Grant Agreement:	604102	Project Title:	Human Brain Project

Document Title:	Applications: Second Science Report
Document Filename	SP11_D11.4.4_2nd_Resubmission_FINAL
Deliverable Number:	D11.4.4
Deliverable Type:	Report
Work Package(s):	WPs 11.1, 11.2, 11.3 (WPs involved in writing this document)
Dissemination Level:	PU = Public
Planned Delivery Date:	Submission: RUP M30 / 31 Mar 2016; Resubmission RUP M36 / 30 Sep 2016
Actual Delivery Date:	Submission: RUP M30 / 30 Mar 2016; Resubmission: RUP M36 / 30 Sep 2016 2nd Resubmission: RUP M40 / 25 Jan 2017

Authors:	Gregory FRANCIS (P1), Ferath KHERIF (P23), Karlheinz MEIER (P45)
Compiling Editors:	Karlheinz MEIER (P45), Martina SCHMALHOLZ (P45)
Contributors:	Eduardo ROS (P58), Florian RÖHRBEIN (P53), Ferath KHERIF (P23), Frank GOTTFRIED (P47), Chris HUYCK (P91), Bernabe Linares BARRANCO (P82), Ulrich RÜCKERT (P102), Ryad BENOSMAN (P107), Alan DIAMOND (P112), Michael SCHMUKER (P112) Thomas NOVOTNY (P112)
Coordinator Review:	EPFL (P1): Jeff MULLER, Martin TELEFONT
Editorial Review:	EPFL (P1): Guy WILLIS, Lauren ORWIN, Colin McKINNON

Abstract:	This document summarises the final achievements and open issues of the Applications Subproject in the Human Brain Project. It is structured according to the three Work Packages in Neuroscience, Medicine and Neuromorphic Computing. The Subproject has achieved it's major goals and showcased early application and benchmarking of the HBP Neurorobotics, Medical Informatics, and Neuromorphic Platforms.
Keywords:	Neuroscience, medicine, computing, applications, artificial retina, Neurorobotics, disease classification, neuromorphic, spike based computing.

Table of Contents

1. Introduction 6
1.1 Aim of this Document 6
1.2 Overview of Subproject 11 Achievements 6
WP11.1 Future Neuroscience 6
WP11.2 Future Medicine 6
WP11.3 Future Computing 7
1.3 Overview of Subproject 11 Problems and Corrective Actions 8
WP11.1 Future Neuroscience 8
WP11.2 Future Medicine 8
WP11.3 Future Computing 8
2. Future Neuroscience (WP11.1) 9
2.1 Description and validation of the visual (retina) processing models 9
2.2 Experimental set-up of simulation environment 9
2.3 Implementation status with the Neurorobotics Platform 10
2.4 Description of major use case(s) and target users 10
2.5 Documentation of models and experimental set-up 11
2.6 Outreach 11
3. Future Medicine (WP11.2) 13
3.1 Model description and validation for dementia classification 13
3.2 Predictive models of major dementia 15
3.2.1 Informatics-based Model: Unsupervised Rule-based Clustering 15
3.2.2 Informatics-based Model: Enriched Automated Diagnostic Tools 17
3.2.3 Informatics-based Model: Deep Learning for Automated Feature Extractions 19
3.2.4 Informatics-based Model: Rasch model and factor analysis for learning disease severity 22
3.2.5 Informatics-based Model: Bi-clustering applied to gene expression and brain volumetric data 24
3.2.6 Informatics-based Model: Bayesian Causal Model 26
3.3 Implementation status of algorithms, classifiers and models with the Medical Informatics Platform (MIP) 31
3.4 Description of major use case(s) and target users 33
3.5 Documentation of models and related tools/applications 34
3.6 Outreach 34
4. Future Computing (WP11.3) 36
4.1 Neuromorphic computing applications: state-of-the art, contributions and value of WP11.3 and future development in the HBP 36
4.2 T11.3.1 (SAP): Neuromorphic Data Mining Systems 39
4.2.1 Description and validation of software models for network architectures in neuromorphic hardware 42
4.2.2 Implementation of network architectures in the NCP and defined benchmarks 43
4.2.3 Description of major use case(s) and target users of your application 44
4.2.4 Documentation of software models and network architecture 48
4.2.5 Outreach 48
4.3 T11.3.2 (MU): Port CABot3 to neuromorphic chips and extend 48
4.3.1 Describe and validate neural and synaptic models for neuromorphic systems 48
4.3.2 Implementation and performance of virtual agent and environment on neuromorphic systems 49
4.3.3 Description of major use case(s) and target users of your application 51
4.3.4 Documentation of models and related experiments 52
4.3.5 Outreach 52
4.4 T11.3.3 (CSIC): Exploitation of Feedback in Ultra-fast Spiking Visual Architectures 53

4.4.1 Description and validation of the neuron models for symbol recognition 53
4.4.2 Implementation of benchmark architecture on the NCP 58
4.4.3 Description of major use case(s) and target users of your application 62
4.4.4 Documentation of models and related experiment set-up 62
4.4.5 Outreach 63
4.5 T11.3.4 (UNIBI): Spiking Associative Networks for Neuromorphic Computing Systems 63
4.5.1 Implementation of benchmark architecture on the NCP 64
4.5.2 Description and usage of implemented SAM models to compare different execution environments (benchmark development) 64
4.5.3 Description of major use case(s) and target users of your application/tools 65
4.5.4 Documentation of models (and tools) with related experiment set-up 65
4.5.5 Outreach 66
4.6 T11.3.5 (UMPC): Asynchronous Computational Retina 66
4.6.1 Hardware implementation: Interface to connect one ATIS camera into SpiNNaker (Milestone 11.5.3.1) 66
4.6.2 Hardware implementation: Interface to connect two ATIS cameras into SpiNNaker (Milestone 11.3.5.2) 68
4.6.3 Hardware implementation : stimulation platform (Milestone 11.5.3.3) 70
4.6.4 Generate datasets for tests and evaluation of computational models (Milestone 11.3.5.4) 70
4.6.5 Computational model: Visual motion (Milestone 11.3.5.5) 71
4.6.6 Computational model: Stereovision (Milestone 11.3.5.7) 76
4.7 T11.3.6 (UOS): Implementing a Spiking Classifier Network on HiCANN 84
4.7.1 Description of data sets, neuron models (or counts) and classifier in experimental set-up 84
4.7.2 Implementation of classifier architecture on Neuromorphic Computing Platform and evaluated parameters 85
4.7.3 Description of major use case(s) and target users of your application 85
4.7.4 Documentation of experimental set-up 85
4.7.5 Outreach 85
Annex A: Overview Component-Task linkage SP11_RUP to SGA1 86
Annex B: References 88

