SP9 Neuromorphic Computing Platform - Results from SGA2 Year 2 (D9.6.2 - SGA2)

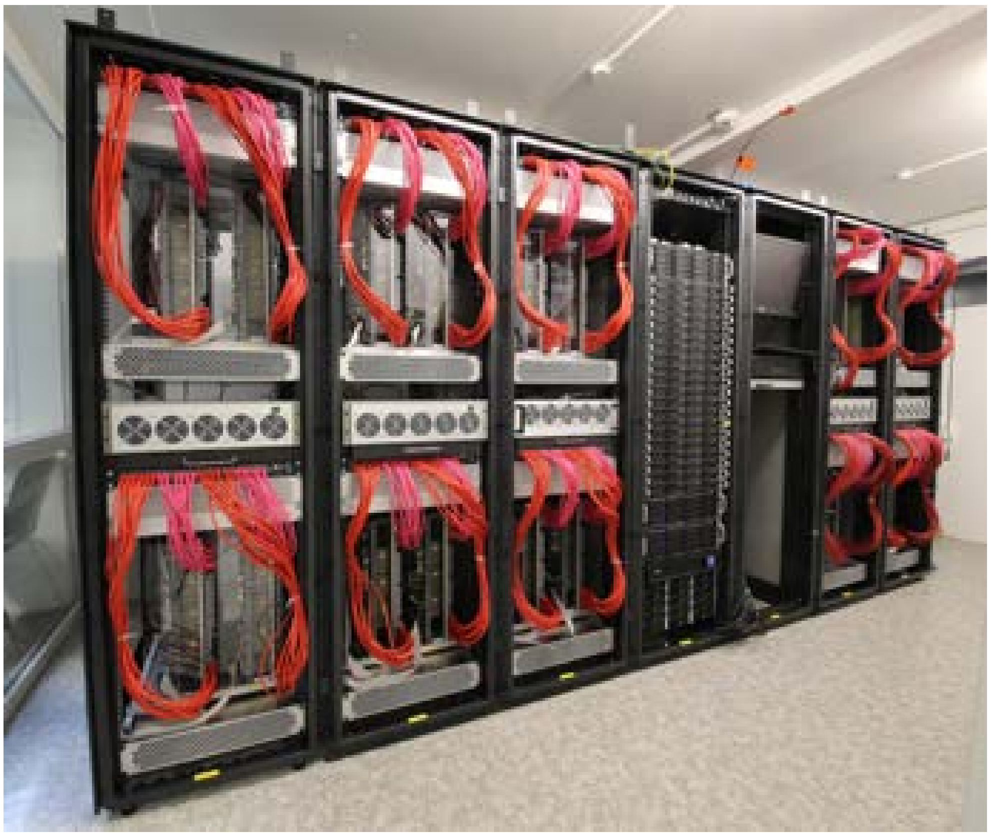
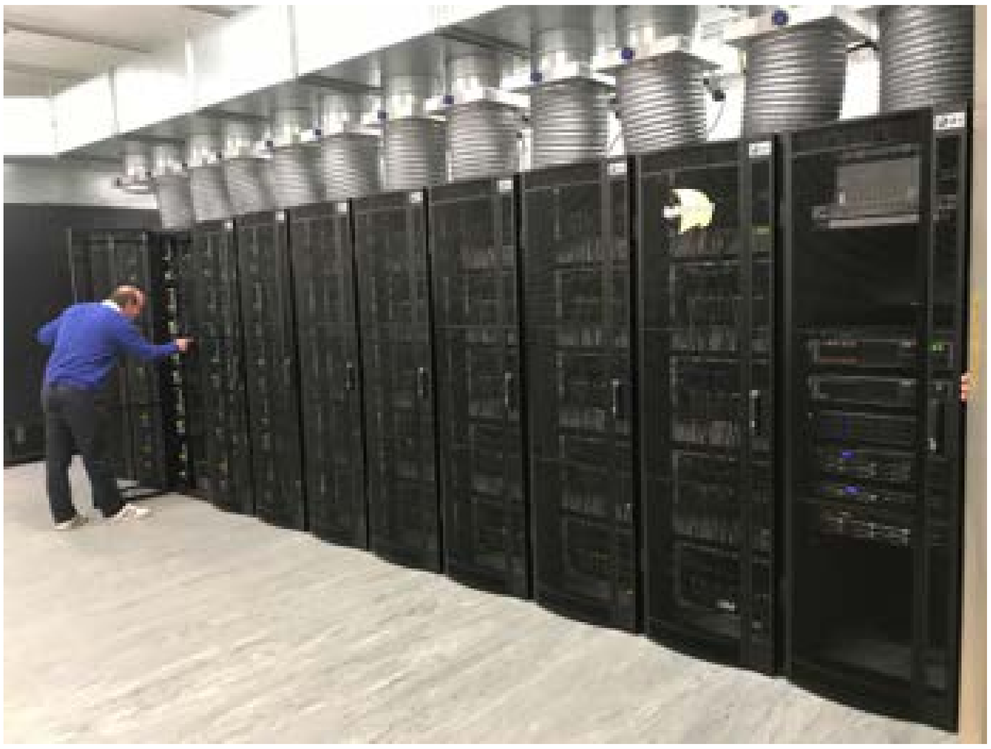



Figure 1: The BrainScaleS (upper image) and SpiNNaker (lower image) machines.

These machines form the HBP Neuromorphic Computing Platform, offering accelerated learning and programmable flexibility respectively.

Project Number:	785907	Project Title:	Human Brain Project SGA2

Document Title:	SP9 Neuromorphic Computing Platform - Results from SGA2 year 2
Document Filename:	D9.6.2 (D62.2 D47) SGA2 M24 ACCEPTED 200731.docx
Deliverable Number:	SGA2 D9.6.2 (D62.2, D47)
Deliverable Type:	Report
Work Packages:	WP9.1 - WP 9.6
Key Result(s):	KR9.1 - KR9.6, KR9.x
Dissemination Level:	PU = Public
Planned Delivery Date:	SGA2 M24 / 31 Mar 2020
Actual Delivery Date:	SGA2 M25 / 06 Apr 2020; Accepted 31 Jul 2020
Author(s):	SP9 members
Compiled by:	Steve FURBER, UMAN (P63)
Contributor(s):	SP9 members
SciTechCoord Review:	Mehdi SNENE, EPFL (P1)
Editorial Review:	Guy WILLIS, EPFL (P1)
Description in GA:	The summary of SP 9 results for project months M13-24, broken down by task. WPs involved: WP9.1, WP9.2, WP9.3, WP9.4, WP9.5 and WP9.6 In order to report fully, the Deliverable may need to include information that the SP does not wish to make public at the time of the report. This material will be included in the Deliverable for the EC and Reviewers, but not in the version published publicly.
Abstract:	This Deliverable describes progress over the second 12 months of SGA2 (April 2019 – March 2020) in the development of the Neuromorphic Computing Platform.
Keywords:	Neuromorphic computing, SpiNNaker, SpiNNaker-2, BrainScaleS, BrainScaleS-2, Theory
Target Users/Readers:	Scientists interested in neuromorphic computing

Table of Contents

Table of Tables

Table of Figures

1. Overview

The SP9 Neuromorphic Computing Platform is part of the Human Brain Project's EBRAINS research infrastructure, offering open access to two of the world's leading neuromorphic (brain-inspired) computing systems. SpiNNaker (Spiking Neural Network Architecture) is a many-core digital computer incorporating a million ARM processor cores, and is the world's largest neuromorphic computing system. BrainScaleS employs physical emulation, wherein analogue electronic circuits model the equations of the biological components directly, and is the world's fastest neuromorphic computing system. Together, these two systems form the EBRAINS Neuromorphic Computing Platform and, as such, are the world's only openly accessible neuromorphic computing resources.

Significant advances over the last 12 months of SGA2 included the demonstration on SpiNNaker of the first robust, real-time implementation of the cortical microcircuit, paving the way for a real-time multi-area cortical model in SGA3, published in the Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, in December 2019. The development of a range of novel learning algorithms on the BrainScaleS-2 prototype chip are also noteworthy. SP9 has, with CDP5, contributed to the development of novel theories of computational principles, including learning-to-learn (L2L) and learning based upon dendritic computation, both of which are broadly applicable to neuromorphic systems. A gradient descent algorithm, e-prop, promises to bring the capabilities of backpropagation in artificial neural networks to spiking neural networks in a biologically-plausible mechanism, opening up the prospect of the wider applicability of spiking networks to industrially-relevant applications. All of these novel learning algorithms will be developed and applied to the Neuromorphic Computing Platform during SGA3.

2. Introduction

During the two years of SGA2, the field of neuromorphic computing received increased attention; in particular, with the growing presence of the Intel Loihi chip across the academic research community. Loihi is not a full product, so companies cannot build product plans around it; rather, it is a "research prototype", available to the academic community, which Intel is using to explore the future potential of such a product.

Over the same period, there has been a growing expectation that neuromorphic technology has something to offer in the commercial AI domain, perhaps by reducing the very high power demands of conventional artificial neural network (ANN) solutions. Event-based AI, or bio-AI, has intrinsic properties that, when correctly realised, offer the prospect of significant power reduction, though there are still challenges in delivering this advantage. Training ANNs is now highly developed through frameworks such as Tensorflow and Keras, so one approach is to train an ANN and then convert it into a spiking neural network (SNN) for the inference phase. However, the simplest conversion approach, which is to use rate-based encoding, where the firing rate of an SNN neuron corresponds to the activation output of the ANN neuron, to replace a digital multiplication with multiplication by repeated addition, and confers no fundamental advantage. To show an advantage, the SNN must use an encoding other than rate-based, and some interesting alternatives are just beginning to emerge from work carried out in SP9. Key Result KR9.6 reports on applications developed after M12 that exploit the 2nd-generation features of the hardware platforms. These applications include tools for mapping deep networks on SpiNNaker-2, hyper-parameter optimisation, structural plasticity, Bayesian sampling and others that are relevant for potential applications of HBP hardware in Artificial Intelligence.

Of course, the major focus of SP9 is on developing neuromorphic computing systems that contribute to brain science through the EBRAINS infrastructure. Here there has been solid progress over the last 12 months of SGA2 in improving the stability and capabilities of the supporting software stacks, and this has been demonstrated in the speeding up the cortical microcircuit model on SpiNNaker, from a 20x slow-down when reported in May 2018 to real-time in December 2019. Parallel advances on BrainScaleS have demonstrated a suite of novel learning algorithms.

3. Key Result KR9.1: Both first-generation machines integrated into Joint Platform

3.1 Outputs

3.1.1 *Overview of Outputs*

3.1.1.1 List of Outputs contributing to this KR

- Output 1: Neuromorphic Computing Platform Remote Access Service (C3042)
- Output 2: SpiNNaker Neuromorphic Computing System (C2)
- Output 3: BrainScaleS-1 (C1)

3.1.1.2 How Outputs relate to each other and the Key Result

Output 1 is used to submit jobs to and extract results from Output 2 and Output 3. Outputs 2 and 3 are the first generation machines (namely SpiNNaker and BrainScaleS respectively) which are integrated into the Joint Platform as part of this KR.

3.1.2 *Output 1*

The Neuromorphic Computing Platform Remote Access Service (C3042) enables users to run simulations/emulations on the BrainScaleS, SpiNNaker and Spikey systems by submitting jobs to a central queue, and then retrieving the results once the simulation is complete. The core of the service is a web service with a REST API (C344), with authentication using the HBP identity service. Users can submit jobs using a Collaboratory app (Job Manager app; C343), through a Python client (C345), or on the command line (part of C345). The service also has a number of other components, such as a job statistics dashboard (C371) and a resource/quota management service (C369, C370).

In the last year, the Job Manager app has had a number of minor improvements, but the major advancements have been in the back end. We have deployed a service for load-balancing and failover (C1638), to increase the availability of the REST API and Collaboratory apps. These services are currently running on two separate machines, using the ICEI VM infrastructure at CSCS. A load-balancing service routes requests alternately to the two servers, and detects if one of the servers is not responding (in which case, a monitoring service alerts the platform maintainers). We have also developed Knowledge Graph (KG) schemas, based on the W3C PROV standard, for neuromorphic simulations, models, and results (C377), and periodically synchronise the job database with the KG, so that neuromorphic workflows can be fully integrated with other tools available on the HBP/EBRAINS infrastructure. We are in the process of rewriting the REST API to communicate directly with the KG, avoiding the need for the periodic synchronisation.

Progress vs State of the Art & Recent Developments

The criterion for “beyond state of the art” for SGA2 is “Worldwide unique integration of neuromorphic computing machines into the HBP-JP”. We consider that this criterion has been satisfied.

Component	Link to	URL
C3042 - SP9 Neuromorphic Computing Platform	Software Repository	https://github.com/HumanBrainProject/hbp_neuromorphic_platform https://github.com/HumanBrainProject/hbp-neuromorphic-client
	Technical Documentation	https://www.hbpneuromorphic.eu/developer_guide/index.html

Component	Link to	URL
Remote Access Service	User Documentation	https://collab.humanbrainproject.eu/#/collab/51 (community account required. Request at: neuromorphic@humanbrainproject.eu) https://electronicvisions.github.io/hbp-sp9-guidebook/

3.1.3 *Output 2: SpiNNaker related*

The SpiNNaker machine continues to be accessible from the Joint Platform via Output 1. The service has been updated to reduce the turnaround time when accessing jobs via the Joint Platform, by pre-cloning the git modules required to operate the machine. These are then updated if required and any branch or tag changes are then made before the job is run.

The SpiNNaker machine has also been made available through Jupyter notebooks, which are authenticated using the HBP authentication service. This allows the users to make use of the SpiNNaker machine in a more interactive way. The SpiNNaker-Neurorobotics Platform (NRP) interface has also been deployed through these Jupyter notebooks, so users can start their own NRP instance on a host machine adjacent to the SpiNNaker machine.

Component	Link to	URL
C2 - SP9 SpiNNaker Neuromorphic Computing System	Software Repository	https://github.com/SpiNNakerManchester/RemoteSpiNNaker
	Technical Documentation	Integrated into code
	User Documentation	https://collab.humanbrainproject.eu/#/collab/51 (community account required. Request at: neuromorphic@humanbrainproject.eu) https://electronicvisions.github.io/hbp-sp9-guidebook/

3.1.4 *Output 3: BrainScaleS-1 related*

The BrainScaleS-1 has been accessible from the Joint Platform via Output 1. During the reporting period, its hardware and software infrastructure was completed and enhanced in several aspects, as described in the following paragraphs.

Component	Link to	URL
C1 - SP9 BrainScaleS-1 Neuromorphic Computing System	Software Repositories	https://github.com/electronicvisions/
	Technical Documentation	Integrated into code and internal specification
	User Documentation	https://electronicvisions.github.io/hbp-sp9-guidebook/

3.1.4.1 Migration Raspberry Pi3/4

Updated BrainScaleS-1 systems are equipped with a newer version of Raspberry Pis, either with version 3B+ or 4. An upgrade-kit was developed for the existing systems. The software stack is updated to the newer version. The new Raspberry Pis have several advantages over the former ones. Now the Raspberry Pis boot over PXE and therefore don't need sd-cards anymore. Maintenance of the Raspbian Linux distribution used now is also easier and the BrainScaleS-1 monitor- and control-software is integrated in the continuous integration tool-chain. The Raspbian linux distribution has been upgraded too. The old one-core processor of the Raspberry Pi 1B was working at its limit. The new Raspberry Pis have additional CPU cores, which are useful for extensive monitoring features closer to the BrainScaleS-1 modules.

3.1.4.2 Analysis of cooling concept of BrainScaleS-1

The cooling concept of the BrainScaleS-1 modules was analysed with regard to noise reduction, efficiency and power consumption. All of these aspects were improved by optimising air guidance through the module and exchanging selected fans, while keeping cooling performance equal. As a result, some of the fans could be slowed down to 50% of their full speed, which also reduced the noise level. The power consumption of all fans could be decreased from around 180W to 70W.

Additionally, a wafer temperature regulation system was developed. This pid regulator controls the fan speed to keep the wafer temperature constant at 48°C.

3.1.4.3 Firmware Improvements for power supply boards

The power supply boards of the BrainScaleS-1 module received a new firmware. Calibration of sensor data on the boards returns a more accurate state of the system, in terms of current consumption and operating point of the analogue circuits. Furthermore, the stability increased as new protection routines were installed and this also reduced the number of repairs required. For example, a current shunt on one board would burn under certain conditions; this situation can now be detected in advance and handled. As a result, no repair has been required since the firmware update.

3.1.4.4 Testing and integration of new wafer set

Wafer modules using wafers of the new HICANN v4.1 wafer set uncovered errors regarding communication with some HICANNs. To ensure that these errors were not introduced by the assembly procedure, a method for testing the wafers before the assembly had to be established. Therefore, an automated test suite for a wafer prober was developed and was used to test all components of the wafers of the new HICANN v4.1 wafer set. The tests that were integrated into the automated test suite covered High-Speed connection, slow control connection and power tests. Each type showed errors or irregularities at different HICANNs on each wafer in a low percentage range. These errors do not hinder wafer usage, but they do reduce the amount of resources that each wafer can provide.

The assembly process has started, using the wafers with the lowest error rates. Four wafers have been assembled and integrated into the BrainScaleS-1 system and this process is still ongoing.

