STUDY AND IMPLEMENTATION OF NEW COMPUTATIONAL PARADIGMS EXPLOITING NEUROMORPHIC HARDWARE ARCHITECTURES

Evelina Forno

Politecnico di Torino
October 16, 2023

INTRODUCTION

- The rise of embedded devices, the Internet of Things (IoT), and edge computing has transformed the way we interact with technology
- Machine learning (ML) techniques in IoT and edge computing to optimize performance, reduce latency, and improve accuracy
- Deploying ML models directly on the remote endpoint provides advantages
  - Real-time inference for lower latency in time-critical applications
  - Data retention on device increases reliability & privacy
- Artificial Neural Networks (ANNs): a flourishing area of ML
  - Too memory- and power-hungry to run on edge systems
  - End of Moore's law leads to exploration of novel platforms

INTRODUCTION

- **Neuromorphic systems**: design innovations inspired by neuroscience and biology
  - Computational model: Spiking Neural Network (SNN)
    - Biologically-inspired
    - Sparse internal activity
  - Event-driven architecture $\Rightarrow$ low power consumption
  - Custom routing for effective data interchange
- Neuromorphic systems **meet edge computing requirements**
  - Low power consumption + localized memory + real-time response

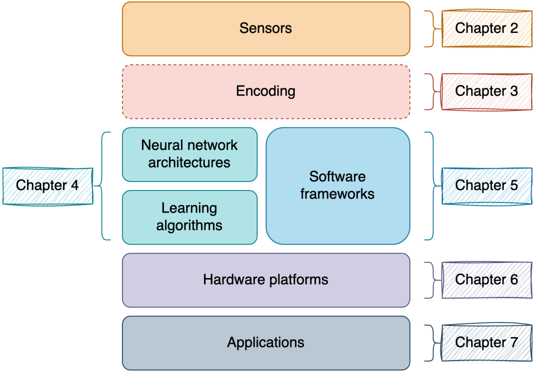
SPIKING NEURAL NETWORKS

- Information in the form of ***spikes*** (aka *action potentials*)
- Computational capabilities comparable to equivalent ANNs while consuming less power
  - A good choice for embedded applications
- Biologically-inspired learning algorithms support online learning
- Well suited for processing temporal information
- Need substantial data transfer between computational and memory units
  - Von Neumann architectures are not suitable
  - **Neuromorphic processors**: colocation of memory and computation

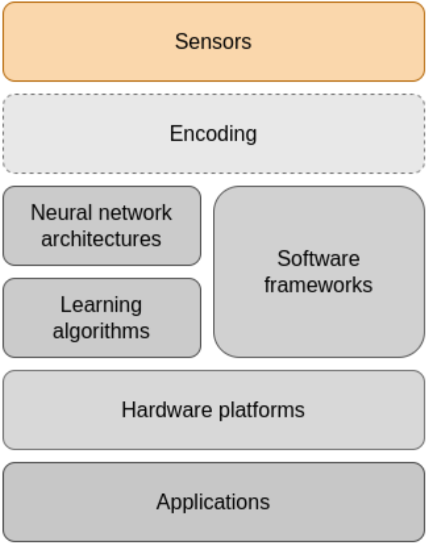
THE BUILDING BLOCKS OF A NEUROMORPHIC PIPELINE

- This dissertation's goals:
  - describe a general approach to the generation of neuromorphic models
  - facilitate the modeling process and the exploration of new neuromorphic computational paradigms
  - implement data analysis and integration in the industrial field
- Need to examine all elements of the neuromorphic pipeline

THE BUILDING BLOCKS OF A NEUROMORPHIC PIPELINE



- The structure of the thesis follows the flow of data in a neuromorphic pipeline
- Each section an overview of a layer of the pipeline stack
- Main task: classification of time-varying signals



EVENT-DRIVEN AND STANDARD SENSORS

EVENT-DRIVEN AND STANDARD SENSORS S

- Input data acquisition from IoT sensors
- Neuromorphic platforms implemented close to the sensors can perform NN-based tasks locally
  - Limit dependency on cloud computing
  - Respect power constraints
  - Maintain data integrity and privacy
- Event-based encoding significantly reduces the size of sensor data

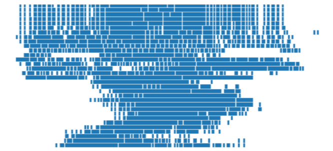
EVENT-DRIVEN AND STANDARD SENSORS S

Standard digital sensors

- Output floating-point or integer samples at each time instant
- Signal needs to be **encoded** into spikes in order to be processed by a Spiking Neural Network

Event-based sensors

- Output spike trains, remain silent when no input is present
- The spike encoding is performed within the sensor and the input can be fed directly to an SNN



DIGITAL SENSORS AND DATASETS S

- Two target datasets:
  - **Free Spoken Digit (FSD)** Dataset (audio signals)
  - **WISDM Smartphone and Smartwatch Activity and Biometrics** Dataset (wearable IMU data)
- Both time-varying signals; distinct regions in the frequency spectrum