List of Figures and Tables

Figure 1: Integration of the retina model into NRP 10
Table 1: List of models developed and data sets used 13
Table 2: Functions and KPI statuses for WP11.2 14
Figure 2: Anatomical label of the six rules, and proportion of data space explained by each rule 16
Figure 3: Brain regions contributing to each set of rules 16
Figure 4: Results of the study 18
Figure 5: Survival analyses 19
Figure 6: Feature extraction for a stacked auto-encoder containing two hidden layers 21
Figure 7: Results for classification based on random patches used as input features 22
Figure 8: The probability density distribution of the three clinical groups 23
Figure 9: The regions highly weighted in estimating disease severity 24
Figure 10: General 'Divide and Recombine' parallel paradigm under HBP context. 27
Figure 11: Divide and Recombine parallel paradigm for general linear model under HBP context. 29
Figure 12: Divide and Recombine streaming paradigm for general linear model under HBP context. 30
Figure 13: Implementation of algorithms, classifiers and models within the MIP 32
Table 3: Major use cases and target users for the MIP 33
Figure 14: Minimalistic STDP memory network as described in the text. 40
Figure 15: Simulation of a single memory network. 41
This figure shows the plotted weight of the single STDP synapse for 8 different training scenarios. An iteration lasts for 100 ms. Until 5900 ms or 59 iterations, a reference signal is presented for training. From 6000 ms onwards, the system is in a "retrieve phase". 41
Figure 16: Blockwise composition of three memory networks denoted as mn1 to mn3. 41
Figure 17: Preliminary results of the memory network implemented on the Spikey chip. Shown is the "retrieve phase" for a training scenario leading to a stable state after a "burn-in" time. 42
Figure 18: Proposed spiking CMAC network 42
Figure 19: Single prediction simulation of the proposed temporal CMAC. 43
Figure 20: Simulation of a predictive maintenance scenario. 44
Figure 21: Illustration of CPUs with attached memory modules and interconnects 45
Figure 22: Distribution of minimum and maximum query response times across 9 different workload traces (WL100-WL112). 46
Table 4: Status of use cases for the NCP 48
Table 5: Status of model types and networks in WP11.3.2 48
Figure 23: Gross Architecture of Agent. Boxes represent subsystems instantated in simulated neurons. Arrows represent synaptic connections between subsystems. 50

Figure 24: Top down view of the path of agent through the virtual environment while learning the cognitive map. 51
Table 6: Status of feedback exploitation in ultra-fast spiking visual architectures 53
Figure 25 (a)Typical state evolution and spike production sequence for a signed spiking neuron with leak and refractory period 53
Figure 25: (b) Detail of compact ConvNet implementation on SpiNNaker. 54
Figure 26: Experimental set-up. 56
Figure 27: Experimental set-up of the system. 58
Figure 28: Recognition capability in different slow rates. 58
Figure 29: Optimization process time and number of iteration results. 60
Figure 30: V_threshold parameter mismatch by layers and compared to all layers. 61
Figure 31: Effect of refractory time mismatch. 61
Figure 32: Effect of kernel mismatch with the increase of RC. 62
Figure 33: Poker card convolutional neural network. 63
Table 7: Status of spiking associative network models for neuromorphic computing systems 63
Table 8: Internal milestones for T11.3.5 66
Figure 34: Interface board used to feed data from an ATIS camera to SpiNNaker 67
Figure 35: SpiNNaker packet structure and event to MC packet mapping. 68
Figure 36: Designed interface board plugged between an ATIS camera and the SpiNNaker system. 68
Figure 37: Designed interface boards plugged in two ATIS cameras feeding data into the SpiNNaker system. 69
Figure 38: SpinnakerSpiNN5 hardware description and FPGAs links. 69
Figure 39: XY platform - 1a) & 1b) Y axis. 2) X axis. 3) Eink display. 4) Electronics. 70
As the maximal frequency for the stepper drivers was 200kHz, the possible speeds were :70
Figure 40: Capture of a moving bar in the database. 71
Figure 41: Number of neurons needed for a simulation versus the number of pixel 72
Figure 42: Simplified principle 72
Figure 43: On a moving bar 73
Figure 44: Moving bar and optical flow 73
Figure 45: CounterClockwise rotation of a pen, speed and direction are encoded among the HSV map. 74
Figure 46: Normalized membrane potential responses for (a) OnBeta Cells, (b) OffBeta Cells, (c) OnDelta cells for an input spike at t = 200ms for different weights. 75
Figure 47: 76
Figure 48: Model used to detect sub-millisecond spike synchrony for sound localisation. 77
Figure 49: Input and output plots 78
Figure 50: Stimulation starts at t = 100 ms, and spans the whole length of an epipolar line (64 pixels). 79

Figure 51: Raster plot of the neurons in the disparity plane..... 79
Figure 52: Disparity computed as the difference between the coordinate of each epipolar line as represented in the disparity plane..... 80
Figure 53: Raster plot of the epipolar line populations..... 81
Figure 54: Raster plot of the neurons in the disparity plane..... 81
Figure 55: Disparity computed as the difference between the coordinate of each epipolar line as represented in the disparity plane..... 82
Figure 56: Live demo of disparity computation from the input from two ATIS. 83
Figure 57: Live demo of disparity computation from the input from two ATIS. 83
Table 9: Data sets, neuron models, classifiers and implementation status in T11.3.6 84

1. Introduction

1.1 Aim of this Document

This document reports on the results achieved by Subproject 11 (SP11) of the Human Brain Project (HBP) between Ramp-Up Phase Months 18–30, including published and submitted journal papers.

1.2 Overview of Subproject 11 Achievements

Subproject 11 was created to provide project internal guidance for the Information and Communications Technology (ICT) Platforms before their completion at the end of the Ramp-Up Phase. For this purpose, the three Work Packages of SP11 have been tightly integrated with Subprojects 8, 9 and 10. At the same time, the work in SP11 provided a first glimpse of possible future applications for the Platforms.

Major parts of the scientific, technical and computing work (application tasks and related Platforms) done in the Ramp-Up Phase can be linked to tasks in SGA1. The high-level overview table in Appendix A demonstrates this.

The work of SP11 has been successful and is reported in this document. In the following, we provide a short list of achievements ordered by Work Package.

WP11.1 Future Neuroscience

Retinal and cortical models of visual processing were developed and validated as planned. They are being put to use as input for other systems (e.g. object tracking) and as a way of explaining psychophysical data (e.g., visual crowding).

WP11.1 also connected researchers in SP11 and the developers of the Neurorobotics Platform (NRP) in WP10. They successfully integrated the retinal model into the NRP.

WP11.2 Future Medicine

The work on model description and validation for dementia classification shows that there is more than one pattern of regional brain pathology characterised by an Alzheimer phenotype. The regions contributing to each set of rules or subgroups are very specific.

The development of enriched automated diagnostic tools demonstrated that individuals with mismatched labels showed intermediate characteristics in both anatomy patterns and

memory performance. This characterises different mechanisms related to Alzheimer's disease (AD).

Deep learning for automated feature extractions showed that low-level features extracted from the scans without any preliminary knowledge, combined with a fairly easy deep learning scheme, has promising classification potential.