3.1.4.5 Integration of new Analogue-to-Digital (ADC) subsystem into software framework

The communication layer for the ADC subsystem has been developed. It allows for configuration of the ADC front-ends, trigger sources and recording lengths for all channels. Additionally, aggregated and compressed data can be retrieved from SDRAM and are re-assembled into separate traces. The hardware abstraction layers required for integration into the current BSS-1 software stack are in development.

3.2 Validation and Impact

3.2.1 Actual and Potential Use of Output(s)

Output 1 (Neuromorphic Computing Platform Remote Access Service):

The service is in use and provides an easy, installation free “first contact” with both NMC systems, directly integrated in the HBP Joint Platform Collaboratory. The service can also easily be used for training.

Output 2 SpiNNaker:

Users continue to use the SpiNNaker machine via the Joint Platform, with over 8,000 jobs in total completed to date (up from around 5,000 at the start of April 2019), submitted by 88 users (up from 69 at the start of April 2019). The SpiNNaker machine provides the HBP Research Infrastructure with a computing system capable of performing large-scale brain simulations, executed in real-time, allowing users to explore the performance of large-scale networks executing for longer durations. The Jupyter service has executed 14,660 runs, generated by 161 users, on the SpiNNaker machine since the service was started. Both the batch service and the Jupyter notebook service have been validated through continued interaction with users of the service, via the SpiNNaker mailing list and the HBP Support service.

Output 3 BrainScaleS-1:

The BrainScaleS-1 platform has reached a state where it can demonstrate the feasibility of wafer-scale integration for neuromorphic computing, based on accelerated analogue physical models. In total, almost 400,000 jobs have been completed (c. 90,000 jobs since April 2019) by more than 70 users (40 users in the last 90 days). These numbers include jobs submitted via the collab interface, Jupyter, and the local batch submission system.

As soon as a wafer-scale version of the BrainScaleS-2 ASIC is available, the BrainScaleS platform will be capable of emulating the dynamics of learning and development of large networks of structured neurons in a time-continuous model, with an unprecedented energy efficiency.

The contributions of the individual tasks to the output are as follows:

The new Raspberry Pis improve the software maintainability and extend the possible monitoring and alerting options for the BrainScaleS-1 modules.

The cooling optimisations reduce the power consumption of the cooling system, which saves energy and costs, and expands the lifetime of components, because fans don't always run at full speed now.

The new power supply firmware reduced the number of repair actions.

The integration of the new HICANN v4.1 wafer set into the system increases the number of accessible chips with improved analogue neuromorphic circuitry.

The new ADC subsystem improves the readout capability and thus will allow faster and more accurate HICANN calibration results.

4. Key Result KR9.2: Comprehensive software suite for the operation of neuromorphic machines

4.1 Outputs

4.1.1 Overview of Outputs

4.1.1.1 List of Outputs contributing to this KR

- Output 1: Multicompartmental models in PyNN (C349)
- Output 2: SpiNNaker Neuromorphic Computing System (C2)
- Output 3: App for machine-learning with neuromorphic hardware (C1656 & C1644)
- Output 4: MPI-SpiNNaker (C3045)
- Output 5: Neuromorphic Benchmarks (C2735)
- Output 6: MUSIC library (C347)
- Output 7: Hardware integration (C1810)

- Output 8: CSA library (C1769)
- Output 9: BrainScaleS-1 Software
- Output 10: BrainScaleS-2 Software

4.1.1.2 How Outputs relate to each other and the Key Result

Output 2 provides the software for executing PyNN models on SpiNNaker. Currently, this has not been updated to work with Output 1.

Output 3 relies on Output 2 as one of its target platforms. New concepts introduced by Output 1 have been considered in the design, but cannot yet be deployed.

Output 5 relies on Output 2 and Output 9, as these are targeted by the benchmark suite. Improvements to those Outputs may improve benchmark results, and benchmarking may lead to improvements in the respective software stacks of the platforms.

4.1.2 *Output 1: Multi-compartmental models in PyNN*

The next-generation BrainScaleS and SpiNNaker chips will allow multi-compartment neuron models, with non-linear mechanisms such as calcium dynamics. To support such models, work on extensions to PyNN began in SGA1 and continued in the first year of SGA2. In the second year of SGA2, we continued this work, adding support for the specific calcium and NMDA models that will be available in the BrainScaleS 2 chip (see Schemmel *et al*., 2017; https://arxiv.org/abs/1703.07286).

Component	Link to	URL
C349 - PyNN	Software Repository	https://github.com/NeuralEnsemble/PyNN/tree/mc
	Technical Documentation	https://neuralensemble.org/docs/PyNN/2.0/developers_guide.html
	User Documentation	https://neuralensemble.org/docs/PyNN/2.0/

4.1.3 *Output 2: SpiNNaker Neuromorphic Computing System*

The SpiNNaker software continues to be improved. A prototype has been written which allows the execution of the 1mm^2 cortical microcircuit to be executed in real time on the platform. Additionally, this network has been executed for 12 hours without any faults. We believe that this is the first time that this network has been run, both in real time and for such a duration, on any hardware, including HPC systems.

In addition, the software has been updated to run jobs that use even larger shares of the machine's resources, through the improvement of the routing table compression and routing key allocation algorithms. A network with around 8 million neurons and over 800 million synapses has been executed, using over 200 SpiNNaker boards.

The software development has concentrated on making the platform more stable, and on improving the documentation. To this end, the data-loading and extraction protocols have been refactored to ensure correct operation. The code documentation has been reviewed and completed where parts were missing.

The SpiNNaker platform is now considered to be at TRL6. This is demonstrated in four key areas that can be used to describe this TRL:

- Prototype implementations of the software demonstrated on full-scale realistic problems: The software can run the cortical microcircuit which has a realistic number of inputs, and a prototype exists for running this in real time. The software has also successfully run a neural network across over 200 SpiNNaker boards.

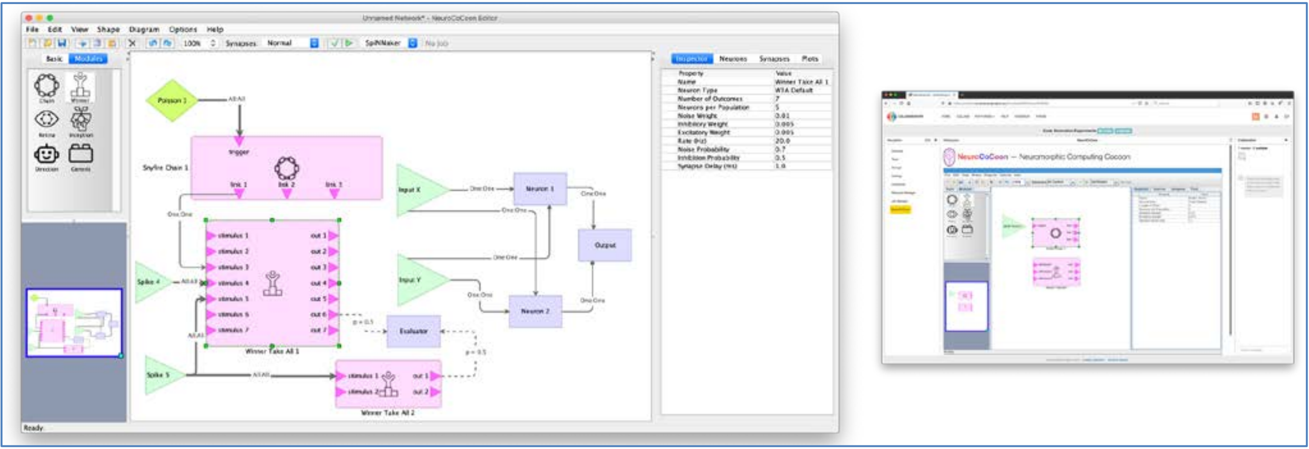
- Partially integrate with existing hardware/software systems: The software integrates with the latest version of PyNN and Neo. The SpiNNaker system has been integrated with the HBP Collaboratory and with Jupyter notebooks, and there is a proof-of-concept integration with the HBP neurorobotics platform. There have also been some successful runs of the SNNToolbox software with SpiNNaker. SpiNNaker has also been successfully used with real robots, including a basic integration with MiroBot and deeper integration with the TUM Pushbot.
- Limited documentation available: There is actually extensive documentation; including training documentation, in the form of installation instructions, lab manuals and presentations; hardware documentation, in the form of the chip datasheet and implemented protocol documentation; and user documentation, in the form of API-level documentation of the code.
- Engineering feasibility fully demonstrated: The large SpiNNaker machine has been built and successfully tested at different scales of operation. The software has also been shown to run large networks on the 1 million core machine.

Component	Link to	URL
C2 - SP9 SpiNNaker Neuromorphic Computing System	Software Repository	https://github.com/SpiNNakerManchester/sPyNNaker8
	Technical Documentation	https://spinnaker8manchester.readthedocs.io/en/latest/
	User Documentation	https://spinnakermanchester.github.io/

4.1.4 *Output 3: App for machine-learning with neuromorphic hardware*

The app for graphical neuromorphic model building (C1655) from SGA2 year one has been extended to support modular network components as an enabler for its practical use to describe larger machine-learning experiment set-ups (C1656). Modular components go beyond classical visual grouping of multiple network entities into a single node. Each module encapsulates expert knowledge about robust implementation techniques for typical subtasks on neuromorphic hardware. Only externally relevant neuron and synapse parameters are exposed in a consolidated inspector view. Coordinated sharing of consistent parameter sets in the module is expressed by the type concept for neurons and synapses in the underlying visual language. Module configuration in the editor includes scaling of the internal architecture and changes to the number of externally visible input or output ports.

The library of available composable modules (C1644) is provided as templates of ultimately PyNN-based generator functions for the module architecture. For example, the winner-take-all module provides a very universal sub-network, while the module for a head direction network is specific to the robotics domain. Beyond local installation of the app, deployment as a client-side web app inside the Collaboratory allows seamless remote access to the large-scale SP9 neuromorphic hardware platforms.



Property	Value
Name	Winner Take All 1
Neuron Type	WTA Default
Number of Outcomes	7
Neurons per Population	5
Noise Weight	0.01
Inhibitory Weight	0.005
Excitatory Weight	0.005
Rate (Hz)	20.0
Noise Probability	0.7
Inhibition Probability	0.5
Synapse Delay (ms)	1.0

Figure 2: App for graphical neuromorphic model building

Local app inspecting the parameters of a winner-take-all module instance (left) and embedded collaboratory app (right)

Progress vs State of the Art & Recent Developments

Usage of a formally defined, typed visual language and structurally variable modules, with template-based code generation, is an advance beyond existing graph editors for visual descriptions of network architectures.

Component	Link to	URL
C1656 - SP9 App for machine-learning with neuromorphic hardware	Software Repository	https://github.com/hbp-unibi/NeuroCoCoon
	Technical Documentation	https://github.com/hbp-unibi/NeuroCoCoon/blob/master/Installation.md
	User Documentation	https://github.com/hbp-unibi/NeuroCoCoon/blob/master/Usage.md
C1644 -Library of modular and composable sub-networks for computing with neuromorphic hardware systems	Software Repository	https://github.com/hbp-unibi/NeuroCoCoon/tree/master/ncc/src/de/unibi/hbp/ncc/lang/modules
	Technical Documentation	https://github.com/hbp-unibi/NeuroCoCoon/blob/master/Extending.md
	User Documentation	https://github.com/hbp-unibi/NeuroCoCoon/blob/master/Usage.md

4.1.5 Output 4: MPI-SpiNNaker

Flexible on-line reconfiguration and memory management system for SpiNNaker

The SpiNNaker neuromorphic architecture is increasingly used, not only for real-time simulation of brain-scale biological neural networks, but also to support innovative brain-inspired computational paradigms. In both domains, there is increasing demand for flexibility, in terms of network configuration and run-time redesign of network parameters and simulated neurons models. Due to the intrinsically high parallelism and complexity of the interconnected processing units, broadcasting updates to the cores is time consuming. Hence, static solutions, where the network is re-loaded from an external host, are highly inefficient. To address these requirements, we have designed the Application Command Protocol (ACP), which provides a mechanism to remotely trigger the execution of high-level op-codes by the cores and manage their application memory. We have demonstrated ACP in two SNN applications: i) SNN configuration, where simulation data are efficiently generated through ACP in the memory of computing nodes and ii) SNN reconfiguration, where ACP is used to change SNN network parameters at runtime and to easily switch from learning to test phase in a SNN classification application. The ACP protocol enabled a more flexible computational model and memory management system.

Message Passing Interface Component for SpiNNaker

Several studies have shown that neuromorphic platforms allow flexible and efficient simulations of SNN by exploiting the efficient communication infrastructure optimised for transmitting small packets across the many cores of the platform. However, the effectiveness of neuromorphic platforms in executing massively parallel general-purpose algorithms, while promising, is still to be explored. In the case of SpiNNaker, the implementation of MPI must deal with a resource limit, both in terms of memory and computing power. However, it can take advantage of the technology offered by on-chip routers, thereby obtaining efficient communication. The MPI-SpiNNaker software stack creates a simple working framework, offering a universally known programming model capable of making the SpiNNaker architecture available for a wide range of applications, both synchronous and asynchronous. (Benchmark in Urgese *et al.* 2019)

Component	Link to	URL
C3045	Software Repository	https://github.com/neuromorphic-polito/SpinACP.git https://github.com/neuromorphic-polito/SpinMPI.git
	Technical Documentation	

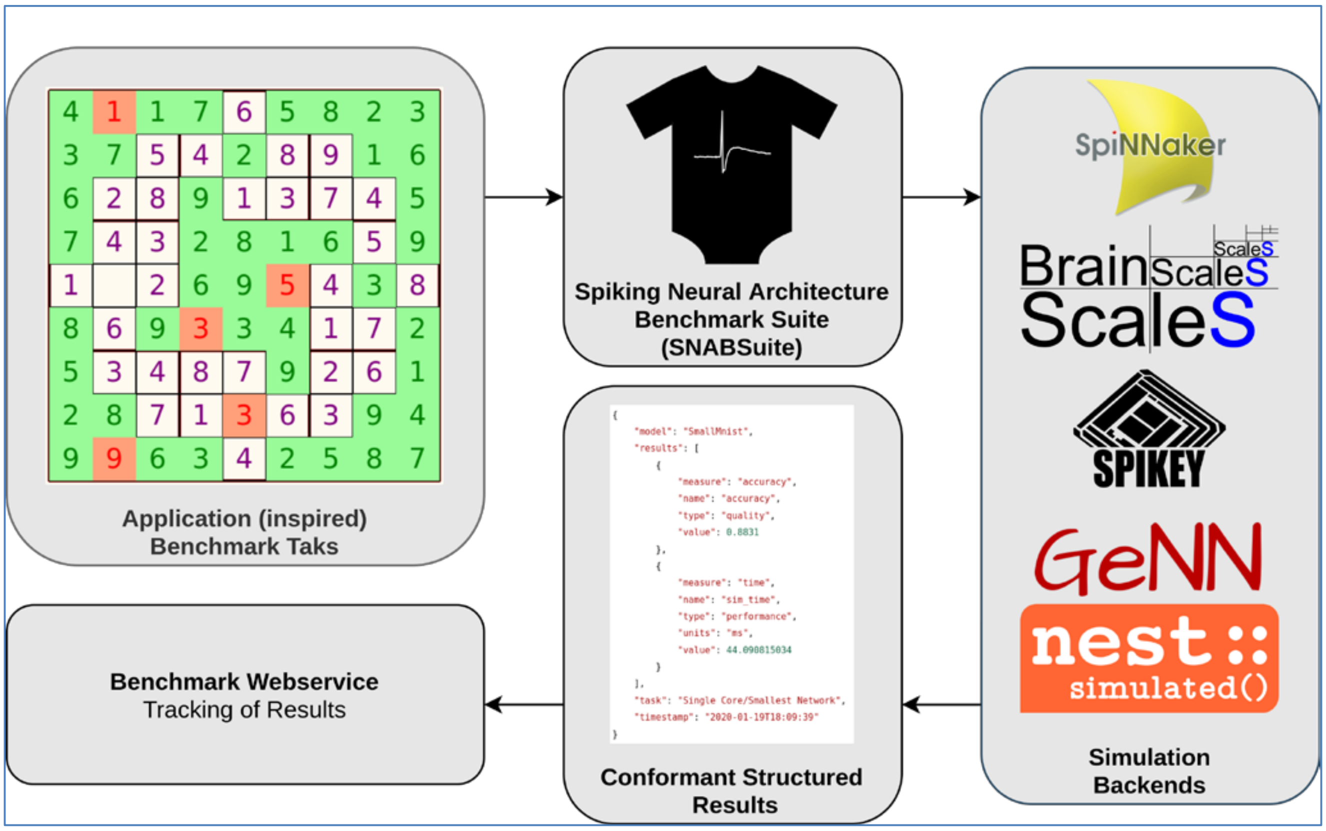
	User Documentation	

4.1.6 Output 5: Neuromorphic Benchmarks

In the second half of SGA2, the black-box benchmark framework SNABSuite (C2735) was extended to support more application driven benchmarks. Most notably, the framework now includes benchmarks on solving constraint satisfaction problems with the help of winner-take-all architectures (Ostrau *et al.*, 2019), classification benchmarks on the MNIST hand-written digits dataset, and central parts of a simultaneous localisation and mapping algorithm. The coupling to BrainScaleS has been improved, increasing the efficiency of the evaluation and utilising more features of the software stack. Furthermore, all benchmarks can be executed on Nvidia GPUs using the GeNN code-generation framework. The target TRL 6 has been reached, as a full software pipeline has been set into place, which automatically executes all benchmarks on SpiNNaker, BrainScaleS and NEST on a regular basis (Figure 3).