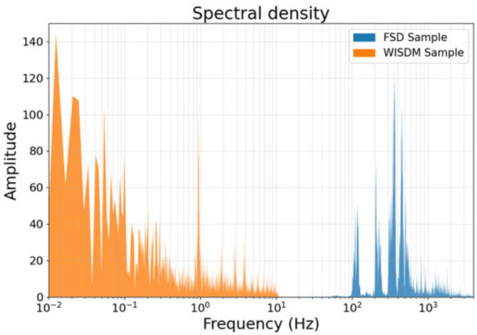


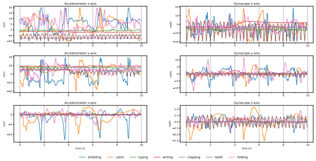
Figure: Forno et al., 2022

WISDM → very low frequency
FSD → middle frequency

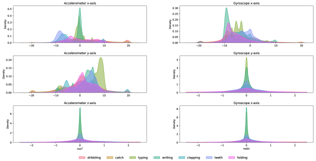
DIGITAL SENSORS AND DATASETS S

The WISDM dataset

- Smartphone + smartwatch IMU sensors for collecting datasets of human activity
- Signal segmentation affects accuracy and classification time



10 s smartwatch samples for *general hand-oriented* subset. Fra et al., 2022.



KDE of smartwatch values on the 6 IMU sensor axes. Fra et al., 2022.

The FSD dataset

- Crowd-sourced open database of spoken digits (variable quality
- `.wav` files at 8 kHz
- Word classification challenges
- Optional metadata for speaker identification

EVENT-BASED TACTILE SENSING S

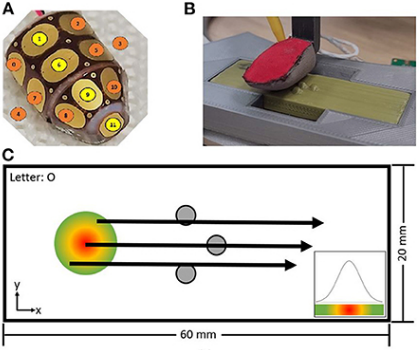


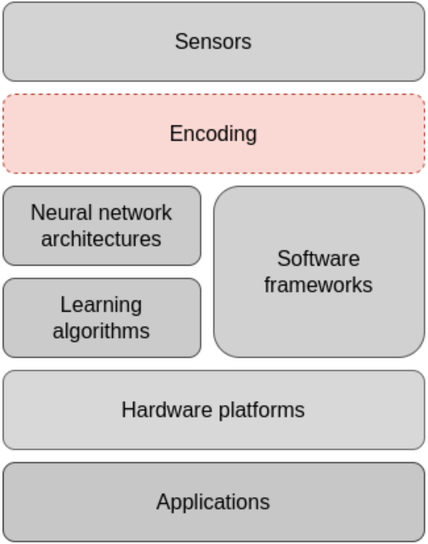
Figure: (A) Diagram of the sensorized fingertip. (B) Experiment configuration with a braille sample. (C) Random distribution for the starting location. Müller-Cleve et al., 2022.

The Braille dataset

- A novel dataset of haptic information based on the braille alphabet, recorded at IIT
- Digital capacitive sensor output encoded as spikes
- Sensorized fingertip (12 capacitive sensors) moving over 3D-printed braille letters (A-Z + space) with a constant sliding distance and velocity
- Starting location with random Gaussian noise

SENSORS: SUMMARY S

- SNN-based solutions require a sensing input in the form of spikes
- **Event-based sensors**, while promising, are still under active development
- **Digital sensors** are more accessible because of low cost and market dominance
- Interfacing a neuromorphic computational pipeline with digital inputs requires proper **encoding**



INPUT ENCODING AND PRE-PROCESSING

INPUT ENCODING AND PRE-PROCESSING E

- Many encoding schemes derived from the animal neuron
  - Rate-based coding
  - Temporal coding
- We will examine the effects of various spike encoding methods on the performance of a spiking CNN
  - Training by *transfer learning*
  - Classifying time-varying input signals from WISDM and FSD
- **Signal preprocessing** inspired by the human auditory system
  - Frequency decomposition
  - Feature extraction

SIGNAL PRE-PROCESSING E

Frequency decomposition

	FSD	WISDM
Butterworth	77.50 %	66.67 %
Gammatone	84.00 %	46.67 %

Table: FSD: 32 and 64 channels. WISDM: 4, 8, and 16 channels.

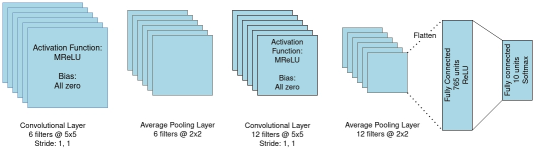
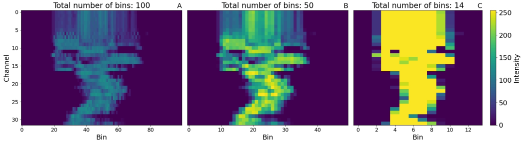
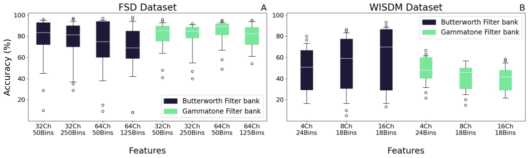


Figure: Architecture of the convolutional neural network.
All figures: Forno et al., 2022.