Studies of the informatics-based model and factor analysis for learning disease severity led to the conclusion that clinically diagnosed groups have significantly different distributions of severity, but with a big overlap between them. This is expected due to the blurry clinical diagnostic criteria.

Bi-clustering applied to gene expression and brain volumetric data showed that the co-module genes were enriched for expressions in various regions of the brain.

The status of algorithms, classifiers and models being implemented with the Medical Informatics Platform (MIP) is described in detail in the main body of this document.

WP11.3 Future Computing

The development of neuromorphic data mining systems was successfully studied in two major use cases: Predictive maintenance and the application of neuromorphic compatible classifiers to non-uniform memory access (NUMA) scheduling in a combined online transaction processing (OLTP)/online analytical processing (OLAP) In-Memory Database.

Porting the Cell Assembly Robot (*CABot*), a video game agent, to neuromorphic chips has been carried out for two major uses cases: the SpiNNaker agent; and, a cognitive model of categorisation model. The agent, though simple, works. The cognitive model is being published and categorises like the brain's putative explicit system.

The exploitation of feedback in ultra-fast spiking visual architectures has been implemented as an event-driven Convolutional Neural Network architecture with optimized parameters to perform recognition of the card that has been programmed on a 4-chip Spinnaker board.

The construction of spiking associative networks for neuromorphic computing systems led to the development of tools (spiking associative memory (SAM) generation, benchmark generation, PyNNLess, performance analysis, parameter optimization) that are now open to all HBP users. For the neuromorphic researchers these tools simplify the hardware access and systematic hardware testing. These tools are especially helpful for the design of the next generation of neuromorphic hardware within HBP.

Asynchronous computational retina work developed a pure, event-driven visual computational approach that uses precise timing mechanisms to design new computational techniques in visual processing. The task produced a full event-driven visual processing system linking a neuromorphic retina directly to the SpiNNaker system by an Asynchronous Event Representation (AER) bus. The architecture allowed the first real-time development and implementation of new, visual, event-driven computation techniques.

The implementation of spiking classifier networks was based on previous work carried out on the Heidelberg spiking neural network chip *Spikey*. The classifier design has been modified and adapted for implementation on different neuromorphic platforms. The researchers analysed how each implementation performs differently with varying model sizes (up to 30,000 neurons, 18 million synapses), with different number of classes (2-10 digits), and when incorporating increasing numbers of "virtual receptors" (VRs) in input space (20 – 500+). The second part of the work focused on using a large model on higher capacity "large-scale" hardware.

The current report on future computing also contains an introductory section discussing state-of the-art of neuromorphic computing, the contributions of SP11 to the field and the future developments planned in the HBP.

1.3 Overview of Subproject 11 Problems and Corrective Actions

In the following section we discuss the problems encountered during the project work and the plans for a continuation of aspects of SP11 in the HPB SGA1 phase.

WP11.1 Future Neuroscience

Although the retinal and cortical models were successfully integrated, this integration did not happen soon enough to fully explore all of the intended topics related to psychophysical data. Such investigations are on-going. In particular, we continue to explore Weber's Law for brightness perception, Bloch's law for brightness perception, and visual afterimages.

Although the retinal model was successfully integrated into the NRP, a similar effort to integrate the cortical model revealed that the current computational resources are unable to simultaneously generate the virtual environment and (a reduced version of) the cortical model. A way of parallelizing the computations is being explored.

The work will be transferred to the SGA1 phase of the HBP.

WP11.2 Future Medicine

No major problems have been encountered during the work of WP11.2. All Tasks will be continued in the first SGA phase of the HBP.

WP11.3 Future Computing

Due to the parallel development of neuromorphic chips and their applications only a few experiments could be executed on real hardware. The availability of software models and executable software specifications helped to overcome this problem. The communication between WP11.3 and SP9 (Neuromorphic Computing) was excellent due to the tight integration of the two activities with joint regular meetings and close collaboration of individual groups. It is worth noting that with one exception (SAP), the WP11.3 groups were selected through an open call and could only start work 6 months into the project.

During the preparation of the SGA1 proposal it was decided to continue Tasks 11.3.2, 11.3.4 and 11.3.6 in the SGA1 phase as they represent challenging use cases for the HBP Neuromorphic Computing Platform (NCP) hardware and will act at the same time as benchmarks for the next generation neuromorphic machines.

2. Future Neuroscience (WP11.1)

This report describes the results achieved by the end of the 30-month Ramp-Up Phase for Tasks 11.1.1 (Psychophysics of perception: the Weber-Fechner law) and 11.1.2 (Integrated brain-body control benchmarks).

Retinal and cortical models of visual processing were developed and validated, and the retinal model was successfully integrated into the NRP.

2.1 Description and validation of the visual (retina) processing models

The retinal model was validated as part of previous milestones (MS200 and MS201) against neurophysiological and behavioural measures[1,2]. In particular, the retinal model processing produces representations of light intensity that correspond to Weber's Law.

A technical validation of the retinal model has been completed that corresponds to milestone MS202: integration of the retinal model with the NRP. This milestone was satisfied later than the planned Month 18 date because the NRP was not available until Month 18.

As part of previous milestones (MS200 and MS201) the cortical model has also been validated relative to neurophysiological and behavioural measures[3]. The validation process of such a model never really ends, but the model behaves properly for the target topics (e.g., the dynamic rate of information processing in the model is realistic, the model produces "illusory contours", and the model explains a wide variety of masking effects).

The retinal and cortical models have also been successfully integrated.

2.2 Experimental set-up of simulation environment

The virtual environment in the NRP uses a small humanoid type of robot in a room with a textured floor, panelled walls, and two large screens on opposite walls. The content on the screens can be changed according to experimental plans. The current plan is to feed the retinal output into a system developed by Egidio FALOTICO from SP10 as part of a visual tracking system. Figure 1 is from a recorded movie of the NRP and the retinal processing model. The virtual environment is shown in the bottom window.