Progress vs State of the Art & Recent Developments

Benchmarking of neuromorphic hardware is a hot topic (for example, see the comment by Intel's Mike Davies in Nature Machine Intelligence, 2019). To our knowledge, SNABSuite is the only framework that incorporates several benchmarks and covers several hardware systems at the same time. Of a similar nature is the current endeavour to simulate the full-scale cortical microcircuit model on SpiNNaker, NEST, and GeNN, which is reported in several publications. However, these use platform-specific implementations, which is contrary to our approach.



```
{
    "model": "SmallMnist",
    "results": [
        {
            "measure": "accuracy",
            "name": "accuracy",
            "type": "quality",
            "value": 0.8831
        },
        {
            "measure": "time",
            "name": "sim_time",
            "type": "performance",
            "units": "ms",
            "value": 44.090815034
        }
    ],
    "task": "Single Core/Smallest Network",
    "timestamp": "2020-01-19T18:09:39"
}
```

Figure 3: Benchmark work-flow: from an application to the tracking of results in the web-service.

Component	Link to	URL
C2735 - SNABSuite	Software Repository	https://github.com/hbp-unibi/snabsuite
	Technical Documentation	https://hbp-unibi.github.io/SNABSuite/index.html
	User Documentation	https://github.com/hbp-unibi/SNABSuite/blob/master/README.md

4.1.7 Output 6: MUSIC library

The MUSIC communication framework enables modular development of neuronal network models and tools, and supports integration of neuromorphic hardware with HPC and neurorobotic hardware.

During the second half of SGA2, a new version of the MUSIC main development line was released (v1.1.16). This release partly serves as a reference point for current MUSIC users, but also as a preparation for an upcoming new release series supporting multiple communication algorithms (v2). In addition, a first version of MUSIC with a novel API, supporting user-requested features such as dynamic ports, has been released (v3.1.0) The two development lines (v1-2 and v3) will run in parallel, until the novel API has gained acceptance in the community.

Component	Link to	URL
C347 SP9 MUSIC library	Software Repository	https://github.com/INCF/MUSIC
	Technical Documentation	https://github.com/INCF/MUSIC/wiki/files/music-manual.pdf
	User Documentation	https://github.com/INCF/MUSIC/wiki

4.1.8 Output 7: Hardware integration

The MUSIC SpiNNaker-adapters are software components which allow any MUSIC-aware software to interface with SpiNNaker.

During the second half of SGA2, the MUSIC SpiNNaker-adapters were improved and bugs fixed. Additional examples of their use have been written.

Component	Link to	URL
C1810 - MUSIC - hardware integration	Software Repository	https://github.com/incf-music/spinnaker-adapters
	Technical Documentation	
	User Documentation	https://github.com/incf-music/spinnaker-adapters/blob/master/README.md

4.1.9 Output 8: CSA library

CSA (Connection-set algebra) is a formalism for expressing neuronal network connectivity. Novel connectivity patterns can be constructed in an abstract manner from basic patterns, using operators in a way that allows for efficient parallel instantiation of connections. A demonstration implementation in Python exists (https://github.com/INCF/csa). During the second half of SGA2, the novel C++ implementation was finalised and released (v1.0.0) and three releases of the Python implementation were made (v1.1.8, v1.1.10, v1.1.12).

Component	Link to	URL
C1769 -SP9 Connection-set algebra library	Software Repository	https://github.com/INCF/libcsa
	Technical Documentation	
	User Documentation	

4.1.10 Output 9: BrainScaleS-1 Software

The operating system (Müller *et al.* 2020a) was continuously developed, including the addition of new features, user wishes and bug fixes. Among these, the following improvements were put in place:

- HICANN chips on a wafer can be influenced by neighbouring chips when they are in an undefined state. To guarantee no mutual influence, an automated initialisation was added to the SLURM resource allocation mechanism. When requesting HICANNs, neighbouring chips are identified and a database checked to see if they are already in a defined state. If not, they are initialised and marked in the database to shorten setup times of following runs.

- A feature was added that retries to initialise the on-wafer bus network, based on verification measurements from on-chip test features.
- A configuration mode was added that allows the HICANNs to be configured incrementally, without user interaction. The automated configuration tracks the configured state and re-configures only the components with settings that were changed between two executions. This allows faster and less error-prone iterative experiments.

Component	Link to	URL
C1 - BrainScaleS-1 Neuromorphic Computing System	Software Repositories	https://github.com/electronicvisions/
	Technical Documentation	Integrated into code and internal specification
	User Documentation	https://electronicvisions.github.io/hbp-sp9-guidebook/

4.1.11 Output 10: BrainScaleS-2 Software

In pursuit of the software specification in reference document D9.2.1 - SGA2, the following additions and developments were carried out (Müller *et al.* 2020b):

- The abstraction of the FPGA instruction set now fully implements write and read operations to and from all on-chip register types, the event communication and ADC response data, as well as FPGA SPI devices. Building and execution of a timed sequence of register-like write and read accesses, in the form of a "playback programme", is fully implemented. Event support, logging and printout improvements have been added to the now-complete transport-layer formatting library "hxcomm".
- Most full-custom chip configuration entities have been abstracted into corresponding container types, enabling configuration of: synapses, synapse drivers, neurons, readout-chains and routing, plasticity processor unit programme loading, execution control and result fetching, amongst other things.
- In addition, the implementation of containers in the logical configuration layer has been started. The array and matrix shape of configuration entities on the chip, e.g. the synapse matrix or parallel-column-ADC readout has been realised as alike container structures simplifying access. An "atomic" neuron container, embodying digital configuration alongside analogue parameters belonging to the said neuron, has been developed as a structured abstract representation.
- Verification of software changes was enhanced by an automation of the hardware simulation framework for the continuous integration workflow. Upload of a change to the code review service automatically triggers a build of the software suite and a startup of a hardware simulator instance. Software unit tests are automatically verified, connecting to the started simulator instance.
- The automatically generated C++-Python wrapper source code, as well as the Python-only parts of the interfaces, have been migrated from Python 2 to Python 3.
- In-code documentation of C++ sources using doxygen has been extended to almost full coverage in hxcomm, as well as fisch; it also covers the essential parts of the low-level configuration, logical configuration and experiment control layers (haldls, stadls and lola). Building the documentation is integrated in the build flow alongside of the shared library for C++ or Python libraries. For internal use, UHEI's Jenkins continuous integration server builds and deploys the documentation as a html-based website. Publishing this code documentation to the live "SP9 Guidebook" is work-in-progress.
- Extensions for the custom vector unit of the PowerPC-based plasticity processor unit have successfully been ported from gcc 4.9.4 to gcc 8.1, which acts as a full replacement. In addition, we now support the usage of "embedded" C++17.

Component	Link to	URL
C457 - SP9 BrainScaleS 2 Neuromorphic Computing System	Software Repositories	https://github.com/electronicvisions/
	Technical Documentation	D9.2.1, also in-code documentation
	User Documentation	https://electronicvisions.github.io/hbp-sp9-guidebook/

4.2 Validation and Impact

4.2.1 *Actual and Potential Use of Output(s)*

- **Output 1** (Multi-compartmental models in PyNN). This functionality is still in the early stages of development. As yet, we are not aware of any users. Within the HBP, the Arbor development team plans to implement support for the multi-compartment PyNN API during SGA3.
- **Output 2** The SpiNNaker software is deployed as discussed in KR1. In addition to this, the software integration tests are run before any major changes are integrated into the main software, with these and some additional longer-running tests based on various user-provided scripts being run on a daily basis. This ensures that any breaking changes to the software are detected early.
- **Output 3** (App for machine-learning with neuromorphic hardware). Development versions of the app have been used in introductory spiking neural network courses for master and PhD students. The current refined version of the app is considered stable enough for such controlled guided classroom scenarios on the order of 10–20 participants.
- **Output 4** We tested the new SW library for the porting on SpiNNaker of a parallel DNA sequence-matching algorithm implemented by using the MPI programming paradigm. In the test, all cores of the board are configured for executing in parallel an optimised version of the Boyer-Moore (BM) algorithm. Exploiting this application, we benchmarked the SpiNNaker platform in terms of scalability and synchronisation latency. Experimental results indicate that the SpiNNaker parallel architecture allows a linear performance increase with the number of used cores and shows better scalability than a general-purpose, multi-core computing platform (Urgese *et al*., 2019).
- **Output 5** (Neuromorphic Benchmarks). With the increasing interest in benchmarks, we expect a growing number of users for the web-service, which is tracking benchmark results. It requires and uses the full tool-chain of the SP9 infrastructure, and contributes in form of the aforementioned web-service.
- **Output 6** (MUSIC). MUSIC has been used as a middleware in the SPORE NEST module for studying synaptic plasticity with online reinforcement learning since 2017 (Kaiser *et al*., 2019, Kappel *et al.*, 2018) as well as interfacing neuronal simulators to ROS (Weidel *et al*., 2016, Bahaguna *et al*. 2018) and the OpenAI gym (Jordan *et al*., 2017).
- **Output 7** (Hardware integration) A demonstrator application interfacing SpiNNaker with robotic hardware, using MUSIC SpiNNaker-adapters, is being developed.
- **Output 8** (CSA) CSA is used in the HBP Neurorobotics Platform (NRP) and has been used as an intermediate language in the generation of neuronal network connectivity from visual representations (Herbers, 2017).
- **Output 9** (BSS-1) and **Output 10** (BSS-2): The developments and added features for both BSS-1 and BSS-2 operating systems are used by all users of the neuromorphic hardware, especially the experiments carried out and described in KR 9.3.

4.2.2 Publications

- Brocke, Ekaterina. "Method development for co-simulation of electrical-chemical systems in Neuroscience." Doctoral thesis TRITA-EECS-AVL ; 2020:9, KTH Stockholm. **(**Output 6**,** P2398) Relevance: the thesis describes an extension of MUSIC towards multiscale co-simulatio
- Urgese, Gianvito, Francesco Barchi, Emanuele Parisi, Evelina Forno, Andrea Acquaviva, and Enrico Macii. "Benchmarking a Many-Core Neuromorphic Platform with an MPI-Based DNA Sequence Matching Algorithm." Electronics 8, no. 11 (2019): 1342, DOI 10.3390/electronics8111342. (Output 4, P2286)
- Oliver Rhodes, Luca Peres, Andrew G. D. Rowley, Andrew Gait, Luis A. Plana, Christian Brenninkmeijer and Steve B. Furber. "Real-time cortical simulation on neuromorphic hardware". Philos Trans A Math Phys Eng Sci. 2020 Feb 7;378(2164):20190160. DOI:10.1098/rsta.2019.0160 (Output 2, P2299)
- Ostrau, Christoph, Christian Klarhorst, Michael Thies, and Ulrich Rückert. "Comparing Neuromorphic Systems by Solving Sudoku Problems." In Conference Proceedings: 2019 International Conference on High Performance Computing & Simulation (HPCS). 2019. **(**Output 5, P2047**)**
- Eric Müller, Sebastian Schmitt, Christian Mauch, Sebastian Billaudelle, Andreas Grübl, Maurice Güttler, Dan Husmann, Joscha Ilmberger, Sebastian Jeltsch, Jakob Kaiser, Johann Klähn, Mitja Kleider, Christoph Koke, José Montes, Paul Müller, Johannes Partzsch, Felix Passenberg, Hartmut Schmidt, Bernhard Vogginger, Jonas Weidner, Christian Mayr, Johannes Schemmel "The Operating System of the Neuromorphic BrainScaleS-1 System", 2020a, arXiv 2003.13749 (Output 9, P2490)
- Eric Müller, Christian Mauch, Philipp Spilger, Oliver Julien Breitwieser, Johann Klähn, David Stöckel, Timo Wunderlich, Johannes Schemmel "Extending BrainScaleS OS for BrainScaleS-2", 2020b, arXiv 2003.13750 (Output 10, P2491)

5. Key Result KR9.3: Operational prototype of a 2nd generation BrainScaleS chip featuring on-chip local plasticity and non-linear dendritic processing

5.1 Outputs

5.1.1 Overview of Outputs

5.1.1.1 List of Outputs contributing to this KR

- Output 1: Deep learning with time-to-first-spike coding
- Output 2: Sampling-based Bayesian computation
- Output 3: Structural plasticity
- Output 4: Control of criticality and computation
- Output 5: Insect-inspired navigation
- Output 6: Spiking Heidelberg Digits
- Output 7: BrainScaleS-2 setups

5.1.1.2 How Outputs relate to each other and the Key Result

The research/tests described here make use of the BrainScaleS-2 chips [Schemmel, 2020] and serve as validation of the manufactured hardware systems and the necessary software environment.

Component	Link to	URL
C0457: SP9 BrainScaleS 2 Neuromorphic Computing System	Technical Documentation	Starting to appear at: https://electronicvisions.github.io/hbp-sp9-guidebook/pm/bss2.html
	User Documentation	Starting to appear at: https://electronicvisions.github.io/hbp-sp9-guidebook/pm/bss2.html

5.1.2 *Output 1: Deep learning with time-to-first-spike coding*

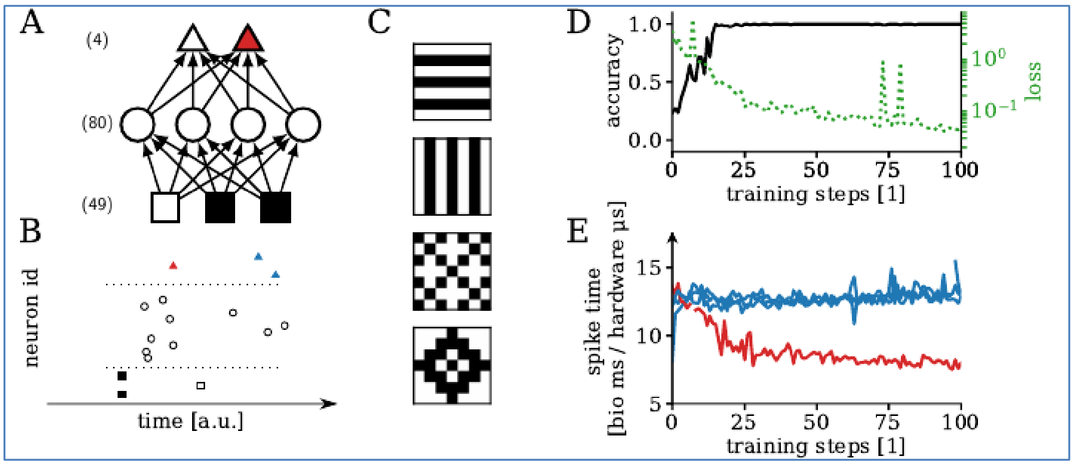


Figure 4: Feed-forward network for fast inference

For fast inference, we train a feed-forward network (Figure 4A) that uses time-to-first-spike coding, for both input and classification (B). We analytically derived differentiable equations for the spike time of leaky integrate-and-fire neurons [Goeltz, 2019] that provide learning rules based on error-backpropagation. On a sample data set (C) with inference on BrainScaleS-2 and update calculation on a host, training succeeds fast (D) and neurons coding correct classes (E, red) decrease their spike time compared to wrong classes (blue). The speed of inference (about 10µs per pattern) is independent of emulated network size, classification of larger data sets is work in progress.

5.1.3 *Output 2: Sampling-based Bayesian computation*

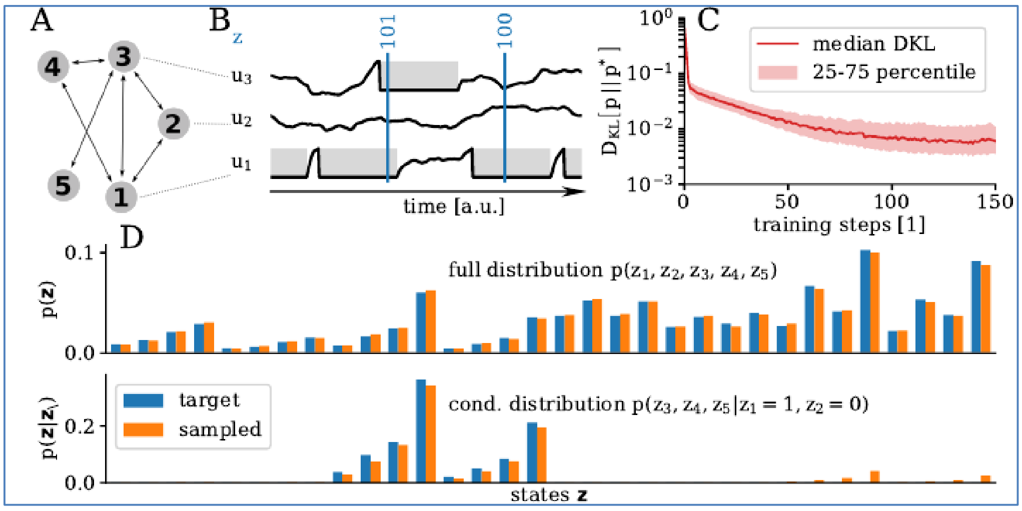


Figure 5: Network of LIF neurons for Bayesian inference

Networks of LIF neurons can perform Bayesian inference through sampling on probability distributions defined over binary variables [Petrovici, 2016]: During post-spike refractoriness, a neuron is considered to be in the state z=1, and z=0 otherwise (Figure 5A,B). We used contrastive Hebbian learning with updates calculated on a host PC, and monitor the Kullback-Leibler divergence to the target distribution (C) [Billaudelle, 2019b]. After training, the network reliably performed Bayesian inference on its target distribution (D).