Feature extraction

- Sonogram → number of bins



ENCODING TECHNIQUES E

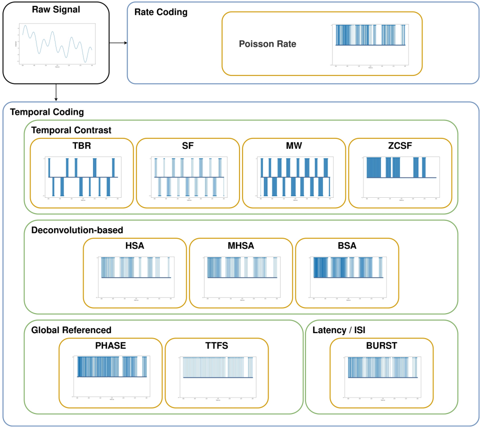


Figure: Forno et al., 2022.

Accuracy comparison

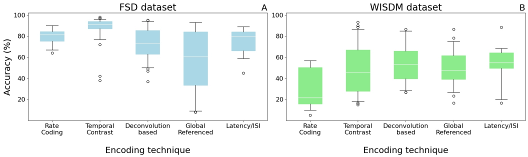


Figure: Median accuracy values of each encoding class. Forno et al., 2022.

Spike count

- *Deconvolution-based* → highest SC
- $\tau_{ref} > 0$ degrades performance

SUMMARY OF ENCODING STRATEGIES E

Encoding class and technique			Temporal data		Spatial data[1]
			Very low f	**Middle f**	
Rate coding	**Poisson Rate**		✗	✓	✓
Temporal Coding	**Temporal Contrast**	TBR	✓	✓	✗
		SF	✓	✓	✗
		MW	✓	✓	✗
		ZCSF	✓	✓	✗
	Deconvolution-based	HSA	–	–	✗
		MHSA	–	–	✗
		BSA	✓	–	✗
	Global Referenced	PHASE	✗	✓	✓
		TTFS	✗	✓	✓
	Latency/ISI	BURST	✗	✓	✓

EVENT-BASED ENCODING OF TACTILE DATA E

- Sigma-delta modulator

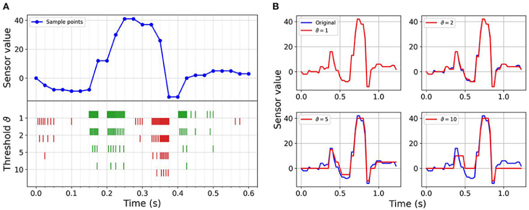


Figure: **(A)** Sensor reading sequence for a sample letter + sigma-delta modulated spikes. **(B)** Reconstructed sequences from events compared to the original signal. Müller-Cleve et al., 2022.

Signal reconstruction before time binning

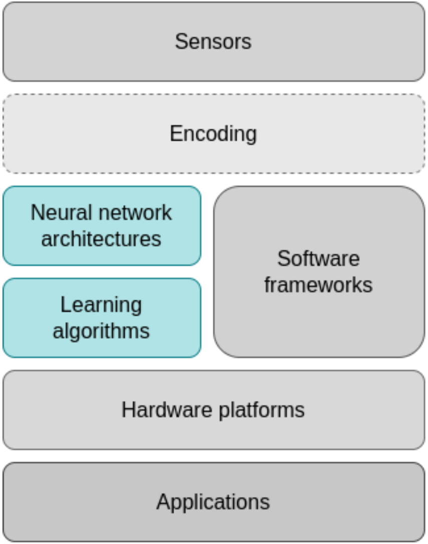
- Higher threshold $\Rightarrow$ fewer spikes $\Rightarrow$ worse reconstruction error
- The compression ratio γ grows faster than the reconstruction error ϵ

Signal reconstruction after time binning

- Higher threshold $\Rightarrow$ worse reconstruction error

INPUT ENCODING AND PRE-PROCESSING: SUMMARY E

- The frequency bandwidth of the input data has an impact on the quality of the encoding
- The *spike count* generated by an encoding technique influences the performance of the downstream SNN
  - Lower bound to communication sparsity
- Digital sensors coupled with software encoders can be useful for prototyping different coding techniques and tuning their interactions with other elements

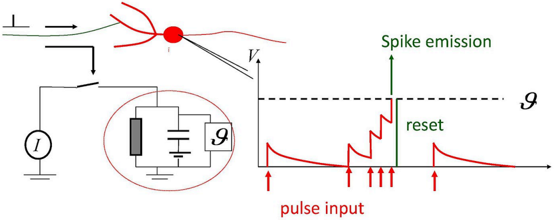


NEURAL MODELS

NEURAL MODELS M

Leaky Integrate and Fire (LIF)

- Optional refractory period τ_{ref}
- Reduces to the (m)ReLU activation function
- Variants such as ALIF add synaptic plasticity



Multi-compartmental neurons

- Model the behavior of dendrites and cell body separately
- Neuroscience simulations
- **Urbanczik-Senn model**: 2 compartments + 2 synapse types
- **Pyramidal model**: 3 compartments + 3 synapse types

LEARNING METHODS (M)

- Research still in progress for a reliable SNN learning algorithm
  - Global methods: Spike-Time-Dependent Plasticity, Back-Propagation Through Time
  - Local methods: Hebbian learning, E-prop
- ANN-to-SNN conversion provides an alternative method: **Transfer learning**
  - Training an ANN, then transfer the resulting weights to an SNN with the same topology
  - Maintain the ANN functionality while lowering power consumption thanks to spike transmission

SPIKING NEURAL NETWORK ARCHITECTURES M

- **Spiking CNN**
  - ANN-to-SNN conversion
- **Recurrent SNNs**
  - Legendre Memory Unit (LMU)

Comparison of equivalent spiking and non-spiking neural networks for HAR

- Accuracy; number of parameters; energy consumption

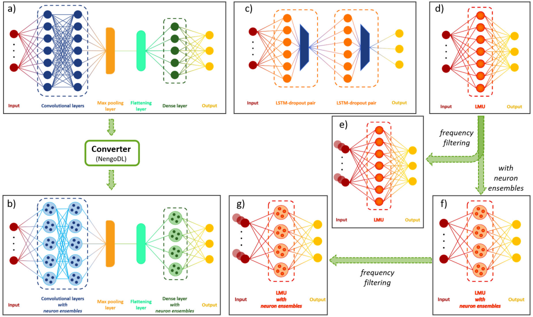


Figure: A comparison of convolutional and recurrent SNNs for Human Activity Recognition. Fra et al., 2022.