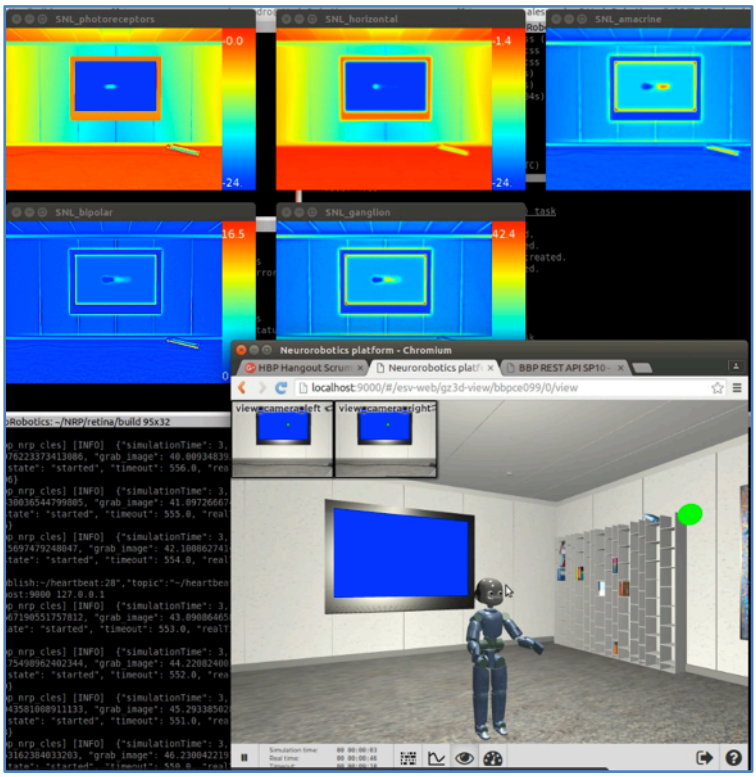

Figure 1: Integration of the retina model into NRP

2.3 Implementation status with the Neurorobotics Platform

The retinal model has been integrated with the NRP so that image frames of the virtual environment, from the perspective of a NRP robot, are submitted to the model and processed by the model to produce spike trains of ganglion cells. In Figure 1 the smaller windows display outputs from various stages of the retinal model.

In contrast to the success of the retinal model's integration with the NRP, the cortical model has encountered computing boundaries. The initial attempt to integrate the cortical model in the NRP was done without the retinal model (their integrations were been done in parallel). A small version of the cortical model (50 x 50 pixels, with approximately 162,500 model neurons) was coded in the NEST software and provided to the NRP team. They reported that the computers running the NRP virtual environment could not simultaneously run a model with so many neurons. A request was made to reduce the model size by an order of 10, but this would result in a 5 x 5 pixels input frame, which is too small to be of any practical value in the NRP.

Further discussions with the NRP developers revealed that they are working on ways of separating the computing resources for the NRP and interacting models. Such separation should allow for the cortical model to interact with the NRP virtual environment in the future.

Because the cortical model could not be implemented in the NRP with current computing resources, we did not pursue further efforts to combine the retinal and cortical models in the NRP. In principle, since the cortical and retinal models have been integrated outside of the NRP, it should be possible to do so with the NRP. These problems mean that we have been unable to complete milestone MS203 (First experiment completed) by Month 30.

2.4 Description of major use case(s) and target users

Application of the models to use cases and target users will largely depend on whether the retinal model output is sufficient by itself (as for the on-going experiment on visual tracking) or if the cortical model (with its mechanisms for perceptual grouping) is needed. In the latter situation, the system is not functional until the planned separation between the NRP and model computations is completed.

Target users for the retinal model include scientists interested in utilising the computational processing of the retina. This may include people interested in computer vision, biological processing, and human vision. The retinal output could provide the signals that support further processing.

Target users for the cortical model (perhaps combined with the retinal model) include scientists interested in perceptual grouping, brightness perception, the neurophysiological representation of visuo-spatial information, attention, and figure-ground distinctions.

2.5 Documentation of models and experimental set-up

The source code for the retinal model, documentation for its use, and a video of its integration with the NRP is uploaded to the NRP OwnCloud:

https://neurorobotics-files.net/owncloud/index.php/apps/files/ajax/download.php?dir=%2Fgeneral-files%2Fother_material%2FSP10_videos&files=retina_first_steps_integration.ogv

Source code for the cortical model, a draft manuscript describing its application to cases of visual crowding, and an analysis script is posted at the Open Science Framework. The files can be viewed at:

https://osf.io/4fhxs/?view_only=a1d4cee6f6514b3da9e8bf40c132b085

Details and source code have been provided as a Data Information Card (DIC) in the HBP Collaboratory.

2.6 Outreach

Published manuscripts:

- Francis, G. *Contour erasure and filling-in: Old simulations account for most new observations*. i-Perception, 2015. **6:**116-126. doi:10.1068/i0684.
- Manassi, M., Hermens, F., Francis, G., and Herzog, M.H. (2015). *Release of crowding by pattern completion*. Journal of Vision, 2015. **15:**1-15. doi: 10.1167/15.8.16.
- Clarke, A.M., Herzog, M.H., and Francis, G. *Visual crowding illustrates the inadequacy of local versus global and feedforward versus feedback distinctions in modelling visual perception*. Frontiers in Psychology: Perception Science, 2014. **5:**1193. doi: 10.3389/fpsyg.2014.01193.
- Martínez-Cañada, P., Morillas, C., Romero, S., and Pelayo, F. *Modeling Retina Adaptation with Multiobjective Parameter Fitting*. In: Advances in Computational Intelligence 2015. Springer International Publishing. pp. 175-184.
- Martínez-Cañada, P., Morillas, C., Pino, B., and Pelayo, F. *Towards a Generic Simulation Tool of Retina Models*. In: Artificial Computation in Biology and Medicine 2015. Springer International Publishing. pp. 47-57.
- Martínez-Cañada, P., Morillas, C., Nieves, J.L., Pino, B., and Pelayo, F. *First Stage of a Human Visual System Simulator: The Retina*. In: Computational Color Imaging 2015. Springer International Publishing. pp. 118-127.

Manuscript under revision or preparation:

- Francis, G., Manassi, M., and Herzog, M.H. *Neural dynamics of grouping and segmentation explain properties of visual crowding*.
- Martinez-Cañada, P., Morillas, C., Pino, B., Ros, E., and Pelayo, F. *A Computational Framework for Realistic Retina Modeling*. IJNS.

Conference presentations, colloquia:

- Francis, G. A small part of the Human Brain Project: *Neural dynamics of visual segmentation*. Forty-first Annual Interdisciplinary Conference. Breckenridge, CO, USA, 31 January – 5 February 2016.
- Francis, G. Cortical *Dynamics of Perceptual Grouping and Segmentation: Crowding*. Psychonomic Society 56th Annual Meeting. Chicago, IL, USA, 21 November 2015.
- Francis, G. *A small part of the Human Brain Project: Neural dynamics of segmentation and crowding*. Purdue University, West Lafayette, IN, USA, 31 August 2015.
- Francis, G. *Cortical dynamics of perceptual grouping and segmentation: Crowding*. Models of cortical networks and function: A farewell symposium for Prof. Gregory Francis. Geneva, Switzerland, 7 July 2015.
- Francis, G. *Cortical dynamics of perceptual grouping and segmentation: Crowding*. Ecole Polytechnique Federale de Lausanne, Switzerland, 7 May 2015.
- Francis, G. *Contour erasure and filling-in: Old simulations account for most new observations*. Vision Sciences Society annual meeting. Naples, FL, USA, 15-20 May 2015.
- Herzog, M. H., Manassi, M., Hermens, F. & Francis, G. *Crowding, patterns, and recurrent processing*. Vision Sciences Society annual meeting. Naples, FL, USA, 15-20 May 2015.
- Francis, G. *A NEST-based simulation of visual processing*. First NEST User Workshop. Geneva, Switzerland, 20-22 April 2015.
- Francis, G. *Thoughts from a cognitive psychologist. "Are we building the right thing? - Requirements from theory for simulation environments and neuromorphic computing."* European Institute for Theoretical Neuroscience, Paris, France, 2-4 March 2015.
- Francis, G. *A cortical model explains human responses to doubly illusory contours*. Human Brain Project Summit. Heidelberg, Germany, September 2014.
- Francis, G. *Specific learning effects from non-specific mechanisms in visual perception*. Perceptual Learning Workshop, Jongny, Switzerland, August 2014
- Herzog, M. (2015). *A fresh, old view on Vision*. Talk, Beijing, China, 8 December 2015.
- Herzog, M. (2015). *A fresh, old view on Vision*. Talk, Paris, France, 14 December 2015.