5.1.4 Output 3: Structural plasticity

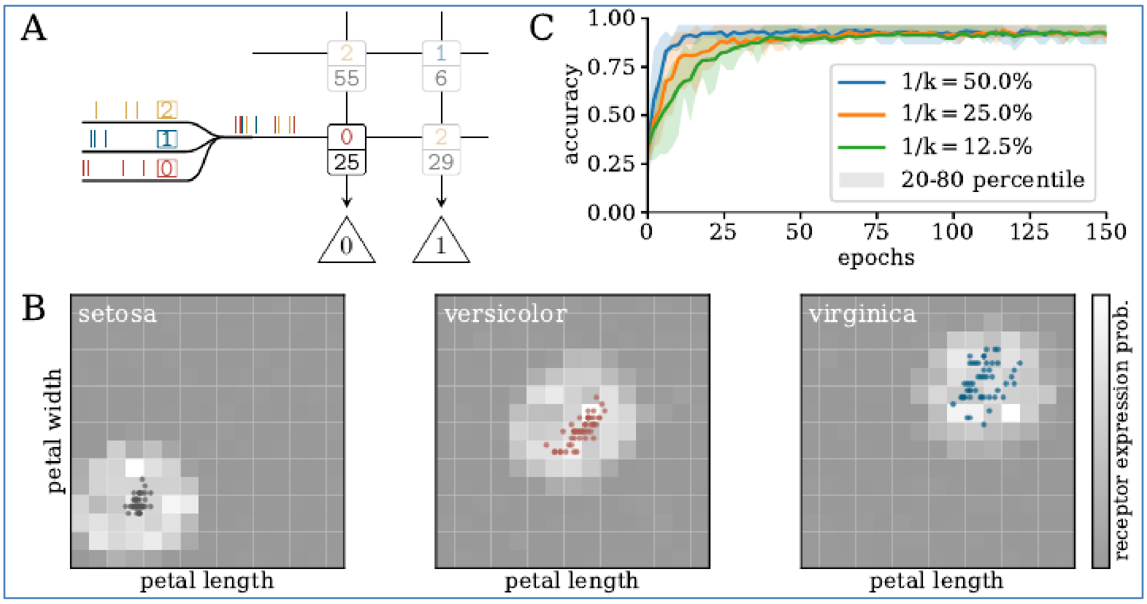


Figure 6: Structural plasticity on BrainScaleS-2

We utilise the event routing scheme inherent in the BrainScaleS-2 architecture (Figure 6A) to efficiently implement structural plasticity [Billaudelle, 2019a]. During learning, the connectome emerges from a pool of potential connections. The synaptic fan-in of a neuron is kept constant over time, resulting in a sparse connectivity structure. Our structural plasticity algorithm leads to the formation of receptive fields closely resembling the topology of the data set (B). It converges to an optimal classification performance independent of the imposed sparsity level (C), here for the Iris data set.

5.1.5 Output 4: Control of criticality and computation

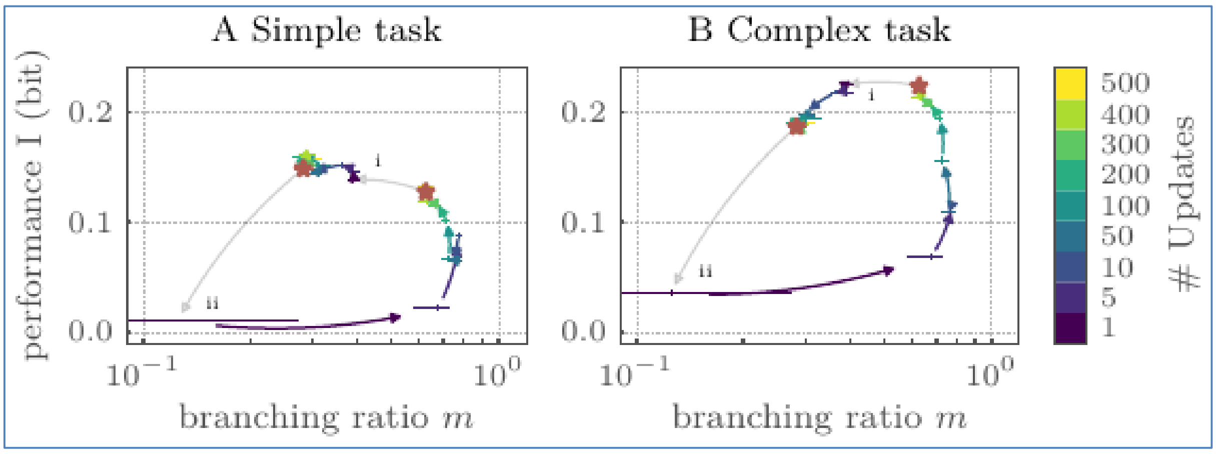


Figure 7: Influence of criticality on task performance

We developed a spiking network with on-chip synaptic plasticity for which the distance to criticality can be easily adapted by changing the input strength [Cramer, 2019a]. With this setup, we then demonstrate a clear relation between criticality, task-performance and information-theoretic fingerprint. The critical state is assumed to be optimal for any computation in recurrent neural networks, because criticality maximizes a number of abstract computational properties. Whereas the information-theoretic measures all show that network capacity is maximal at criticality, this is not the case for performance on specific tasks: Only the complex, memory-intensive tasks profits from criticality (Figure 7B), whereas the simple tasks suffer from it (A). Thereby, we challenge the general assumption that criticality would be beneficial for any task, and provide instead an understanding of how the collective network state should be tuned to task requirement to achieve optimal performance.

5.1.6 Output 5: Insect-inspired navigation

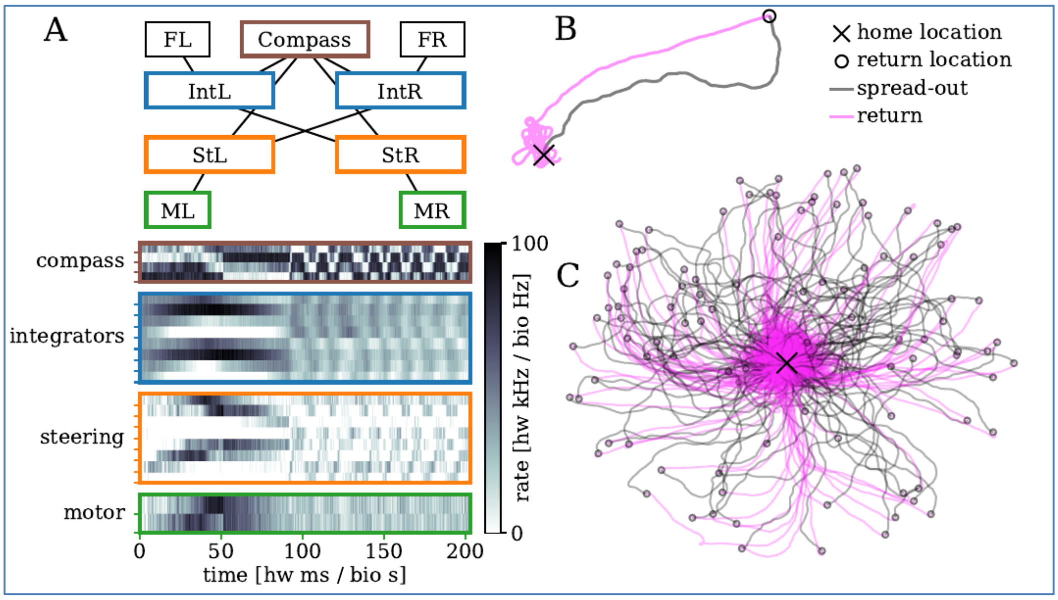


Figure 8: Virtual insectoid agent on BrainScaleS-2

A virtual insectoid agent on BrainScaleS-2 uses path integration to navigate back home after spreading out randomly [Billaudelle, 2019b]. The network is schematically depicted next to the activity histogram (Figure 8A). Information flows from the sensory layer at the top through an integration and a steering layer to the motor neurons at the bottom. A typical trajectory of the virtual insect which turns to random looping around the home position upon reaching it is shown in (B). (C) shows an overlay of 100 trajectories like that, each with a different random outbound journey.

5.1.7 Output 6: Spiking Heidelberg Digits

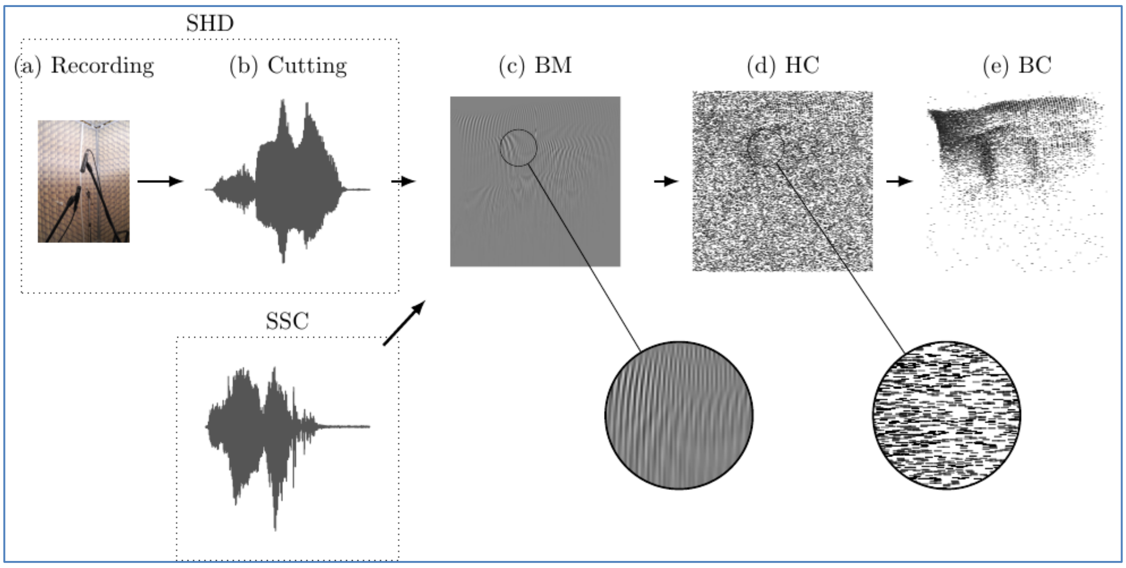
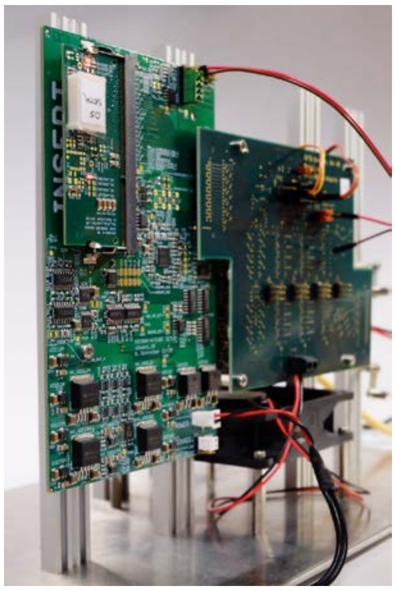


Figure 9: Representation of the Spoken digits dataset

Spiking neural networks are the basis of versatile and power-efficient information processing in the brain. To accelerate the development of spiking neural network models, objective ways to compare their performance are indispensable. Presently, however, there are no widely accepted means for comparing the computational performance of spiking neural networks. To address this issue, we introduce a general audio-to-spiking conversion procedure and provide two novel spike-based classification datasets [Cramer 2019b, Figure 9]. The datasets are free and require no additional pre-processing, which renders them broadly applicable to benchmark both software and neuromorphic hardware implementations of spiking neural networks.

5.1.8 Output 7: BrainScaleS-2 setups



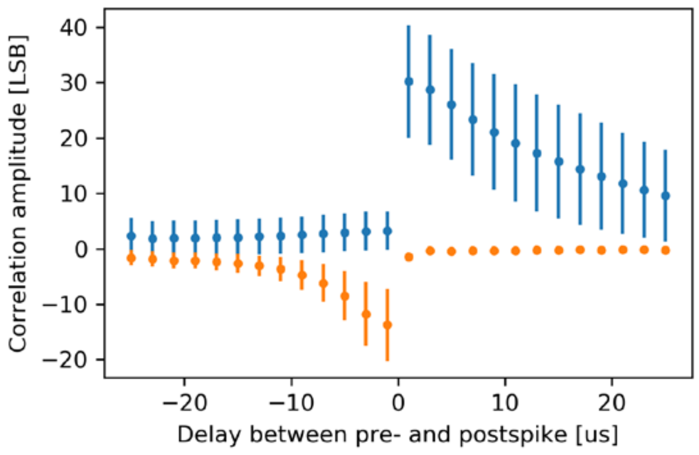

Correlation amplitude [LSB]
40
30
20
10
0
−10
−20
−20
−10
0
10
20
Delay between pre- and postspike [us]

Figure 10: BrainScaleS-2 HICANN-X setup

We developed new HICANN-X setups which can readily be used for experiments. Fourteen of them are already located in our laboratory. First experiments have been successful. In Figure 10, next to the photo of one of the setups, the synapses' correlation sensors are characterised. The plot shows means and standard deviations of all 32,768 synapses which are located in one quadrant of the chip. The timing of pre- and postsynaptic spikes is measured in every synapse. If presynaptic events are followed by postsynaptic ones, these positive delays result in causal amplitudes (blue); the inverse order implies anti-causal correlation (orange). These correlations can be processed making use of the local microprocessor, allowing for weight updates and on-chip learning based on spike-timing dependent plasticity (STDP).

5.2 Validation and Impact

5.2.1 *Actual and Potential Use of Output(s)*

The outputs are first test- and demonstration uses of the Key Result (the chip + system) and at the same time show some of the potential of the systems. The exploitation of the BrainScaleS-2 single-chip (and then also multi-chip) systems for research and, potentially, industrial use is planned for SGA3. The BrainScaleS-2 chips/systems developed and tested in SGA2 will become part of the EBRAINS platform in SGA3.

5.2.2 *Publications*

- [Billaudelle, 2019a] S. Billaudelle, B. Cramer, M. A. Petrovici, K. Schreiber, D. Kappel, J. Schemmel, and K. Meier, “Structural plasticity on an accelerated analog neuromorphic hardware system” arXiv preprint arXiv:1912.12047, 2019 (Output3, P2240)
- [Billaudelle, 2019b] S. Billaudelle, Y. Stradmann, K. Schreiber, B. Cramer, A. Baumbach, D. Dold, J. Göltz, A. F. Kungl, T. C. Wunderlich, A. Hartel, E. Müller, O. Breitwieser, C. Mauch, M. Kleider, A. Grübl, D. Stöckel, C. Pehle, A. Heimbrecht, P. Spilger, G.Kiene, V. Karasenko, W. Senn, M. A. Petrovici, J. Schemmel, K. Meier, “Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate” arXiv preprint arXiv:1912.12980, 2019 (Output 5, P2241)
- [Cramer, 2019a] B. Cramer, D. Stöckel, M. Kreft, M. Wibral, J. Schemmel, K. Meier, V. Priesemann, “Control of criticality and computation in spiking neuromorphic networks with plasticity”, arXiv preprint arXiv:1909.08418, 2019 (Output 4, P2355)
- [Cramer, 2019b] B. Cramer, Y. Stradmann, J. Schemmel, F. Zenke, “The Heidelberg spiking datasets for the systematic evaluation of spiking neural networks”, arXiv preprint arXiv:1910.07407, 2019 (Output 6, P2356)
- [Goeltz, 2019] J. Göltz, A. Baumbach, S. Billaudelle, O. Breitwieser, D. Dold, L. Kriener et al., “Fast and deep neuromorphic learning with time-to-first-spike coding” arXiv preprint arXiv:1912.11443, 2019 (Output 1, P2239)
- [Schemmel, 2020] Johannes Schemmel, Sebastian Billaudelle, Phillip Dauer, Johannes Weis “Accelerated Analog Neuromorphic Computing” arXiv 2003.11996 (BrainScaleS-2)

6. Key Result KR9.4: Operational prototype of a second generation SpiNNaker chip featuring 10-fold improved energy efficiency, 144 Cortex M4F and 36 GIPS/Watt per chip

6.1 Outputs

6.1.1 Overview of Outputs

6.1.1.1 List of Outputs contributing to this KR

Output 1: SpiNNaker next generation (NM-MC2, SGA2) chip (Component id: C2628), demonstrating new key chip hardware feature for >10-fold energy efficiency improvement

6.1.2 Output 1: SpiNNaker next generation chip

Measurements of the SpiNNaker2 prototype JIB1 have been performed, with focus on the energy efficiency of the processing elements (PE) for neuromorphic and machine learning applications, with the following main results:

- Power measurement results of the PE at 250MHz from 0.50V supply voltage
  - CoreMark Benchmark achieving **20uW/MHz (20pJ/operation)**
  - Matrix multiplication using 16x4 MAC arrac (int8) achieving **3.4Tops/W**.
- Operational chip-to-chip Links at 1GBit/s/lane with 4pJ/Bit energy consumption.