SPIKING NEURAL NETWORK ARCHITECTURES M

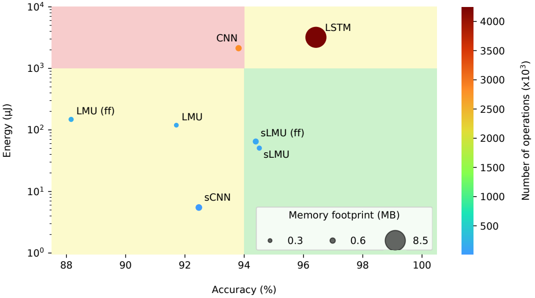


Figure: Fra et al., 2022.

- Energy estimated for Intel Movidius (for ANNs) and Intel Loihi (for SNNs)
- SNNs & LMUs use at least one order of magnitude less energy than ANNs
- CNNs and LSTMs are the largest in terms of memory / number of parameters

The Spiking LMU displays the best accuracy/energy/memory trade-off in this examination

MODEL COMPRESSION

M

- Reduce the size of the network prior to deployment
- **Model compression** with 2 phases:
  - **Synapse reduction**: delete connections with smallest weights
  - **Fine-tuning**: briefly re-train the remaining connections

Spiking CNN trained by transfer learning for FSD and WISDM classification

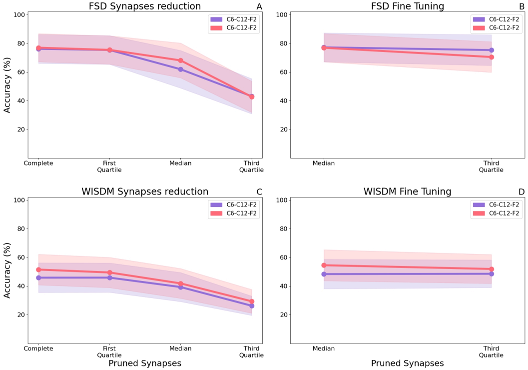
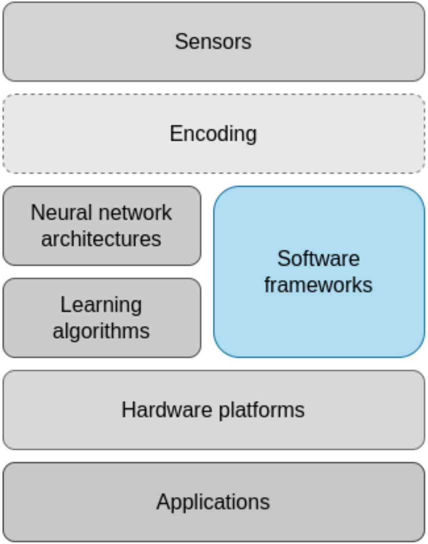


Figure: Median test accuracy after synapse reduction (A, C) and fine-tuning (B, D) for classification of the FSD and WISDM datasets. Forno et al., 2022.

NEURAL MODELS: SUMMARY (M)

- The dynamics of a network can be modeled down to single neurons
- This flexibility must be tailored to the target application
  - For most computation purposes, a simple LIF is sufficient
- The choice of the classifier network determines the efficiency and accuracy of the system
  - ANN-to-SNN conversion (i.e. sCNN): ease of implementation and training (transfer learning)
  - Recurrent SNNs (i.e. LMU): correlate time-varying events on a longer scale
- Model choices are informed by available **software and hardware**



SOFTWARE FRAMEWORKS

SOFTWARE FRAMEWORKS (P)

- Neuromorphic applications need a powerful software stack to enable research and development
- **SNN specification software** must handle the simulation complexity of SNNs and ease the difficulties in designing them. Some examples:

PyNN	Nengo	EONS
Python front-end for most neuromorphic simulators and hardware Widespread adoption by the research community	All-in-one simulator based on the Neural Engineering Framework NengoDL converter (DNN $\rightarrow$ event-based)	Evolutionary approach to network design Supports mapping to neuromorphic hardware platforms

SNN OPTIMIZATION SOFTWARE P

- Hyperparameter optimization (HPO) with the Neural Network Intelligence (NNI) toolbox

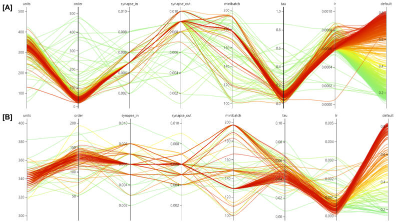


Figure: Optimization in 2 steps (A) and (B), restricting the search space. Fra et al., 2022.