3. Future Medicine (WP11.2)

This report describes the results achieved by the end of the 30-month Ramp-Up Phase for Task 11.2.1 (Biological signatures of diseases).

Model description and validation for dementia classification shows that there is more than one pattern of regional brain pathology characterised by an Alzheimer phenotype. Individuals with mismatched labels show intermediate characteristics in both anatomy patterns and memory performance, which characterises different mechanisms related to AD. Low level features extracted from scans, together with a deep learning scheme, has promising classification potential.

The informatics-based model and factor analysis show that clinically diagnosed groups have different, but overlapping, distributions of severity.

Bi-clustering applied to gene expression and brain volumetric data showed that the co-module genes were enriched for expressions in various regions of the brain.

The status of algorithms, classifiers and models being implemented with the Medical Informatics Platform (MIP) is described in detail.

3.1 Model description and validation for dementia classification

The computer-aided diagnosis based on the identification of the biological signatures of AD has proven to be a promising method of early detection, an important condition for treating the disease more effectively. Most of the machine learning tools that have been developed are based on the evaluation of magnetic resonance imaging (MRI) scans with multiple modalities. These are sometimes supplemented with additional information, such as positron emission tomography (PET) scans, or genetic or cerebrospinal fluid values to improve the classification accuracy.

Table 1: List of models developed and data sets used

Data Sets	Classification and clustering	Informatics based models	Algorithms and Benchmarks	Comments
Research data from ADNI and 3C: imaging, genetic and clinical variables	Semi-supervised clustering algorithm	Rule-based classification Six rules were derived for explaining AD	Density-based algorithm compared to the use of state of the art “black box” methods	The results show that low dimension factors can explain both healthy and AD patients
Research data from ADNI and 3C: imaging, genetic and clinical variables	Support vector machine (SVM) classifier trained on the pathology proven data and tested on the previous research data (ADNI)	Automated diagnosis based on neuroimaging data (T1w MRI scans)	The results were compared to clinical diagnosis performed by neurologists (expert knowledge)	The results show that automated classification based on neuroimaging data can identify asymptomatic individuals at risk of dementia
Research data from ADNI and 3C: imaging,	Deep learning algorithm	Automated feature learning	Neural net/stacked auto-encoder	The results show that the algorithm was able to learn the best features for

genetic and clinical variables			Compared atlas based features vs. random based features Compared to clinical label	optimum accuracy
Research data from ADNI and 3C: imaging, genetic and clinical variables	Bi-clustering applied to gene expression and brain volumetric data	Identify co-modules	The analysis of the co-modules is mainly gene-centric. Bgee's TopAnat showed that the co-module genes were enriched for expression in various regions of the brain.	GeneMANIA allowed the construction of a network whose central genes have known roles in brain functions or have previously been implicated in various neurological pathologies, such as AD, Parkinson's disease, or autism.
Research data from ADNI and 3C: imaging, genetic and clinical variables	Rasch model and factor analysis for learning individuals' disease severity	Latent variable identification	The results were compared to clinical diagnosis performed by neurologists (expert knowledge)	The results show that the algorithm was able to learn the disease severity going beyond simple binary classification (patients vs controls)
Research data from ADNI and 3C: imaging, genetic and clinical variable	Bayesian causal model	Bayesian Linear Regression Streaming variational Bayes	The results were compared to data driven methods	The Bayesian Formalism provides us the necessary armamentarium to deal with it and offers general sophisticated ways to extend to other models and managing high dimensional and Multimodal Data

We built several predictive models of dementia by tuning the use of data driven machine learning methods. The list of tools developed are reported in Table 1 and summarised in more detail here below. The models were validated using different cohorts from research studies (ADNI) to population based studies (3 C).

Table 2: Functions and KPI statuses for WP11.2

Function	Function Name	Possible KPI statuses	Current KPI status	Target
11.2.1.1	Description format for the biological disease signature	Identify multimodal clinical data Data pre-processing Data aligned Feature selection	All achieved	M3 M6 M9 M12
11.2.1.2	Informatics based model for generating biological disease	Implement test different algorithms Model configuration	All achieved	M12 M18 M18

	signature	Benchmark algorithms Select algorithms		M18
11.2.1.3	Biological signature of major dementia.	Scale up a pilot study to a large scale level Apply algorithm Clinical interpretation Internal validation Model prediction	All achieved	M18
11.2.1.4	Causal mechanisms for major dementias	Define multi-scale description of the data Build an a priori bio-physiological model Compare model behaviour to data Identify the pathways	All achieved	M30

3.2 Predictive models of major dementia

3.2.1 Informatics-based Model: Unsupervised Rule-based Clustering

3.2.1.1 Objectives

There is a great amount of uncertainty regarding the accuracy of diagnostic classification in the early stages of AD. This is due to the underlying heterogeneity in etiologies leading to similar phenotypes. To explain the observed heterogeneity, we use a rule-based clustering algorithm, and identify homogeneous subgroups of patients. The hypothesis is that such subgroups have the same underlying causes.

3.2.1.2 Methods

We used high-resolution T1-weighted 3D data from the Alzheimer's Disease Neuroimaging Initiative (ADNI) datasets, which included 66 healthy controls and 48 AD patients. Participants were matched for age and gender. Firstly, the data were normalised to a common template, using new segmentation and “Diffeomorphic Anatomical Registration using Exponentiated Lie algebra” (DARTEL) processing in SPM8. This allowed us to extract measures of Grey Matter volume (GMv) from each voxel. Next, we summarised the data into regions of interest, based on the Automated Anatomical Labeling (AAL) atlas (Figure 2).

The rule-based algorithm aims to explain the variability between individuals, and describes a population by a group of “local over-densities”. These are defined as subspaces over combinations of variables. The algorithm performs an exhaustive search of the data space to predict the outcome variables; in this case, the health status of each subject in terms of the presence or absence of AD. In our experiment, the predictive variables are the 90 brain region volumes, age, gender, and individual subject global volumes.

3.2.1.3 Results

After convergence and cross-validation tests, Hypercube showed that the data could be explained by six different rules for AD patients, and five rules for healthy controls. Bringing these rules together maximises the difference between healthy controls and AD patients. At a population level, this result shows that there are six ways of presenting with

an Alzheimer phenotype. These six ways correspond to six different sets of regions (see Figures 3 and 4). At an individual level, nonlinear effects are captured by the fact that each participant can be explained by more than one rule (see Table 1 for the proportion of overlap between rules). Critically, in prediction mode, AD and control rules explain 98% of AD patients and 100% of controls.