A JIB2 test chip has been implemented, which contains at least one instance each of theSpiNNaker2 building blocks, as shown in Figure 11.

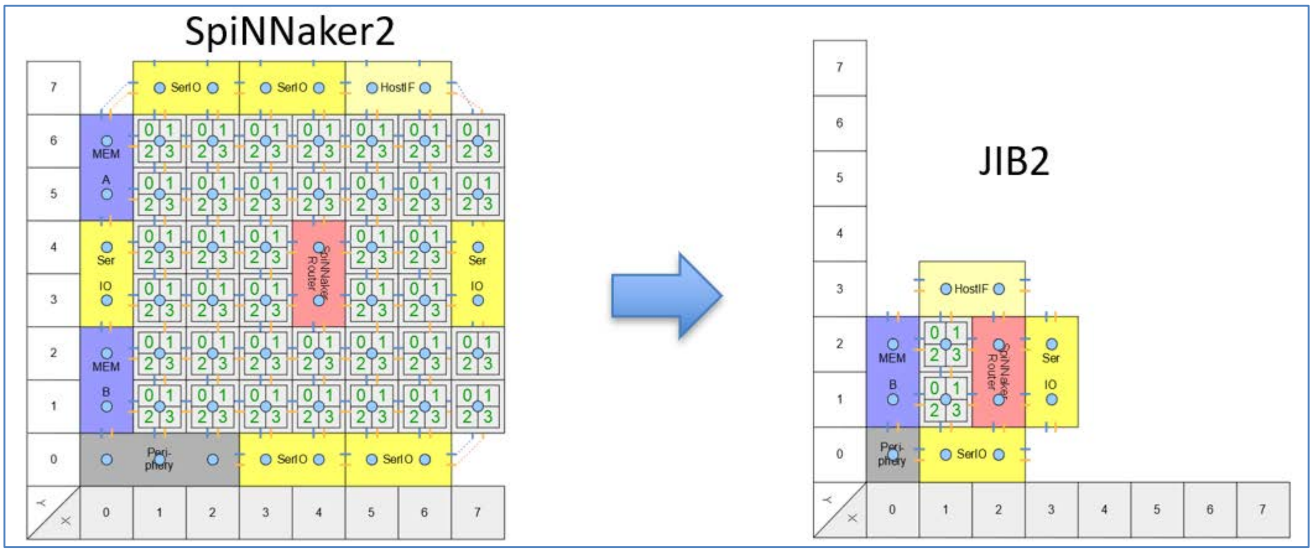


Figure 11: JIB2 test chip

These macros include:

- LPDDR4 Memory interface (PHY+Controller) for external DRAM connection
- Processing Element (PE) and Quad-Processing Element Cluster (QPE)
- Chip-to-Chip Spike Communication Links (horizontal and vertical instance)
- Host-Interface (Ethernet and FPGA connection)
- SpiNNaker Router
- Periphery Block (Management processor, GPIO)

The tape out-ready layout of the JIB2 test chip is shown in Figure 12:

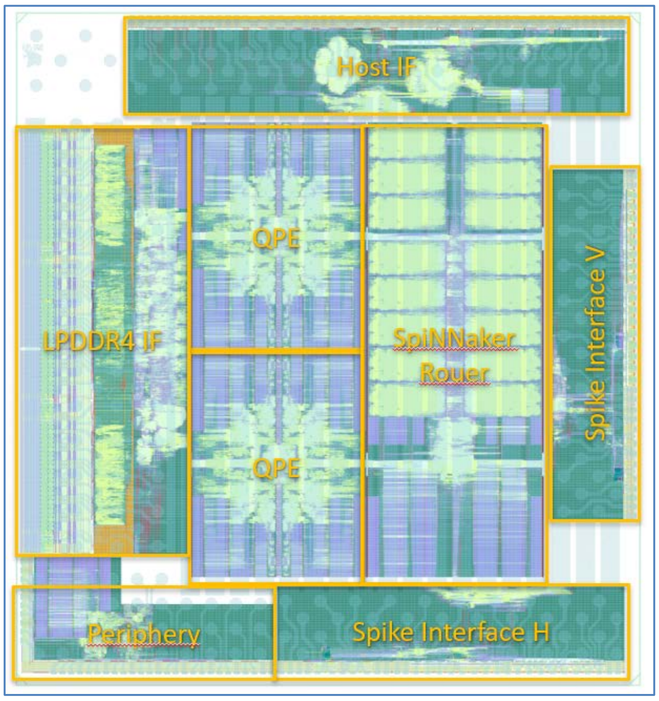


Figure 12: Tape-out ready layout of the JIB2 testchip

With this, all SpiNNaker2 components can be validated in a silicon prototype before integrating them in the final SpiNNaker2 chip, with the full scaling of 152 PEs, 6 Chip-to-Chip Links and two LPDDR4 memory interface instances.

The following table shows the updated performance of the HPB SpiNNaker2 prototypes, compared to SPiNNaker1.

Table 1: SpiNNake-2 prototype performance, compared to SpiNNaker-1

	SpiNNaker 1 [1]	SpiNNaker2 prototype from SGA1 [2] (C467)	SpiNNaker2 prototype JIB1 from SGA2 (C2628)	SpiNNaker2 prototype JIB2 from SGA2 (C2628)	SpiNNaker2 (estimation)
Processing Element Features					
Technology	130nm	28nm	22nm FDSOI		
PE clock frequency	200MHz	125MHz to 500MHz	200 to 400MHz	150 to 300MHz	
PE MAC accelerator	no	no	Yes	Yes	
PE neuromorphic accelerators (exp, PRNG)	no	yes	Yes		
Power Management	no	Dynamic voltage and frequency scaling (3 levels)	Dynamic voltage and frequency scaling (2 levels), adaptive body biasing		
Nominal Supply Voltage	1.2V	0.70V to 1.0V	0.40V to 0.60V	0.50V to 0.80V	
PE processor energy efficiency [pJ/operation] at room temperature	130 (measured)	45 (measured)	20 (measured simulations with neuromorphic testcase) **Efficiency enhancement compared to SpiNNaker1: 11.8x**	20 (estimated from sign-off simualtions with neuromorphic testcase) **Efficiency enhancement compared to SpiNNaker1: 11.8x**	
PE Peak throughput MAC per second	0.10G	0.25G	25.60G	19.20G	

	SpiNNaker 1 [1]	SpiNNaker2 prototype from SGA1 [2] (C467)	SpiNNaker2 prototype JIB1 from SGA2 (C2628)	SpiNNaker2 prototype JIB2 from SGA2 (C2628)	SpiNNaker2 (estimation)
Total MAC per Chip	1.6G	1.0G	0.204T	0.153T	2.9T
Neuromorphic Benchmark Example					
Relative time per 1 Nengo time Step Update (relative), 100 input dimensions	1	0.4	<0.1 **Performance enhancement compared to SpiNNaker1: >10x**		

6.2 Validation and Impact

6.2.1 *Actual and Potential Use of Output(s)*

The SpiNNaker-2 prototypes (Santos, JIB1, JIB2) will continue to be available within the HBP and will be used to support various science projects, in SGA3, including in particular small-scale robotics activities.

The full SpiNNaker2 chip will be used outside the HBP to build the SpiNNcloud system under a EUR 8 million grant from the Saxony Science Ministry, and the European Regional Development Fund (ERDF). Further commercial opportunities will be sought for industrial applications of variants of the chip in the fields of automotive and robotics applications.

6.2.2 *Publications*

- Y. Yan et al., "Efficient Reward-Based Structural Plasticity on a SpiNNaker 2 Prototype," in IEEE Transactions on Biomedical Circuits and Systems, vol. 13, no. 3, pp. 579-591, June 2019. DOI 10.1109/TBCAS.2019.2906401 (P1828)
    - Relevance: showing the implemented prototype in action

7. Key Result KR9.5: Two novel theories of computational principles: learning-to-learn (L2L) and network learning based on dendritic computation

7.1 Outputs

7.1.1 *Overview of Outputs*

7.1.1.1 List of Outputs contributing to this KR

- Output 1: Output 1: Learning-to-learn (L2L) applied to e-prop
- Output 2: Application of L2L to Reservoir Computing
- Output 3: L2L for robotics

7.1.1.2 How Outputs relate to each other and the Key Result

The Outputs describe the application of the L2L theory to different training settings for neuromorphic hardware.

Component	Link to	URL
C2549 - SP9 Learning-to-Learn methods to learn many tasks from few examples	Technical/User Documentation	https://igitugraz.github.io/L2L/

7.1.2 *Output 1: Learning-to-learn (L2L) applied to e-prop*

One of the main impediments for more wide-spread application of neuromorphic hardware is the lack of powerful on-chip learning methods for recurrent networks of spiking neurons (RSNNs), and the large number of training examples that are needed for learning, even for artificial neural networks. TU Graz has developed the e-prop algorithms (Bellec *et al.*, 2019b) for on-chip training of RSNNs that are currently being implemented on SpiNNaker at the University of Manchester. In order to tackle the second problem, reducing the required number of training examples, TU Graz has combined e-prop with Learning-to-Learn (L2L).

The setup of L2L involves an infinitely large family F of learning tasks C. In general, learning is carried out simultaneously in two loops: the inner loop and the outer loop, see Figure 13B. In the inner loop, a neural network N is concerned with solving specific tasks C. On the other hand, the outer loop is responsible for increasing the learning speed in the inner loop. To achieve this, some parameters of N (termed hyper-parameters) are optimised in an outer loop optimisation to encourage fast learning of a randomly drawn task C from F. Outer loop training, which can be implemented by various optimization methods, proceeds on a much slower time scale than the inner loop, which allows integration of performance evaluations from many different tasks C of the family F. One can interpret this outer loop training as an analogue to longer-term evolutionary and developmental processes, as well as prior learning, in brain networks that install abstract prior knowledge about the structure of the family F in the hyper-parameters of N.

In the combination of L2L with e-prop, TU Graz trained an auxiliary RSNN in the outer loop (shown at the top of Figure 13A) that produces learning signals for a main RSNN. This outer loop training can be seen from the functional perspective as corresponding to the optimisation of neuromodulatory systems, which are known to provide target-specific learning signals in the brain, through evolution. As family F of tasks, we chose in our first demonstration the set of all possible movements of a two-link arm (see Figure 13C).

It was demonstrated in Figure 13D, E that L2L enables in this application one-shot learning of new arm movements. In fact, the RSNN could solve this movement control task without first generating an inverse model.

Interestingly, the learning signals that were emitted by the auxiliary RSNN, which had been optimised in the outer loop on this family F of tasks, differed quite strongly from the learning signals (gradients) that are used by backpropagation through time (BPTT), see Figure 13F. This suggests that e-prop, in combination with L2L, can surpass even the offline BPTT learning method, by making use of prior knowledge of the types of tasks that are to be learned.

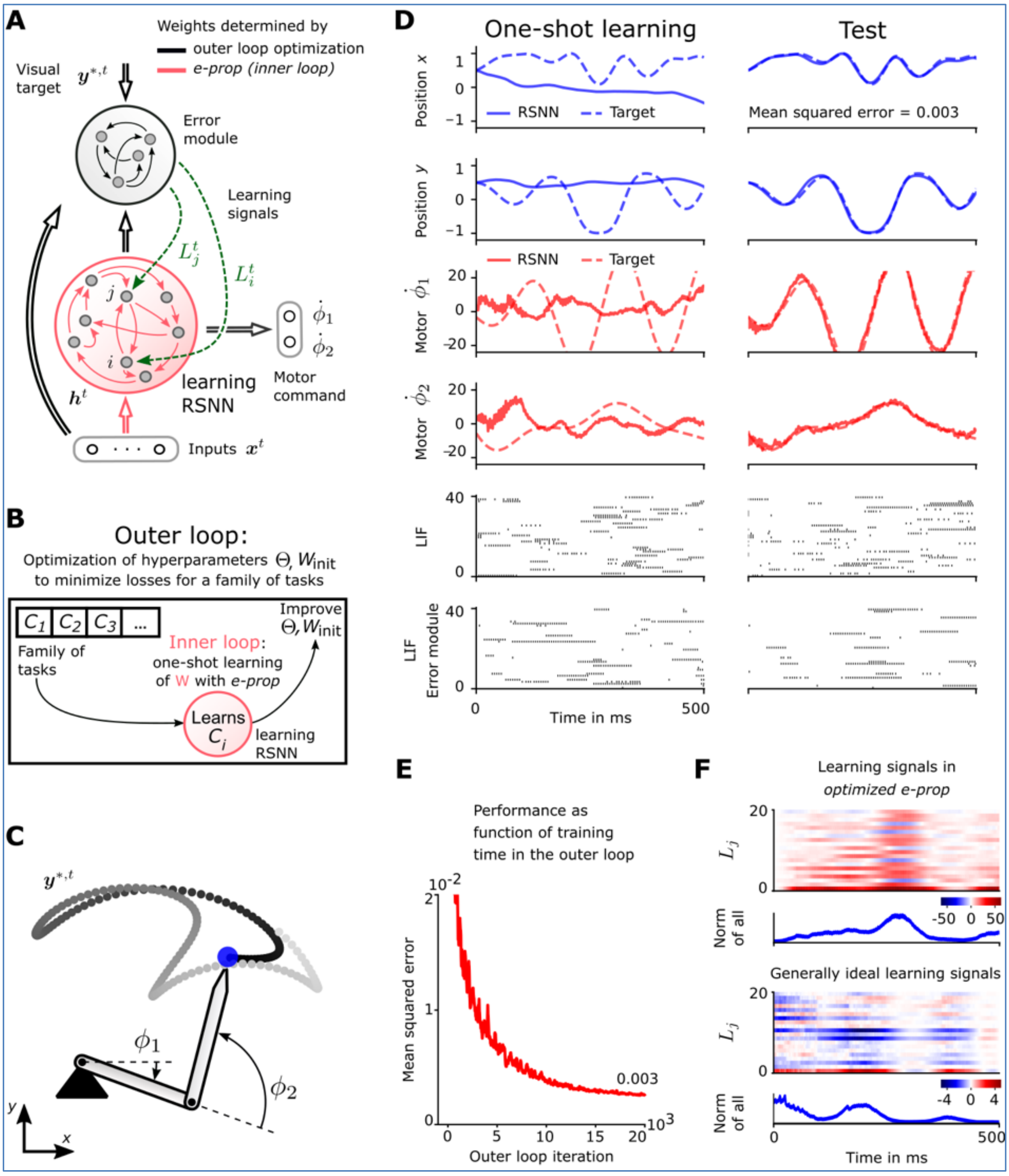


Figure 13: Scheme and performance of L2L with e-prop

A) Learning architecture of L2L with e-prop. In this demo the angular velocities for the joints were controlled by an RSNN. An auxiliary RSNN (error module) produces suitable learning signals for the main RSNN. B) Scheme of the two-tiered optimization procedure in L2L. C) Randomly generated target movements (example shown) had to be reproduced by the tip of an arm with two joints. D) Demonstration of one-shot learning for a randomly sampled target movement. During the training trial the error module sends learning signals (top row of Panel F) to the network. After a single weight update the target movement can be reproduced in a test trial with high precision. E) One-shot learning performance improved as the optimization in the outer loop proceeds. F) Comparison of learning signals emitted by the error module to learning signals as computed with BPTT.

7.1.3 Output 2: Application of L2L to Reservoir Computing

Reservoir computing is a common computing and learning paradigm in neuromorphic hardware.

In Subramoney *et al.* (2019), TU Graz investigated the extent to which one can improve the computing and learning performance by using L2L to optimise the reservoir in the outer loop of L2L.

Specifically, L2L was applied to an RSNN as reservoir, and its hyper-parameters were optimised in the outer loop with BPTT. It was found that the learning speed and performance of this reservoir was substantially enhanced through L2L, see Figure 14. Secondly, TU Graz applied L2L to a new form of reservoir, where not even a linear readout is adapted for specific tasks, and showed that this tends to enable even faster learning. In this particular case, the reservoir learned to retain task-relevant information in its own network dynamics.