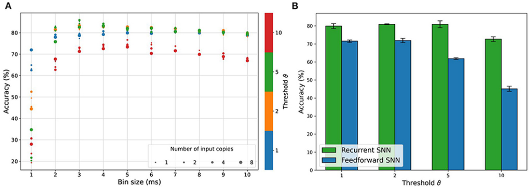
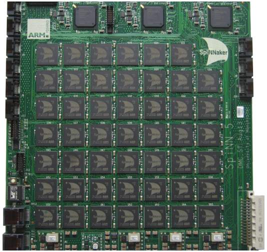


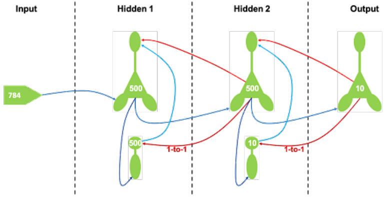
Figure: Application of HPO to feedforward and recurrent SNNs classifying Braille. **(A)** Best test accuracy results produced by the RSNN for all combinations of time_bin_size and nb_input_copies. **(B)** Mean and standard deviation of the FFSNN and RSNN accuracy results, with the best parameters for each encoding threshold. Müller-Cleve et al., 2022.

SYSTEM SOFTWARE: THE SPINNAKER EXAMPLE P

- Neuromorphic platforms require system software to compile and execute SNN applications on the hardware
- **SpiNNaker software stack:** an end-to-end toolchain based on PyNN
  - Host side: Python libraries translate PyNN models into SpiNNaker applications
  - Platform side: an event-driven OS interfaces user applications with the underlying hardware
- Efficient mapping and routing is an important step in the toolchain
  - Placement and routing exploration for implementation of new features

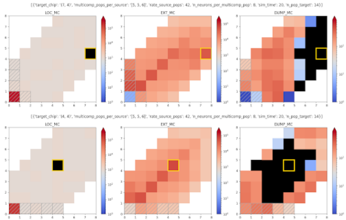


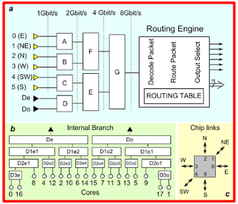
PLACEMENT AND ROUTING EXPLORATION ON SPINNAKER P



2-layer MNIST classifier with *rate-based* multi-compartmental neurons

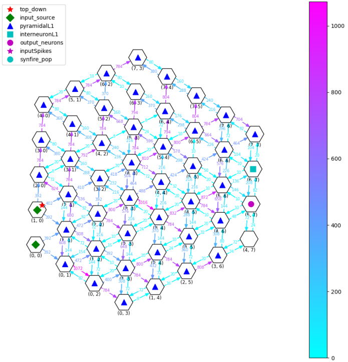
- Tight timing constraints lead to catastrophic packet loss

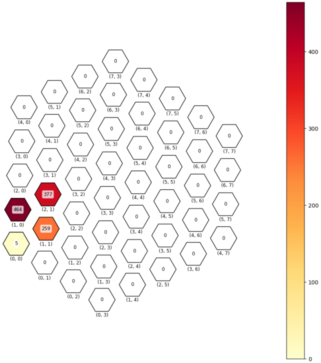




- Communication test with a single neuron: lost packets depend on target placement
- Likely due to multiplexer imbalance at the router's entrance

