neural code and the timescales of natural vision," *Nature*, vol. 449, no. 7158, pp. 92–95, 2007.

[157] C. Kayser, N. K. Logothetis, and S. Panzeri, "Millisecond encoding precision of auditory cortex neurons," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 39, pp. 16 976–16 981, 2010.

[158] M. I. Chelaru and V. Dragoi, "Efficient coding in heterogeneous neuronal populations," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 42, pp. 16 344–16 349, 2008.

[159] N. Perez-Nieves, V. C. Leung, P. L. Dragotti, and D. F. Goodman, "Neural heterogeneity promotes robust learning," *Nature communications*, vol. 12, no. 1, p. 5791, 2021.

[160] A. Vanarse, A. Osseiran, and A. Rassau, "A review of current neuromorphic approaches for vision, auditory, and olfactory sensors," *Frontiers in neuroscience*, vol. 10, p. 115, 2016.

[161] J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco, "A spatial contrast retina with on-chip calibration for neuromorphic spike-based aer vision systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 7, pp. 1444–1458, 2007.

[162] V. Chan, S.-C. Liu, and A. van Schaik, "Aer ear: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.

[163] T. Wan, S. Ma, F. Liao, L. Fan, and Y. Chai, "Neuromorphic sensory computing," *Science China Information Sciences*, vol. 65, no. 4, pp. 1–14, 2022.

[164] H. Akolkar, C. Meyer, X. Clady, O. Marre, C. Bartolozzi, S. Panzeri, and R. Benosman, "What can neuromorphic event-driven precise timing add to spike-based pattern recognition?" *Neural computation*, vol. 27, no. 3, pp. 561–593, 2015.

[165] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff *et al.*, "A 1280× 720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86 μm pixels, 1.066 geps readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 112–114.

[166] M. H. Tayarani-Najaran and M. Schmuker, "Event-Based Sensing and Signal Processing in the Visual, Auditory, and Olfactory Domain: A Review," *Frontiers in Neural Circuits*, vol. 15, no. May, pp. 1–31, 2021.

[167] S. C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.conb.2010.03.007

[168] S. Sheik, M. Pfeiffer, F. Stefanini, and G. Indiveri, "Spatio-temporal spike pattern classification in neuromorphic systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8064 LNAI, pp. 262–273, 2013.

[169] Y. LeCun, "deep learning hardware: past, present, and future," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 12–19.

[170] P. Blouw and C. Eliasmith, "Event-Driven Signal Processing with Neuromorphic Computing Systems," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 8534–8538, 2020.

[171] A. Rubino, M. Payvand, and G. Indiveri, "Ultra-low power silicon neuron circuit for extreme-edge neuromorphic intelligence," *2019 26th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2019*, pp. 458–461, 2019.

[172] D. Gutierrez-Galan, T. Schoepe, J. P. Dominguez-Morales, A. Jimenez-Fernandez, E. Chicca, and A. Linares-Barranco, "An Event-Based Digital Time Difference Encoder Model Implementation for Neuromorphic Systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 1959–1973, 2022.