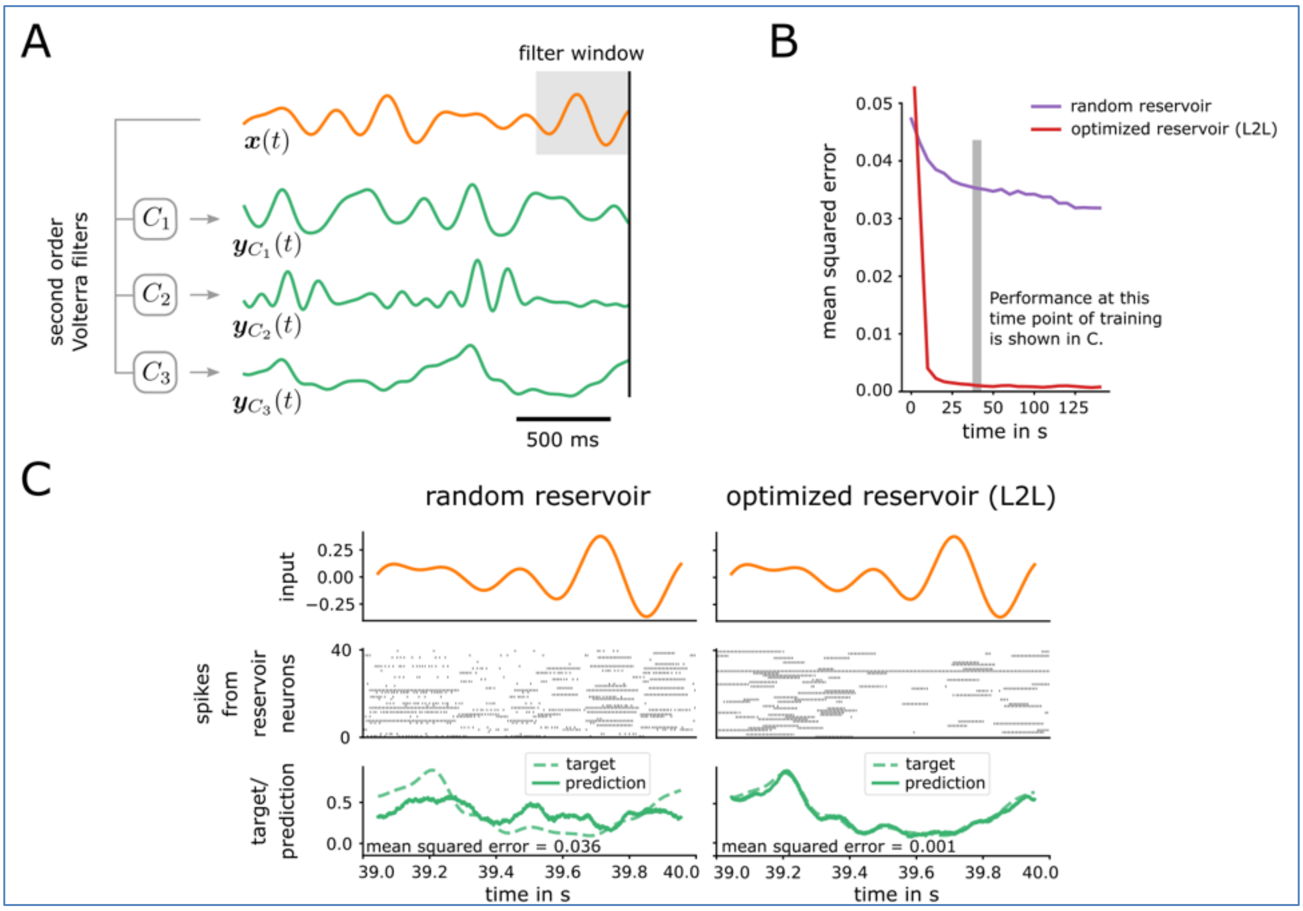


Figure 14: Learning-to-learn a nonlinear transformation of a time series

A) Different tasks arise by sampling second order Volterra kernels according to a random procedure. Input time series are given as a sum of sines with random properties. To exhibit the variability in the Volterra kernels, three examples where different Volterra kernels are applied to the same input are shown. B) Learning performance in the inner loop when adapting the linear readout, both for the case of a reservoir with random weights, and for a reservoir that was trained in the outer loop by L2L. Performance at the indicated time window is shown in Panel C. C) Sample performance of a random reservoir and of an optimized reservoir after readouts have been trained for 10 seconds. Network activity shows 40 neurons out of 800.

7.1.4 Output 3: L2L for robotics

While long-term learning of cognitive and motor skills requires synaptic plasticity, behaviour often requires much faster learning, sometimes within a single or a few trials. One of the important components of behaviour is the ability of the brain to predict how the body responds to its motor commands; that is, perform motor prediction. Inspired by recent evidence in biology, TU Graz used L2L to develop a method for fast learning of motor prediction, that uses only the internal dynamics of an LSNN (an RSNN that includes adapting neurons), without any synaptic plasticity. In a first demonstration with a simple two-link arm model (Figure 15), it was shown that the network could adapt,-without using synaptic plasticity, to arms with different link lengths and masses, within a few 100ms.

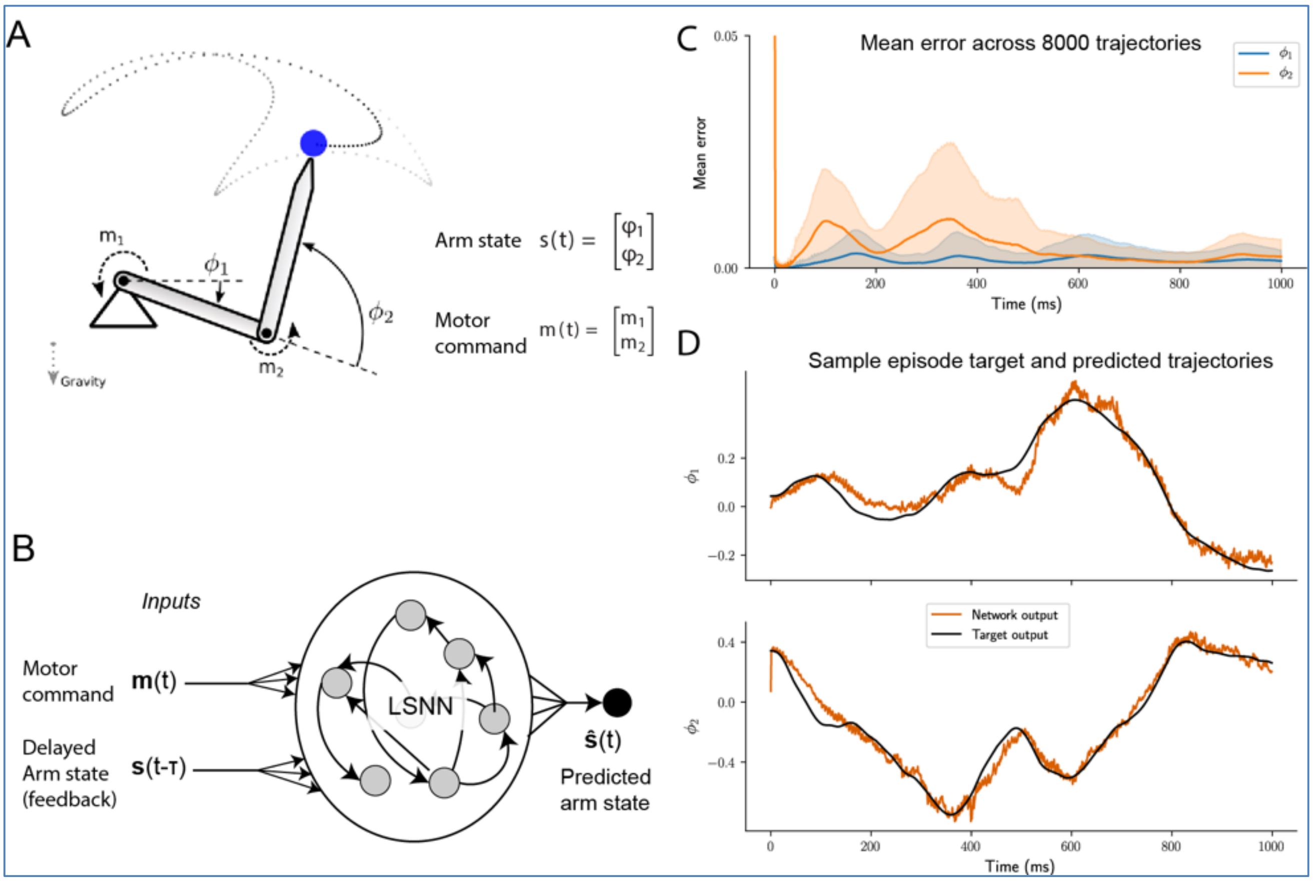


Figure 15: Learning-to-learn for a simple robotic arm:

A) Illustration of the two-link arm model with states given by the angle of the links, and the motor command applied on both the joints. B) Architecture for LSNN with inputs and outputs. All the weights are only updated in the outer loop training using BPTT, and remain fixed during testing. C) Mean error over all test episodes during the 1 second of inner loop learning. D) Target trajectories and network prediction for one sample test episode for an arm with previously unseen link lengths and masses.

7.1.5 Output 4: Network learning based on dendritic computation

Based on the experimental results of Matthew Larkum and others, TU Graz developed a supervised learning rule for a multi-compartment model of a pyramidal neuron, based on the role of dendritic Ca+2 spikes and demonstrated that this new learning rule enables a neuron to learn difficult sequence prediction tasks. A preprint of the planned journal publication is available.

7.2 Validation and Impact

7.2.1 Actual and Potential Use of Output(s)

The Output descriptions above show the application of L2L to different training problems/situations.

7.2.2 Publications

- P1997: Subramoney, A., Scherr, F., & Maass, W. (2019). Reservoirs learn to learn. In: Reservoir Computing, K. Nakajima, I. Fischer, eds., Springer 2020. arXiv preprint arXiv:1909.07486.

- P1998: Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2019). A solution to the learning dilemma for recurrent networks of spiking neurons. bioRxiv, 738385. DOI 10.1101/738385
- P2460: Rao, A., Subramoney, A., Legenstein, R., Maass, W.: A biologically motivated rule for supervised learning with spiking neurons. Preprint

8. Key Result KR9.6: Applications exploiting the new features of second generation systems

8.1 Outputs

8.1.1 Overview of Outputs

8.1.1.1 List of Outputs contributing to this KR

- Output 1: Mapping Deep Neural Networks (DNNs) to the SpiNNaker architecture.
- Output 2: Learning-to-Learn (L2L) applied to reinforcement learning on BrainScaleS
- Output 3: BrainScaleS applications.
- Output 4: Applications developed in CDP5.

Component	Link to	URL
C2549 - SP9 Learning-to-Learn methods to learn many tasks from few examples	Technical/User Documentation	https://igitugraz.github.io/L2L/

8.1.1.2 How Outputs relate to each other and the Key Result

Output 1 is part of the SpiNNaker2 development and specifically demonstrates that this platform can be applied to solving deep-learning problems.

Output 2 demonstrates the utility of the second generation features of the BrainScaleS-2 system in finding optimised parameters for plasticity rules in specific tasks.

Output 3 sums all outputs reported by the BrainScaleS-2 team that exploit the second-generation features of that system.

Output 4 summarises the outputs produced by CDP-5 “Biological deep learning” that exploit the second-generation features of HBP hardware.

8.1.2 Output 1 - Mapping DNNs on SpiNNaker2

Work was done on distributed mapping of deep neural networks (DNNs) onto the SpiNNaker2 architecture, exploiting the speed-up gained by the multiply-accumulate hardware accelerators in each SpiNNaker2 processing element. Results show that hardware acceleration leads to a full utilisation of off-chip memory bandwidth resources. Data reuse strategies alleviate this limitation, resulting in a factor 3 faster DNN layer execution. As a result, complete DNN visual benchmark networks like VGG-16 or ResNet-50 can be processed in a few tens of milliseconds.

Progress vs State of the Art & Recent Developments

The recent developments enable mapping DNNs on SpiNNaker more rapidly and efficiently than the previous state of the art.

8.1.3 Output 2 - Learning-to-Learn (L2L) applied to reinforcement learning on BrainScaleS

The use of analogue neuromorphic hardware requires cumbersome hand-tuning of hyper-parameters and details of learning algorithms. On the other hand, neuronal networks in the brain have been optimized by evolutionary and developmental processes. TU Graz and UHEI showed that one can apply this biological paradigm also to analogue neuromorphic hardware. More specifically, it was shown that agents, implemented on the HICANN-DLS, can learn very efficiently from rewards, provided that the underlying hyper-parameters and details of the plasticity rule are optimized for this type of task by L2L (Figure 16). The results have now been published in a journal (Bohnstingl *et al.*, 2019).

Progress vs State of the Art & Recent Developments

The new developments have confirmed previous work towards L2L on BrainScaleS and have now been peer reviewed and published in a journal.

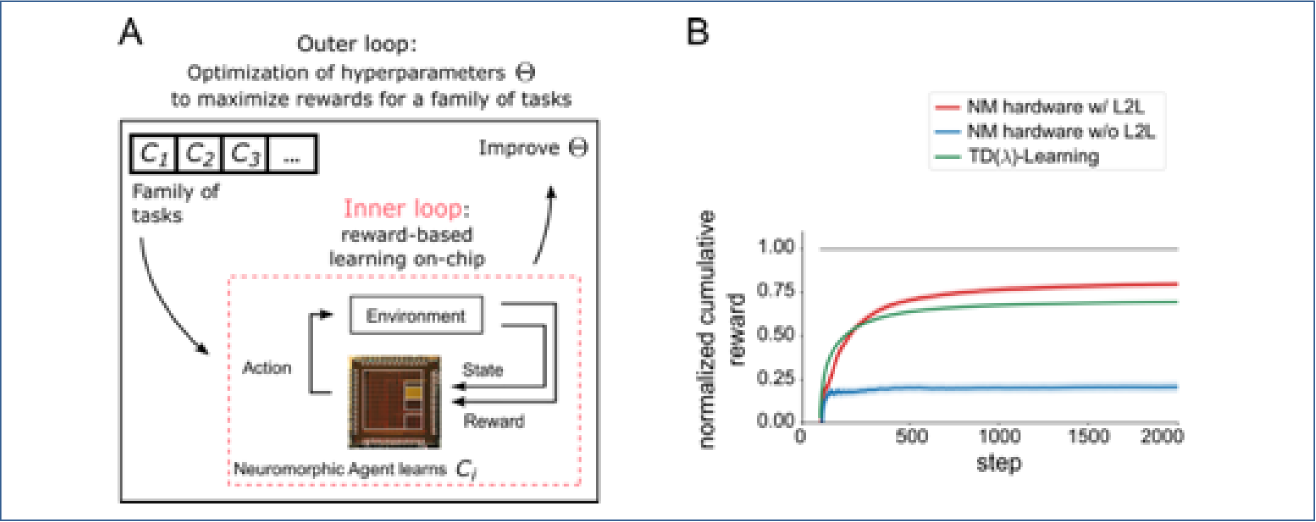


Figure 16: L2L on Brainscales

A) Schematic of the L2L setup. In the inner loop a neuromorphic agent learns a specific task Ci by reinforcement learning. The outer loop optimises learning performance of the agent on all tasks from the family of tasks. B) Learning performance of the neuromorphic agent w/ and w/o L2L on a family of Markov decision processes. TD(λ) is a baseline learning algorithm.

8.1.4 Output 3 - BrainScaleS applications

A number of applications have been developed that exploit the second-generation features of the BrainScaleS architecture. These Outputs include:

- Deep learning with time-to-first-spike coding
- Sampling-based Bayesian computation
- Structural plasticity
- Control of criticality and computation
- Insect-inspired navigation
- Spiking Heidelberg Digits

These Outputs are described in detail in KR9.3.

8.1.5 Output 4 - Structural plasticity on BrainScaleS-2

CDP 5 has reported an additional application that exploits the second-generation features of the BrainScaleS-2 hardware.

An efficient implementation of structural plasticity was proposed and its functionality demonstrated on the BrainScaleS-2 system. The plasticity rule enables neurons to dynamically select a set of suitable synapses, out of a pool of potential connections. This policy optimises performance, while at the same time maintaining a sparse connectome. It makes extensive use of the PPU plasticity processing unit that is available in BrainScaleS-2.

The results demonstrate how to employ on-chip structural plasticity in BrainScaleS-2 and can be applied to other learning frameworks. This expands the set of experimental scenarios that are amenable to emulation on BrainScaleS-2 and also offers similar potential for other neuromorphic architectures (Billaudelle *et al.*, 2019a and b).

A detailed report is available in D9.4.2.

8.2 Validation and Impact

8.2.1 Actual and Potential Use of Output(s)

The reported outputs have been used to demonstrate the viability of 2nd-generation features of the HBP hardware and to provide palpable examples, showing how to use them in scientific experiments. They lead the way to future use of the 2nd-generation HBP hardware platforms after SGA2. Since the 2nd-generation hardware systems are only in prototype stage and have not been made accessible to the public, we cannot report on usage, the user base or user feedback. Their future development will be pursued outside of the HBP, so any potential exploitation is to be determined in the respective follow-up projects.

8.2.2 Publications

- T. Bohnstingl, F. Scherr, C. Pehle, K. Meier, and W. Maass, "Neuromorphic Hardware Learns to Learn", Frontiers in Neuroscience 13:483, 2019, DOI 10.3389/fnins.2019.00483 (P1901)
    - *Demonstrates how Learning-to-Learn can be implemented and applied on HBP hardware.*
- Sebastian Billaudelle, Yannik Stradmann, Korbinian Schreiber, Benjamin Cramer, Andreas Baumbach, Dominik Dold, Julian Göltz, Akos F. Kungl, Timo C. Wunderlich, Andreas Hartel, Eric Müller, Oliver Breitwieser, Christian Mauch, Mitja Kleider, Andreas Grübl, David Stöckel, Christian Pehle, Arthur Heimbrecht, Philipp Spilger, Gerd Kiene, Vitali Karasenko, Walter Senn, Mihai A. Petrovici, Johannes Schemmel, Karlheinz Meier "Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, arXiv preprint arxiv: 1912.12980, 2019a (P2241)
- S. Billaudelle, B. Cramer, M. A. Petrovici, K. Schreiber, D. Kappel, J. Schemmel, and K. Meier, "Structural plasticity on an accelerated analog neuromorphic hardware system" arXiv preprint arXiv:1912.12047, 2019b.(P2240)
    - *Demonstrates how 2nd-generation features of the BrainScaleS hardware can be used to implement Structural plasticity.*

9. KR9.x: Results by SP9 not under the KR9.1-6

In addition to the work directly contributing to the six SP9 Key Results, considerable research and maintenance work was also done, which does not fall directly into the Key-Results categories.