PLACEMENT AND ROUTING EXPLORATION ON SPINNAKER P



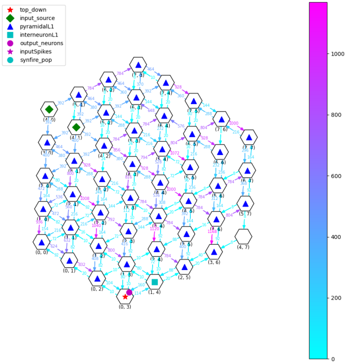


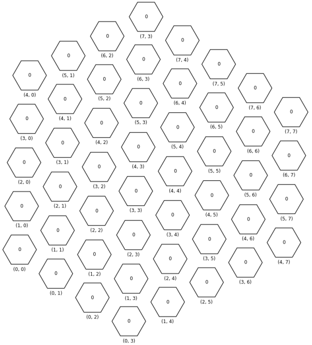
Custom placement algorithm

Default placement

1105 packets lost

PLACEMENT AND ROUTING EXPLORATION ON SPINNAKER P





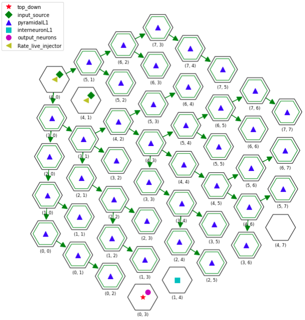
Custom placement algorithm

Custom placement

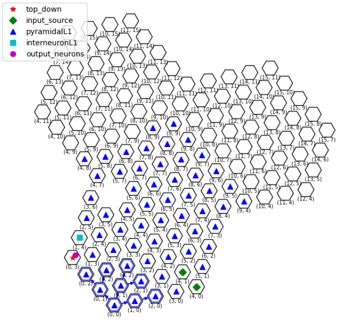
0 packets lost

PLACEMENT AND ROUTING EXPLORATION ON SPINNAKER P

Custom routing algorithm

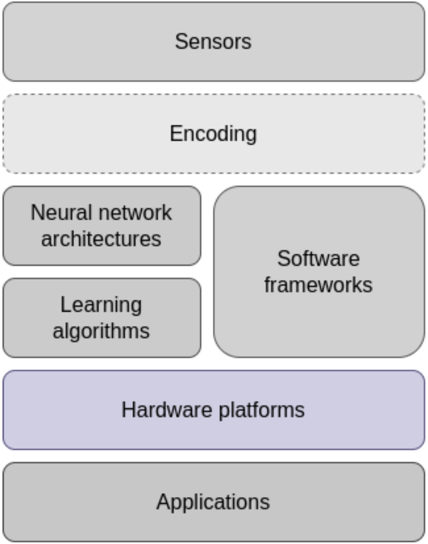


Multi-board place & route



SOFTWARE FRAMEWORKS: SUMMARY

- **SNN specification software**
  - Create simple, abstract and portable descriptions of the desired models
  - Compile and deploy on simulated or physical hardware backends
- **ANN methods** (i.e., HPO) can also be used with SNNs to refine their architecture and improve results
- Flexible **system software** is an important middleman between the specification and deployment software and the neuromorphic hardware
  - SpiNNaker's system software stack can interact with PyNN libraries to implement new features and enable continuous development



HARDWARE PLATFORMS

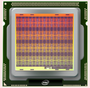
HARDWARE PLATFORMS H

- Acceleration of Spiking Neural Networks

BrainScaleS

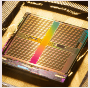
Simulates biological neurons at faster than real-time speed
Mixed analog/digital

Loihi



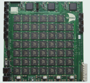
Neuromorphic research chip for the simulation of asynchronous SNNs
Mixed analog/digital

Dynap-SEL



Edge computing applications for IoT and Industry 4.0
Mixed analog/digital

SpiNNaker



Real-time SNN simulation on ARM general-purpose processors
Fully digital

EXPLORING THE SPINNAKER COMMUNICATION INFRASTRUCTURE WITH MPI (H)

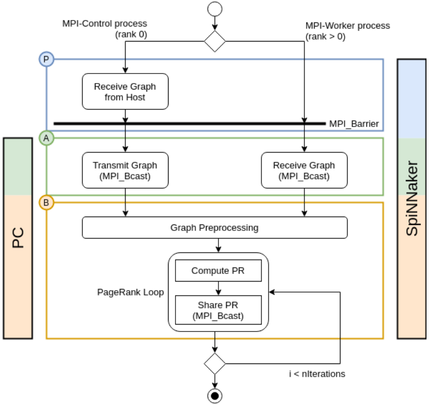


Figure: Forno et al., 2021

- **SpiNMPI: the MPI library for SpiNNaker**
  - Study the interconnection scheme using distributed-memory parallel algorithms
- Benchmark program for SpinMPI: **MPI-PageRank**
- Comparison with existing SNN implementation of PageRank on SpiNNaker

SPINMPI PERFORMANCE ANALYSIS

H

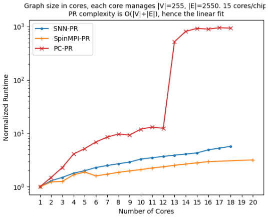
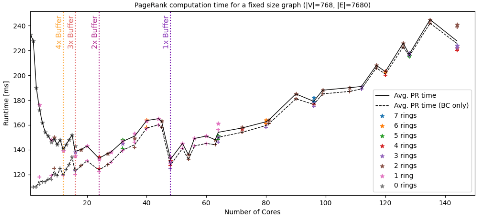


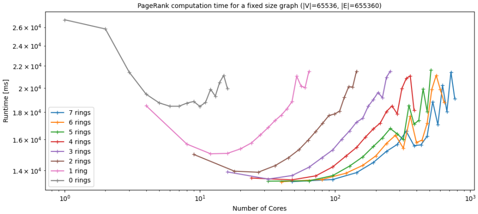
Figure: Forno et al., 2021.

- MPI-PR scales around 1.75× faster than SNN-PR on 15 cores
- Uneven trend due to DTCM/RAM memory location



Minimizing write/read access to the MPI communication buffer decreases the broadcast time. Vertical lines highlight points where the number of required buffer accesses changes.

Figure: Forno et al., 2021.

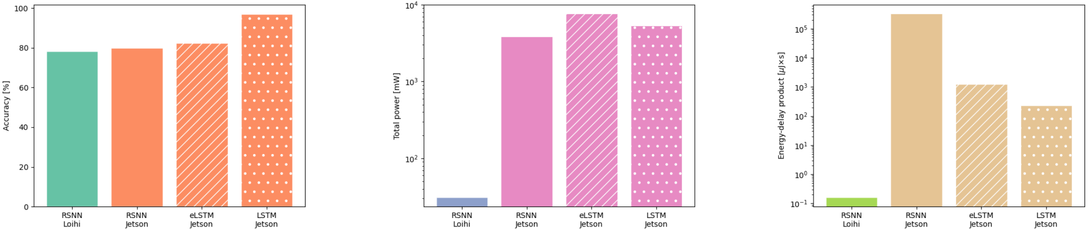


Different placements of the same number of cores affect execution time.

Figure: Forno et al., 2021.