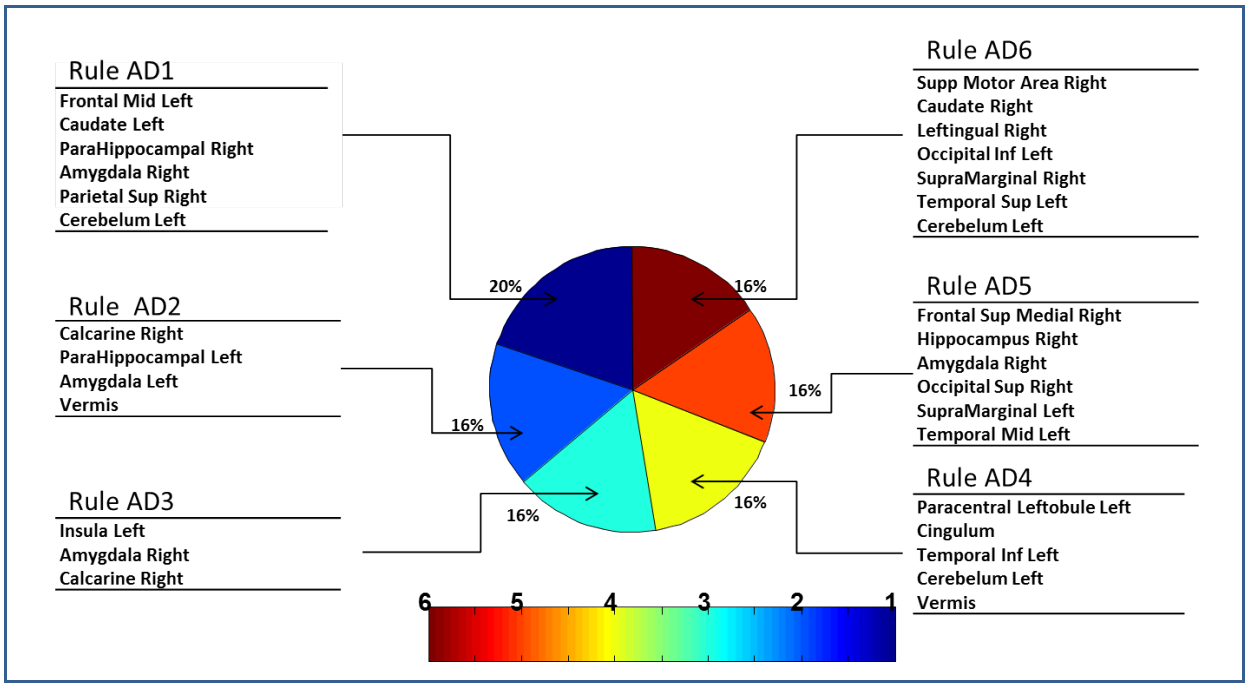

Figure 2: Anatomical label of the six rules, and proportion of data space explained by each rule

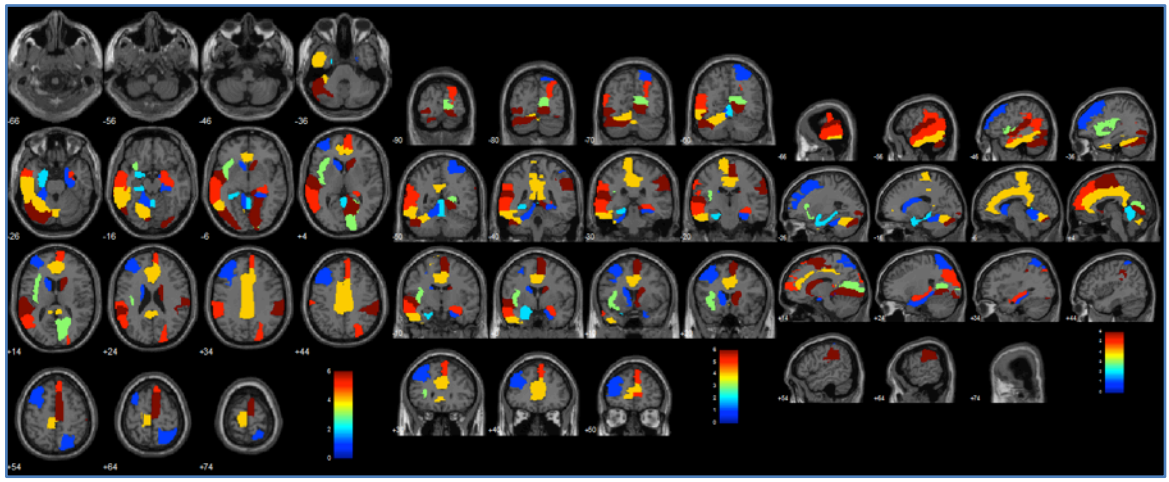

Figure 3: Brain regions contributing to each set of rules

3.2.1.4 Conclusion

The results show that there is more than one pattern of regional brain pathology characterised by an Alzheimer phenotype. The regions contributing to each set of rules or subgroups are very specific. Patients differ from controls according to a very systematic pattern that involves regions known to show atrophy from pathological examination. Specifically, the results showed that: a) the pattern of differences between AD patients and controls involved regions beyond the medial temporal lobes; b) that there is evidence for the existence of several subgroups of AD patients; and c) that these subgroups can be predicted with high accuracy from a low number of deterministic rules.

Given the small number of patients and variables included in the pilot study, we have concluded that it would be very worthwhile to carry out a further study. This would include a much greater number of subjects, and a considerably greater range, diversity, and amount of data describing their states. This follow-up would aim to confirm the preliminary results and specify with greater precision how many sets of rules are needed to identify patients with the AD phenotype. Adding more data may result in fewer rules if some of the patterns identified in our small pilot sample are idiosyncratic and replaced by a more consistent set of factors. The inclusion of new data and auxiliary variables (genetic, cognitive, etc.) would provide a smaller-grained, direct descriptive explanation of the underlying causative pattern identified in this preliminary analysis.

3.2.2 Informatics-based Model: Enriched Automated Diagnostic Tools

3.2.2.1 Objectives

It has been predicted that delaying AD onset by just one to two years would result in 9.5 to 23 million fewer symptomatic and dependent cases by 2050. AD pathology, like that of Parkinson's disease, precedes symptoms. This is demonstrated in the significant redundancy of brain organisation, with a resulting capacity for reorganisation in the face of pathology.[4] In light of this, it is legitimate to propose a strategy of preventive therapy for dementia patients. This would require accurate diagnosis prior to the onset of symptoms, or demonstrable signs and syndromes. Identifying accurate biomarkers, independent of symptoms, is critical to such a strategy.