9.1 Outputs

9.1.1 Overview of Outputs

The KR9.x section describes results of the work in SP9 in SGA2 year 2, which do not belong directly into the KR1-6 sections

9.1.1.1 List of Outputs contributing to this KR

- Output 1: Use Cases (SGA2-SP9-UC001, SGA2-SP9-UC002, SGA2-SP9-UC003)
- Output 2: Continuously running the first generation SpiNNaker and BrainScaleS machines
- Output 3: Embedding next generation wafers
- Output 4: NEAL Components and Agents
- Output 5: L2L system for few shot object recognition
- Output 6: Event-based electronic olfaction with SpiNNaker 2
- Output 7: Additional theory advances
  - E-prop: A biologically plausible and hardware friendly approximation to optimal gradient descent learning of recurrent SNNs
  - Deep Reinforcement Learning (Deep RL) with recurrent SNNs
  - Efficient ANN-to-SNN conversion for state-of-the-art image classification with few spikes per image
  - A theory of brain computation as basis for the design of more powerful SNN architectures
  - New learning methods for spike-based robot control

9.1.2 Output 1: Use-cases

9.1.2.1 SGA2-SP9-UC001 Open loop run of a complex spiking network with input data, output data and network reconfiguration by learning

SpiNNaker has demonstrated support for this Use Case with a number of example networks. The cortical microcircuit example is a complex spiking neural network, with 80,000 neurons and 0.3 billion synapses, and has input data in the form of Poisson spikes from the surrounding brain, and output data in the form of the spikes recorded. The output data were verified against the NEST simulator, and statistical analysis showed that the results were valid.

The cortical microcircuit example does not include learning, however. Network configuration by learning has been demonstrated with a different example in the form of the use of Structural Plasticity and STDP in the solving of the MNIST data set (Bogdan et. al. 2018). Again, this has input data, in the form of the digits to be recognised, and output data, in the form of the categories of the digits recognised.

BrainScaleS has demonstrated support for this Use Case in the context of time-to-first-spike coding. The network is one of the first neuromorphic implementations of classification with spike timing only, compared to the less efficient rate coding. The learning is done iteratively, with the chip-in-the-loop. Time-to-first-spike has been implemented successfully on both generations of BrainScaleS.

Spike-based sampling is also used to demonstrate open-loop learning on the BrainScaleS-2 platform where it is shown that neurons can perform Bayesian inference through sampling on probability distributions defined over binary variables.

Work has also been done to bring a version of the cortical column to the first generation system in the context of parameter translation and mapping studies.

9.1.2.2 SGA2-SP9-UC002 Closed loop run of a complex spiking network with input data, output data and network reconfiguration by learning

SpiNNaker has demonstrated support for this use case in a number of examples. Of particular note is the work on using the e-prop learning rule on SpiNNaker in conjunction with a grasping robot. The input to this network was a DVS sensor on a robotic head. The events were then fed to a neural network with eProp learning running on SpiNNaker in real-time, using the live-spike-streaming features of the software. The network was then trained to recognise four classes of objects, one of which was just “background” with 200 samples. The output of the network was the motion of the grasping of the appropriate object. The network performed with an accuracy of over 99%.

BrainScaleS has demonstrated support for this Use Case with several networks; for example, showing structural plasticity and learning on-chip, simulating a virtual environment for insect navigation, as well as the control of critically with liquid computing as a possible application. For the latter, a clear relation between criticality, task-performance and information-theoretic fingerprint was demonstrated. The structural plasticity algorithm leads to the formation of receptive fields closely resembling the topology of the data set, allowing efficient and automatic use of the limited hardware resources.

9.1.2.3 SGA2-SP9-UC003 Learning-to-learn (LTL) in a complex spiking network with input data, output data and network reconfiguration by learning

See the KR9.5: L2L descriptions

9.1.3 Output 2: Continuously running the first generation SpiNNaker and BrainScaleS machines

The large neuromorphic compute systems SpiNNaker-1 (1 million core machine in Manchester, C2) and BrainScaleS (Wafer scale analogue physical compute system in Heidelberg, C1) are continuously operated and maintained for project-internal and public access. The public “Getting started” page for both systems can be found on the HBP public website at: https://www.humanbrainproject.eu/en/silicon-brains/start/.

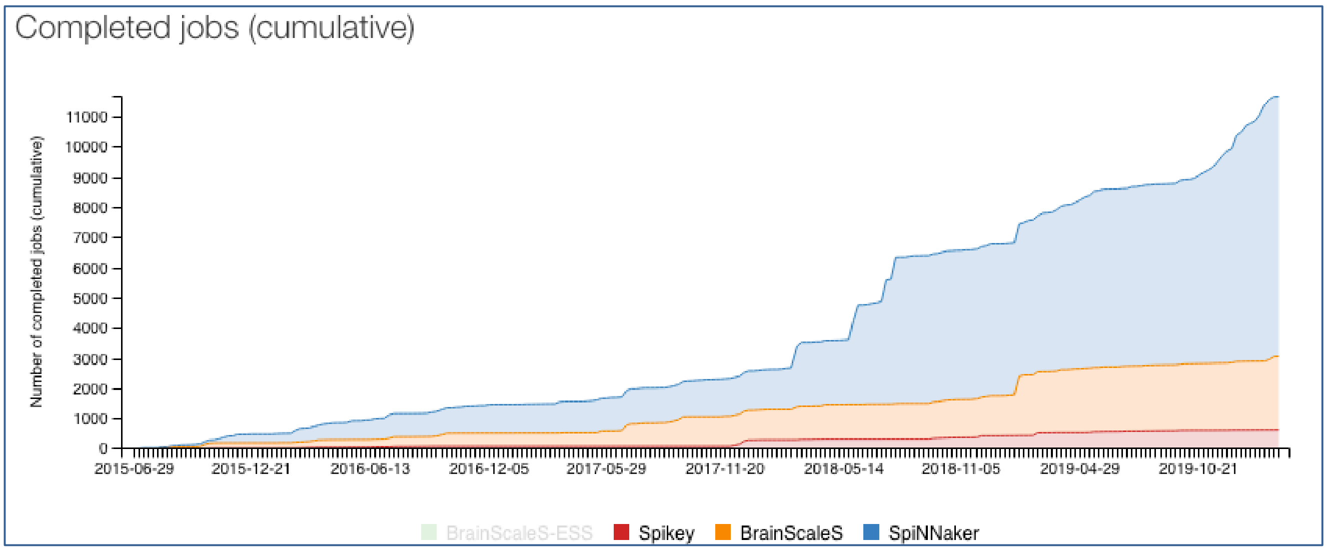


Figure 17: Cumulative NMC job count via NMPI - from the [NMPI dashboard]

Both systems are accessible via Jupyter Notebooks from the HBP Collaboratory, using an NMPI web-front end.

In addition to this "low barrier to entry" access, for "power" access, there is also faster direct access to the machines. Most jobs on the BrainScaleS-1 were performed using direct access.

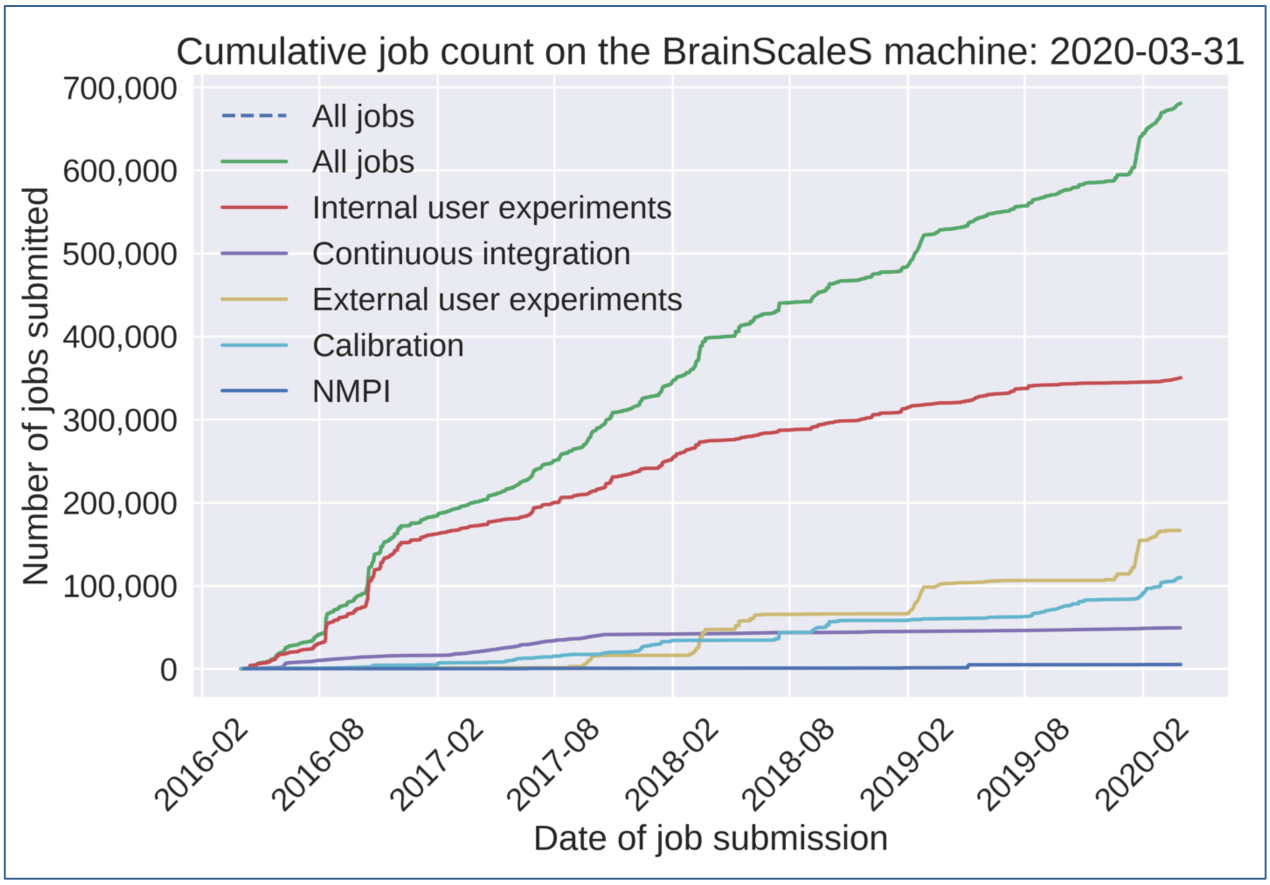


Figure 18: BrainScaleS job count - all jobs, including NMPI submitted jobs

Component	Link to	URL
C0001 — BrainScaleS-1	Technical Documentation	https://flagship.kip.uni-heidelberg.de/jss/FileExchange/D9.7.1_Neuromorphic_Platform_Specification_-_public_version.pdf?fID=1887&s=qqdXDg6HuX3&uID=65
	User Documentation	http://electronicvisions.github.io/hbp-sp9-guidebook/

Component	Link to	URL
C0002 - SpiNNaker-1	Technical Documentation	https://spinnakermanchester.github.io/docs/
	User Documentation	https://spinnakermanchester.github.io/

9.1.4 Output 3: Embedded next generation wafers

A mechanical demonstrator for embedding next generation wafers into PCBs has started to be fabricated. The wafer has a diameter of 300mm and one redistribution layer with daisy chain segments of 50 µm line width and 150 µm via landing pads. In total, there are around 300,000 vias between wafer and redistribution layer. The wafer is fully covered with such chain segments, which are deposited by semi-additive copper technology. The fabrication of this demonstrator has progressed to wafer fabrication, which included the thinning and polishing of the wafers to a final thickness of 250 µm. Remaining tasks to be completed are the embedding of the wafers into the PCBs, via formation and routing at PCB level, as well as electrical measurements.

9.1.5 Output 4: NEAL Components and Agents

The NEAL project has developed Python PyNN classes for neural components that can be run in both Nest and SpiNNaker. Separate systems have been developed to run on BrainScaleS.

The Python codes are available on http://www.cwa.mdx.ac.uk/NEAL/neal.html. Each runs in NEST and SpiNNaker software. Automated tests are included with the software. There are tutorials for all six components, and for the finite state automata pattern. The two patterns are the finite state automata (FSA) and the timer. The six components, which make use of the patterns, are: natural language parsing, planning, associative memory, natural language generation, rule based system, and cognitive mapping.

The components can be readily customised. Cognitive mapping, associative memory, natural language generation and planning can be specialised with simple python code. Generation and associative memory can be specialised with text files specify the language or memory structure; interaction can be specified with simple python code.

The components and patterns for BrainScaleS are the associative memory, finite state automata, parser, and timer.

The SpiNNaker and NEST code is from the same code base, with a switch at the beginning of each particular run to select the back end. While systems may vary in some firing details between the SpiNNaker and NEST runs, overall higher-level emergent behaviour is the same. Extensive tests are provided to assure that the systems run as intended.

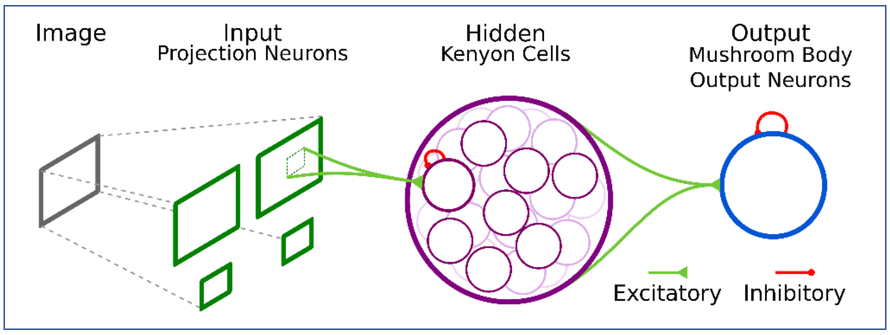
Tutorials are provided for the FSA, parser, generator, associative memory, cognitive maps and planner. Several of these tutorials have agents associated with them, though these agents run in a stand-alone environment instead of the NRP. Agents in the NRP have been developed, but the volatility of the development of the environment has meant that these agents do not always run in the current NRP.

Component	Link to	URL
C3158 - Neural Rule Based System	Software Repository	www.cwa.mdx.ac.uk/NEAL/neal.html
	Technical Documentation	
	User Documentation	www.cwa.mdx.ac.uk/NEAL/neal.html

9.1.6 *Output 5: L2L system for few shot object recognition*

Standard artificial neural networks have achieved superhuman accuracy for object recognition; however, they usually require a large number of labelled examples and long sessions of supervised training. On the contrary, in animals (even insects), learning is unlikely to be supervised in the same way. Typically, animals require fewer samples to learn to recognise a concept.

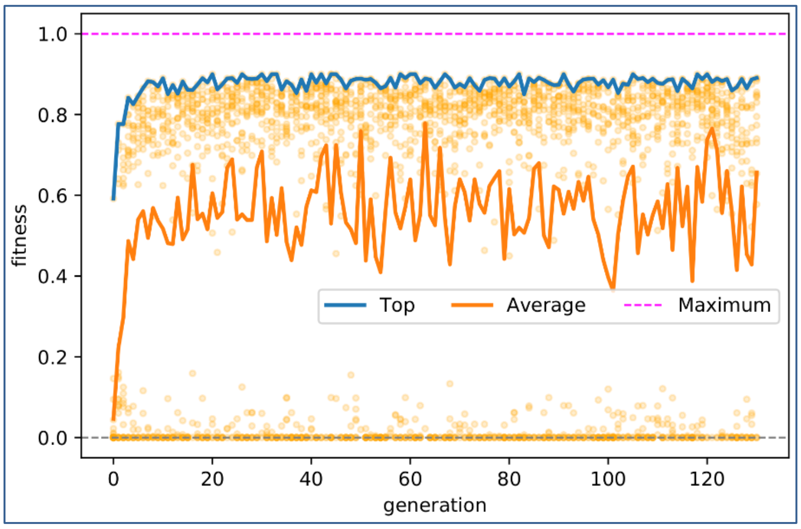
We have investigated a spiking neural network with architecture inspired by insects' mushroom body (MB) to solve a few-shot recognition problem. The input to our model are characters from the Omniglot dataset, which are transformed into rank-order coded spikes by a bio-inspired algorithm.



The model consists of a two-layered spiking network. The hidden layer further expands the input activity into a hyper-dimensional space to ease the output layer’s classification task; furthermore, neurons in the hidden layer sample the input with a distant-dependent connectivity rule. The output layer activity is used to recognise the input, unique recognition is achieved by changing synaptic efficacy, via a simple spike-time-dependent plasticity rule and competition, with a soft-winner-takes-all circuit.

Animals have the advantage of millions of years of evolutionary refinement of their recognition mechanisms. Here, we tune the hyper-parameters of our model (e.g. parameters for the network architecture and the learning rule) with a learning-to-learn methodology inspired by evolution.

We are currently investigating meta-learning algorithms and parameters which best suit our problem. For example, a network recognising a single alphabet of Omniglot and fine-tuned by a genetic algorithm. In this experiment, the normalised average fitness rapidly grows as generations pass (see figure below).