BRAILLE CLASSIFICATION ON LOIHI VS. GPU

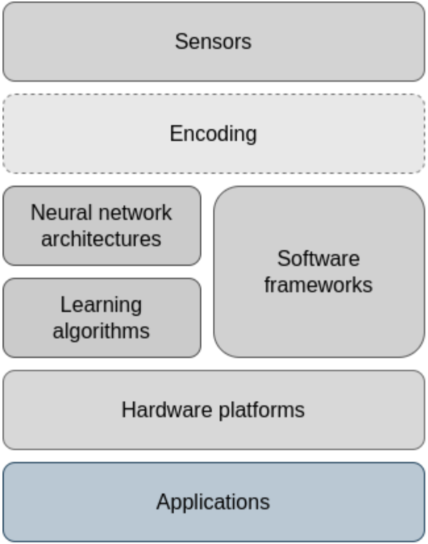
H



- Accuracy: RSNN + Loihi underperforms by 17% compared to the best-performing non-spiking classifier (LSTM + GPU)
  - Comparable results to RSNN and eLSTM on GPU
- Energy-delay product for LSTM + GPU is 1475× greater than RSNN + Loihi

HARDWARE PLATFORMS: SUMMARY H

- SpiNNaker and Loihi as representatives of the 1^{st} generation of neuromorphic hardware
- SpinMPI + massively parallel non-spiking algorithm reveals bottlenecks in the SpiNNaker communication architecture
  - Complex interactions limit scalability
- RSNN operating on Loihi offers a significant energy/accuracy tradeoff
  - Accuracy underperforms by 17% compared to LSTM on GPU
  - ... But 3-orders-of-magnitude gains in energy efficiency



BRINGING IT ALL TOGETHER: TOWARDS A COMPLETE NEUROMORPHIC PIPELINE

BRINGING IT ALL TOGETHER: TOWARDS A COMPLETE NEUROMORPHIC PIPELINE (A)

- Existing neuromorphic systems cannot be stationed as standalone edge devices
  - Separate host uploads the network setup and input data
- Some neuromorphic designs incorporate von Neumann-based coprocessors
  - NeuroEdge: NM500 neuromorphic processor in a Raspberry Pi
  - Loihi: microcontroller-class x86 chips at the mesh’s periphery perform data encoding/decoding
- Issues in interfacing neuromorphic sensors, encoders, models, software tools and hardware with each other and with traditional computing frameworks

CONFIGURING AN EMBEDDED NEUROMORPHIC COPROCESSOR WITH RISC-V (A)

- ODIN: a Spiking Neural Network coprocessor
- Integration with a low-power RISC-V microprocessor (Rocket Chip) via SPI
  - Chipyard framework for SoC
  - ODIN as a Memory-Mapped I/O (MMIO) peripheral
- Whole system synthesized for PYNQ Z2 board

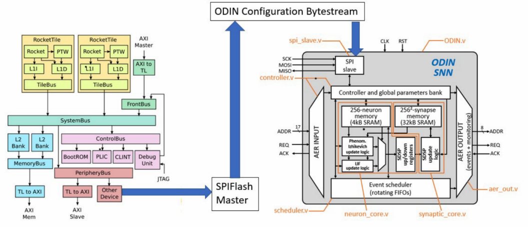


Figure: Forno et al., 2021.

RISC-V coprocessor to set up the SPI controller, initialize ODIN, and gather results through its AER output interface

FROM SENSOR TO NEURON A

A neuromorphic approach for on-edge HAR applications

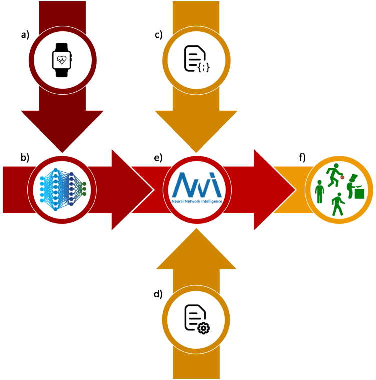


Figure: Fra et al., 2022.

- WISDM smartphone and wristwatch activity and biometrics dataset
- Raw data-only classification
- Nengo as basic framework

Preliminary processes:
(a) Dataset selection
(c) HPO search space specification
(d) HPO experiment configuration

Main pipeline:
(b) Neural network architecture selection
(e) Hyperparameter optimization
(f) Classification

FROM SENSOR TO NEURON A

A time-varying signal benchmark for spike encoding techniques

- Pipeline expanded to support:
  - Encoding and preprocessing filters
  - Transfer learning (CNN → sCNN)
  - Model compression stage
- Validation of encoding methods by applying the same pipeline to two distinct datasets (FSD and WISDM)

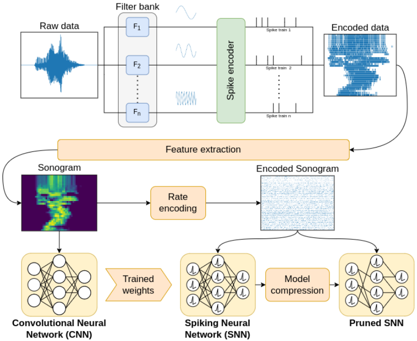


Figure: Forno et al., 2022.

FROM SENSOR TO NEURON (A)

Braille letter reading benchmark on neuromorphic hardware

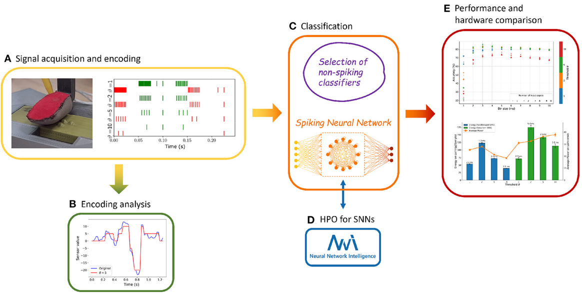


Figure: Müller-Cleve et al., 2022.