3.2.2.2 Methods

We built an automated classifier from a set of MRI scans that came from deceased, pathologically diagnosed individuals. This classifier was evaluated for its prognostic value on clinically categorised living people. Subjects were clinically diagnosed as either healthy controls (HC) or with AD, and then grouped by the presence or absence of AD related atrophy into probable AD or HC. Recent evidence suggests that a clinical diagnosis of AD has 70% sensitivity and 44% specificity when patients are followed to autopsy. We compare the clinical diagnosis with one based on an AD-typical pattern of brain atrophy and biomarker profiles (genetic and proteomic), and aim to correlate the results with clinical evidence of subsequent cognitive decline. We tested the idea that cognitively normal people with an AD related brain atrophy pattern were at high risk for conversion to memory impairment and dementia. We evaluated the relative risk of such conversion, and compared it with other conventional AD risk factors, such as the Apolipoprotein E4 (APOE-ε4) genotype, and AD-associated single nucleotide polymorphisms (SNPs).

3.2.2.3 Results

We used all 33 pathologically verified subjects to train an SVM classifier, and evaluated the performance using a leave-one-out cross-validation. We achieved 88% accuracy in diagnostic discrimination (sensitivity=88.8%, specificity=86.6%). The classifier was then used on ADNI subjects to predict pathological diagnoses based on anatomical patterns of atrophy. All ADNI subjects had been clinically diagnosed. By adding predicted pathological diagnoses, all subjects received a binary label that referred to clinical diagnosis and SVM prediction, e.g. "clinically healthy/predicted AD" or HC_AD. ADNI subjects fell into four groups: 275 clinical healthy controls with normal anatomical patterns: HC_HC, 192 clinical AD with an AD atrophy pattern: AD_AD, 91 clinically diagnosed AD subjects were classified as HC by the SVM, and 83 clinically diagnosed HC subjects had an AD-specific atrophy pattern.

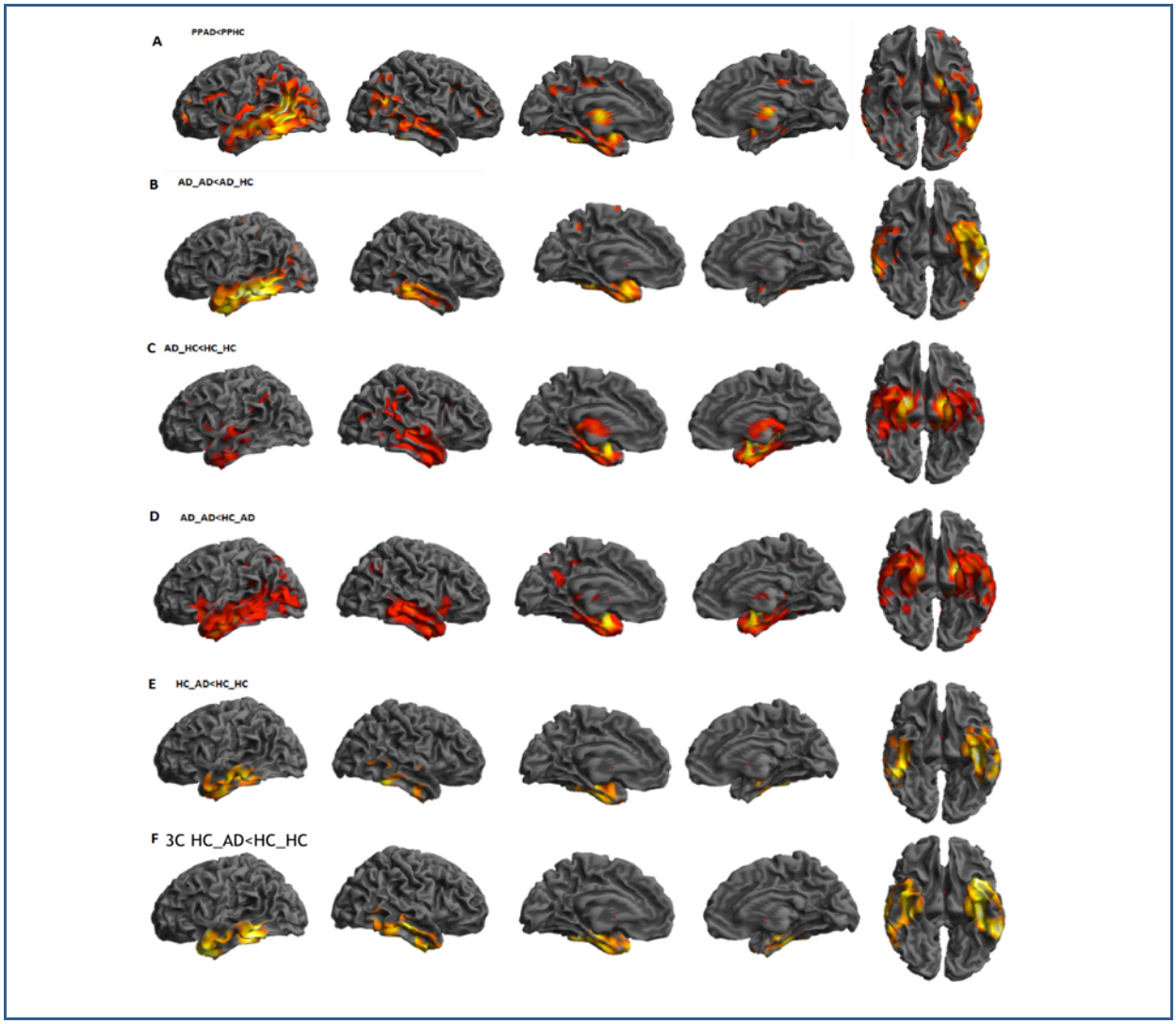

Figure 4: Results of the study

A shows a comparison of pathologically validated AD and HC revealed atrophy. B—E show group comparisons among ADNI subgroups. F compares the atrophy pattern of cognitively normal participants classified as AD (HC_AD) in the 3C study, to those classified as HC_HC.

3.2.2.4 Survival analysis

Clinical HC participants were followed for a median of 35.8 months. We examined whether people diagnosed as healthy (based on clinical scores) at their first visit developed memory impairment during follow-up, as a function of SVM predicted outcome (HC_AD vs. HC_HC). Five years after prediction at the first screening visit, clinical HC classified as AD patients had a survival rate of 66.6% (95% confidence interval [CI], 52% to 83%) in terms of conversion to memory impairment or dementia. On the other hand, HC_HC had an 83% survival rate (95% CI, 76% to 89%). Log-rank testing showed a significant difference in conversion time between the two groups (p= 4.3e-03).

All AD related factors were tested for associations with conversion to memory impairment or dementia using Cox's proportional hazard regression. After adjustment for gender, age, APOE-ε4 genotype and education, we found that, cognitively, HC subjects with atrophy had a 2.5-fold higher risk of developing memory impairment than those without.