The current model has been tested to run on both the SpiNNaker neuromorphic system and GPUs (through GeNN). Furthermore, we have implemented the model so that it can also utilize the greater resources provided by the Juwels computer cluster (both GPU or CPU units).

The model was deliberately built with elements that are compatible with the BrainScaleS I and BrainScaleS II platforms. While the current research was not performed on these platforms, the models in the future could make use of the 10,000x (or 1,000x for BrainScaleS II) speed-up for the costly neural simulation during the learning-to-learn phase.

In the future, we also plan to exploit the temporal nature of the input spikes by developing a more complex learning rule.

Component	Link to	URL
C3157	Data Repository	https://github.com/chanokin/convert_to_roc
	Technical Documentation	
	User Documentation	https://github.com/chanokin/convert_to_roc
C2631 - Neuromorphic network model for object recognition	Model Repository	https://github.com/chanokin/l2l-omniglot
	Technical Documentation	
	User Documentation	https://github.com/chanokin/l2l-omniglot

9.1.7 Output 6 - Event-based electronic olfaction with SpiNNaker

An event-based, electronic gas sensing system has been developed and implemented using metal-oxide sensors and absolute-deadband sampling and event generation. Various sensors implement these principles for vision (DVS, ATIS) and audition (DAS), but so far there is no equivalent for olfaction. We set out to find whether the same principles could be applied to gas sensing with metal-oxide sensors, and if they could potentially assist in mitigating known issues such as slow responses and baseline drift. To demonstrate the capabilities of the event-based approach to gas sensing, we developed implemented a spiking network that can resolve time delays between events generated from sensors 5cm apart, when exposed to pulses of odorants in turbulent airflow. The network enabled an estimation of the direction of the odour source.

The network was implemented on SpiNNaker (SPiNN-3) and deployed on an omnidirectional robot together with two sensor units. In source localisation experiments, that robot was capable of locating and navigating to a pulsed odour source.

Future work in SGA3 will focus on integrating this olfactory front-end into a spiking cognitive architecture via subcortical relay networks. A real-time capable implementation on SpiNNaker will enable olfactory and, when combined with e.g. visual input, multimodal navigation, learning and inference in stationary and robotic sensing systems.

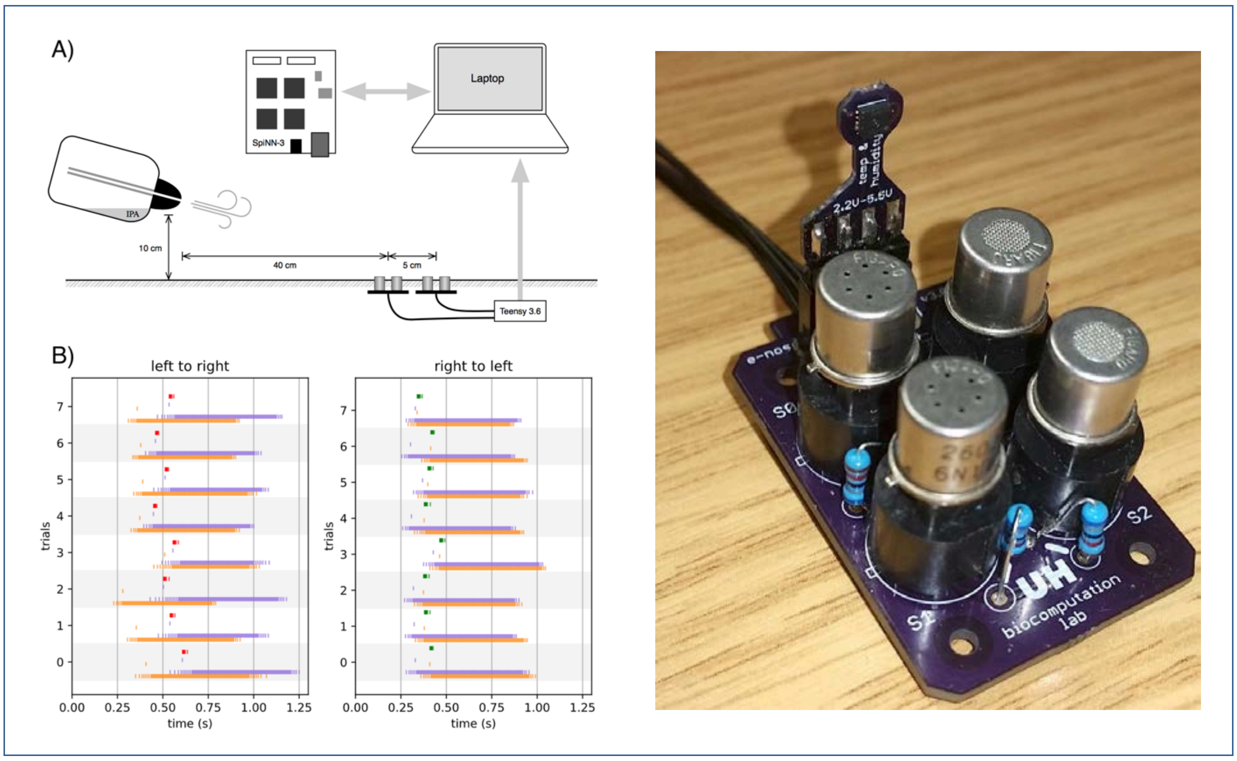


Figure 19: Electronic olfaction

A) The test setup. B) Primary sensor spike trains (yellow/purple) and direction detections (green/red) in real-time experiments. Right image: the gas sensor.

9.1.8 *Output 7: Additional theory advances*

9.1.8.1 Topic 1: A biologically plausible and hardware friendly approximation to optimal gradient descent learning of recurrent SNNs: e-prop

This new learning method for recurrent SNNs promises to be substantially more powerful than previous methods. In fact, it enables SNNs trained by e-prop to approximate for several difficult benchmark tasks from current ML/AI the performance of LSTM networks trained by BPTT. But in contrast to the latter, it can be applied to highly energy-efficient spike-based hardware. Furthermore, it provides competitive performance on hard benchmark tasks such as TIMIT (phoneme recognition). A journal version of the preprint Bellec *et al*., 2019, P1198 is currently under review. The code for e-prop will be made public in March 2020, along with a revision of the paper according to the reviewer recommendations. E-prop is currently implemented on SpiNNaker by the University of Manchester.

9.1.8.2 Topic 2: Deep Reinforcement Learning (Deep RL) with recurrent SNNs.

TU Graz showed in Bellec *et al.* (2019, P1198), that e-prop provides, for the first time, a powerful online Deep RL method that can be implemented on spike-based neuromorphic hardware. This enables spike-based neuromorphic hardware to approach the performance level of state-of-the-art ANNs in AI.

An RSNN can learn with this method to win Atari games, just from observing raw pixel input and getting rewards while playing the game, see Figure 19.

9.1.8.3 Topic 3: Efficient ANN-to-SNN conversion for state-of-the-art image classification with few spikes per image

In order to achieve competitive performance by SNNs on hard image classification benchmark tasks, such as CIFAR 10 or Imagenet, it appears to be necessary to first train ANNs and then port the solution to SNNs. However, the resulting spike-based network only runs in an energy-efficient mode if it uses only a few spikes per image, rather than using rate coding of analogue values. Two new methods for ANN-to-SNN conversion that yield sparsely active SNNs have been reported in Stöckl & Maass, 2019 (P2406) and Stöckl & Maass, 2020 (P2407).

9.1.8.4 Topic 4: A theory of brain computation as basis for the design of more powerful SNN architectures

A model for the online formation of associations between assemblies of "concept cells" has been accepted for publication by a very selective neuroscience journal: Pokorny *et al.*, 2019 (P2279). This model has provided a basis for a new theory of brain computation, which has been presented in the invited talk by Christos Papadimitriou (Columbia University) at the HBP 2020 Summit. This theory was described in detail in the paper Papadimitriou, C.H., Vempala, S.S., Mitropolsky, D., Collins, M., Maass, W. (2020) Brain computation by assemblies of neurons, which is currently under review by the prestigious journal PNAS (Proc. of the National Academy of Sciences of the USA) (preprint DOI 10.1101/869156, P2408).

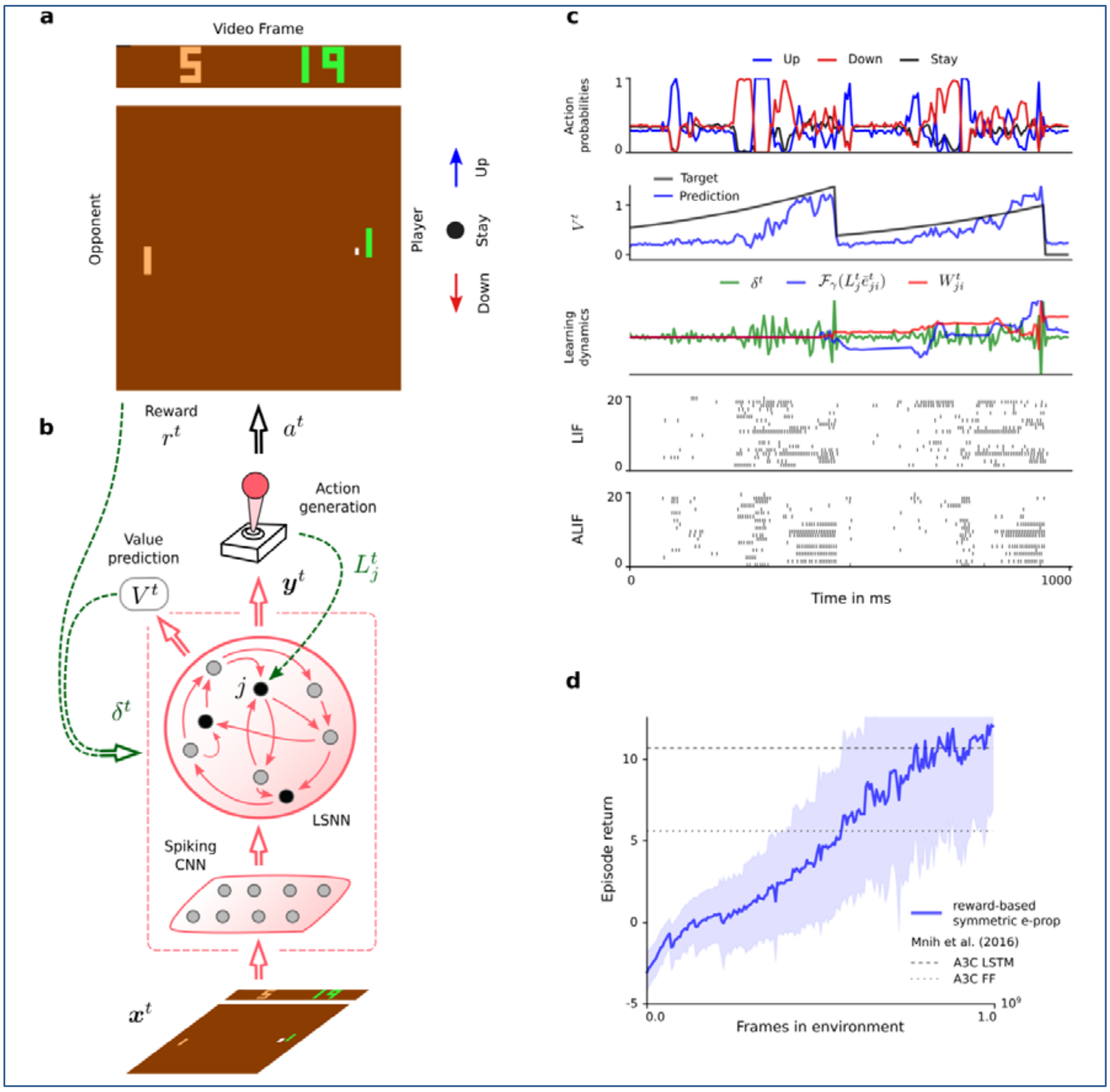


Figure 20: Application of e-prop to the Atari game Pong

a) Here the player (green paddle) has to outplay the opponent (light brown). A reward is acquired when the opponent cannot bounce back the ball (a small white square). To achieve this, the agent has to learn to hit the ball also with the edges of his paddle, which causes a less predictable trajectory. b) The agent is realized by an LSNN. The pixels of the current video frame of the game are provided as input.. c) Sample trial of the LSNN after learning with reward-based e-prop. From top to bottom: probabilities of stochastic actions, prediction of future rewards, learning dynamics of a random synapse (arbitrary units), spiking activity of 20 out of 240 sample leaky integrate-and-fire (LIF) neurons and 20 out of 160 sample adaptive LIF neurons (ALIF). d) Learning progress of the LSNN trained with reward-based e-prop, reported as the sum of collected rewards during an episode. The learning curve is averaged over 5 different seeds. Comparison results for LSTM networks with BPTT and feedforward ANNs with backprop from Mnih et al. (2016).

9.1.8.5 Topic 5: New learning methods for spike-based robot control

New methods for online learning of spike-based robot controller were presented in Kaiser *et al.*, 2019 (P1526)

9.2 Validation and Impact

9.2.1 Actual and Potential Use of Output(s)

The running large scale NMC systems are the NMC part of the HBP joint infrastructure for neuroscience. The NMC systems are demonstrated in use-cases.

The presented new theory work (especially e-prop) may become the foundation for a much wider applicability of spiking neural networks.

9.2.2 Publications

- Bogdan Petruț A., Rowley Andrew G. D., Rhodes Oliver, Furber Steve B. “Structural Plasticity on the SpiNNaker Many-Core Neuromorphic System”, Frontiers in Neuroscience, 2018, vol. 12 DOI : 10.3389/fnins.2018.00434 (Output 1, P1460)
- Huyck: A neural cognitive architecture, Cognitive Systems Research, Vol. 59, 2020, DOI 10.1016/j.cogsys.2019.09.023 (Output 4, P2338)
  - o Relevance: Makes use of the rule based and other components as the basis of a prototype of a compete cognitive architecture implemented in spiking neurons
- Michael Hopkins, Garibaldi Pineda-García, Petruţ A. Bogdan and Steve B. Furber. “Spiking neural networks for computer vision”, Interface Focus 8: 20180007. DOI 10.1098/rsfs.2018.0007 (P1459)
- Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2019). A solution to the learning dilemma for recurrent networks of spiking neurons. bioRxiv, 738385, DOI 10.1101/738385, (Output 7, topic 1+2, P1998
- Stöckl, C., & Maass, W. (2019). Recognizing Images with at most one Spike per Neuron. arXiv preprint arXiv:2001.01682. (Output 7, topic 3, P2406)
- Stöckl, C., & Maass, W. (2020). Classifying Images with Few Spikes per Neuron. arXiv preprint arXiv:2002.00860 (Output 7, topic 3, P2407)
- Pokorny, C., Ison, M. J., Rao, A., Legenstein, R., Maass, W., & Papadimitriou, C. (2019). STDP forms associations between memory traces in networks of spiking neurons. Cerebral Cortex, 2019 DOI 10.1093/cercor/bhz140 (Output 7, topic 4, P2279)
- Jacques Kaiser, Michael Hoff, Andreas Konle, J. Camilo Vasquez Tieck, David Kappel, Daniel Reichard, Anand Subramoney, Robert Legenstein, Arne Roennau, Wolfgang Maass & Rüdiger Dillmann (2019) Embodied Synaptic Plasticity With Online Reinforcement Learning 10.3389/fnbot.2019.00081 (Output 7, topic 5, P1526)

10. CDP5 Results: links to KRc5.1, KRc5.2, KRc5.3 and KRc5.4 in Deliverable D9.4.2

SP9 Partners contributed to CDP5 results, as described in the Deliverable D9.4.2 "CDP5 -Biological deep learning: Results for SGA2 Year 2".

11. Conclusion and Outlook

Overall the SP9 work in SGA2 has gone according to plan, and the Neuromorphic Computing Platform is well-placed to contribute to significant brain science outcomes in SGA3. The decision taken during SGA2 not to fund the large-scale 2nd-generation neuromorphic systems in SGA3 required some adjustments to the SGA2 work. The related science work in SGA3 has been adapted, either to work with the large-scale 1st-generation systems, or with small-scale 2nd-generation prototypes, but these adjustments have been accommodated in the SGA3 planning. Overall, the software support and the neuromorphic systems themselves are well-prepared to support the range of uses of the Neuromorphic Computing Platform planned for SGA3.

The two neuromorphic computing systems supported by the HBP continue to offer the only openly-accessible neuromorphic computing resources available anywhere in the world, and continue to represent world-leading capabilities in terms of scale (SpiNNaker) and speed (BrainScaleS), despite the technologies upon which they are built being over ten years old. The 2nd-generation systems employ more up-to-date technologies with far greater commercial potential, but scaling these up is now outside the remit of the HBP. A EUR 8 million grant from the Saxony Science Ministry has been secured to support the scaling-up of SpiNNaker-2 to form SpiNNcloud, a datacentre neuromorphic cloud service for German industry in the Saxony region. TU Dresden and UMAN are discussing setting up a joint company for the wider commercial exploitation of SpiNNaker-2 technology.

In summary, work during the last 12 months of SGA2 has gone to plan, with necessary changes being made in response to external factors. The Neuromorphic Computing Platform is ready for service and fully capable of meeting the expected demands that will be placed upon it by the exciting science tasks planned for SGA3.