- Time series classification on neuromorphic hardware
  - Spike coding
  - Asynchronous event-driven computation
- Performance analysis: multiple metrics and hardware solutions
- The resulting pipeline can be applied to a wide range of time-dependent data

NEUROMORPHIC PIPELINE: SUMMARY (A)

- Workable prototypes for benchmarking neuromorphic system designs
- Complete end-to-end neuromorphic pipelines ready for deployment in IoT and industrial applications are not yet easily realizable
  - Scarce availability of event-based sensors
  - Lack of resources for native SNN training
  - Gradual adoption of new tools, techniques and models
- The developed method creates a valuable and stable platform to support future work

CONCLUSIONS

- Neuromorphic pipeline with special attention to applications involving the categorization of IoT time-varying data
  - An ideal use case for SNNs (accurate internal representation of spatio-temporal dynamics)
- An embedded neuromorphic application must interact with its environment via **sensors**
  - **Event-based sensors** for specialized tasks bring extreme power efficiency
  - **Digital sensors**: more accessible, low cost ⇒ likely to remain relevant in neuromorphic application for the foreseeable future

CONCLUSIONS

- Issue of **spike encoding** and its repercussions on downstream elements
- A **sufficiently high spike count** is required to properly stimulate all the cascading layers of the classification network
  - Tradeoff between information preservation and energy reduction
- **Rate-based coding** does not properly represent the fine temporal dynamics of an input signal
- **Temporal coding** more fitting for SNNs
  - *Temporal contrast* techniques strike the best balance between accuracy and ease of implementation

CONCLUSIONS

- Among classifier architectures, **spiking CNNs** offer ease of implementation and transfer learning
- Comparative classification of the Braille dataset shows the most suitable architecture for time-varying data in the spiking domain is the **recurrent neural network**
  - Memory trace of past events in recurrently connected reservoirs
  - Correlate time-varying events on a longer scale
- Optimization techniques remain important in addition to architectural design

CONCLUSIONS

- **Neuromorphic hardware** prepares to enter its 2nd generation
  - Communication bottlenecks identified in the SpiNNaker architecture should improve with the latest hardware iteration
- We have showed that communication efficiency also heavily depends on the placement and routing algorithms
  - Need to account for the relative physical location of data and computing elements
  - Fully programmable systems like SpiNNaker enable continual evolution
- Finally, we demonstrated interoperability of neuromorphic and general-purpose processors and gradually built **a neuromorphic pipeline** for classification of time-varying signals

CONCLUSIONS

- For many years, neuromorphic technology has suffered from a lack of accessibility
  - Scarcity of standard APIs to ensure HW/SW interoperability
  - No unified front-end to combine different solutions
- Community efforts have brought new tools to explore novel neural architectures and applications with increased modularity
  - Neural Engineering Framework (NEF) and Nengo
  - Intel Neuromorphic Research Community
- This dissertation represents a first step toward effortless integration of neuromorphic devices into fully embedded applications

The use cases offered here offer a foundation for future research to build upon and expand the possibilities of neuromorphic engineering.

SOURCES

- Vittorio Fra, Evelina Forno, Riccardo Pignari, Terrence C. Stewart, Enrico Macii, and Gianvito Urgese. Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications. *Neuromorphic Computing and Engineering*, 2(1):014006, 2022. This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/4.0/`.
- Evelina Forno, Vittorio Fra, Riccardo Pignari, Enrico Macii, and Gianvito Urgese. Spike encoding techniques for IoT time-varying signals benchmarked on a neuromorphic classification task. *Frontiers in Neuroscience*, 16, 2022. This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/4.0/`.
- Simon F. Müller-Cleve, Vittorio Fra, Lyes Khacef, Alejandro Pequeño-Zurro, Daniel Klepatsch, Evelina Forno, Diego G. Ivanovich, Shavika Rastogi, Gianvito Urgese, Friedemann Zenke, and Chiara Bartolozzi. Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware. *Frontiers in Neuroscience*, 16, 2022. This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/4.0/`.
- Evelina Forno, Alessandro Salvato, Enrico Macii, and Gianvito Urgese. Pagerank implemented with the MPI paradigm running on a many-core neuromorphic platform. *Journal of Low Power Electronics and Applications*, 11(2):25, 2021. This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/4.0/`.
- Evelina Forno, Andrea Spitale, Enrico Macii, and Gianvito Urgese. Configuring an embedded neuromorphic coprocessor using a RISC-V chip for enabling edge computing applications. In *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 328–332. IEEE, 2021. © 2021 IEEE.
- Slide 17: LIF model illustration by *Spiking16* on Wikimedia Commons

THANK YOU FOR YOUR ATTENTION

SUMMARY OF CONTRIBUTIONS

Article	Title	Contribution
Fra et al., 2022	Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications	Network definition, hyperparameter optimization, article writing
Forno et al., 2022	Spike encoding techniques for IoT time-varying signals benchmarked on a neuromorphic classification task	Designed the analysis and wrote the manuscript
Muller-Cleve et al., 2022	Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware	Performed NNI implementation and wrote the manuscript
Forno et al., 2021	Pagerank implemented with the MPI paradigm running on a many-core neuromorphic platform	Conceptualization, methodology, software, validation, visualization, writing
Forno et al., 2021	Configuring an embedded neuromorphic coprocessor using a RISC-V chip for enabling edge computing applications	Supervision and article writing