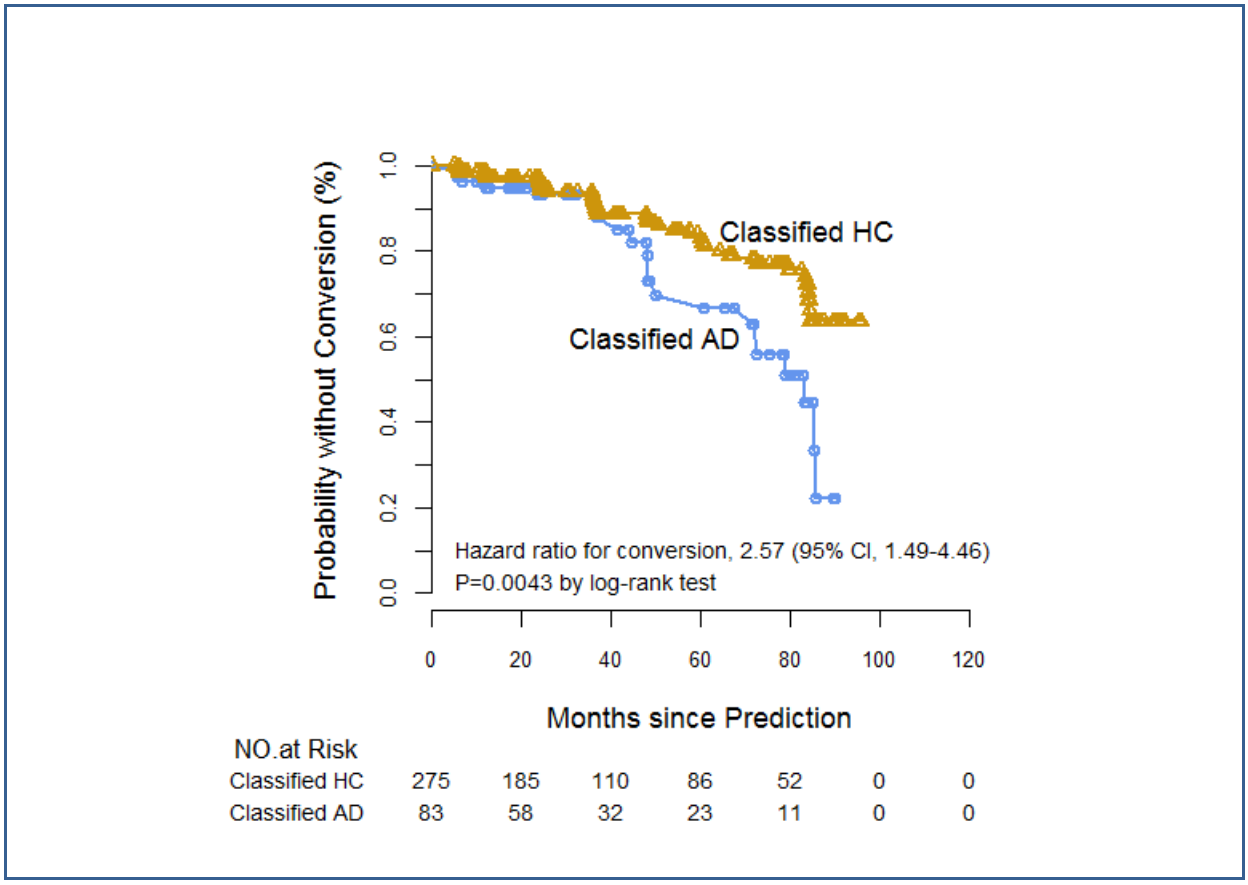

Figure 5: Survival analyses

Our methods identified that cognitively healthy individuals with atrophy have a 2.5 times higher risk of developing memory impairment than those without atrophy.

Conclusion: Individuals with mismatched labels showed intermediate characteristics in both anatomy patterns and memory performance. This characterises different mechanisms related to AD.

3.2.3 Informatics-based Model: Deep Learning for Automated Feature Extractions

3.2.3.1 Objectives

The increasing calculation power of computers has led to a rising interest in complex machine learning methods. In particular, the investigation of artificial neural networks with many hidden layers continuously results in promising new applications. These include image and face recognition, speech recognition and signal processing. Very recently, these deep learning networks have also been used in the classification of AD patients versus healthy control subjects, resulting in accuracies of up to 95%.[5]

3.2.3.2 Methods

We used MRI scans from the publicly available ADNI database. T1-weighted scans were used from 359 HCs and 284 mild AD patients. The groups were matched for age and gender. We tested two methods of feature extraction for training deep learning networks, both of which ensured the anonymity of the individual subjects, and resulted in manageable input vectors. The first of these was the classical region of interest (ROI) approach, using volumes of grey matter regions as input values. The second feature set consisted of a random set of two-dimensional sub-images extracted from each of the MRI scans. The pixel values of these patches were fed directly into the neural network for training and classification. The main difference between our study and previous attempts

to use small sub regions for AD diagnosis is that, in our study, no preliminary group comparison was performed to select affected brain regions. On the contrary, proper analysis of the classification results, based on randomly located training patches, can be used to locate affected grey matter regions in individual AD patients.

One of the most widely used deep neural networks for classification is the stacked auto-encoder. An auto-encoder contains three layers, the first (input) and last (output) layer of which are identical and known. These are connected to the neurons in the middle layer (the hidden layer) by means of weight matrices W_{ji}, where the indices refer to connections from layer i to layer j. There are no connections between neurons in a single layer. For a certain input vector, each neuron in the hidden layer produces an output value defined by the relation $y = f(W_{21}.x + b_2)$. The value y is called the activation of a neuron. The function f can take any form, but because of its saturation properties, and advantageous mathematical properties, mostly sigmoid functions (varying from 0 to 1) or hyperbolic tangents (varying from -1 to 1) are used. The value b_2 is a bias linked to each neuron in a hidden layer. Its value has to be optimised, together with the weights in W_{ij}. In a second step, the output vector y is used as input to reproduce the original input layer: $\hat{x} = f(W_{32}.y + b_3)$. The aim of an auto-encoder is to minimise the difference between $\hat{x}$ and x, by finding an optimal value for W_{ji} and b_j. This way, an initial input vector can be encoded and decoded using the optimal weight matrices and bias vectors. If the number of hidden neurons is lower than the number of elements in the input vector, an auto-encoder can be used as a data compression mechanism. For this type of training, only input values are required; it is therefore called unsupervised learning.

Several of the hidden layers can be stacked in order to capture more complex properties of the input, and these deep structures have been used successfully in many classification tasks. The top layer represents the classification label of the input vector, and once again the weights in the network have to be optimised to obtain good agreement with the given labels. Since output values are now compared to known labels, this technique is referred to as supervised learning.

Due to the very high amount of parameters to fit, training the entire neural network based on an input vector and a classification label is very slow, and often results in low quality local optima. However, it has been shown empirically that unsupervised pre-training of each individual layer, followed by a supervised optimisation of the entire network, yields good results.

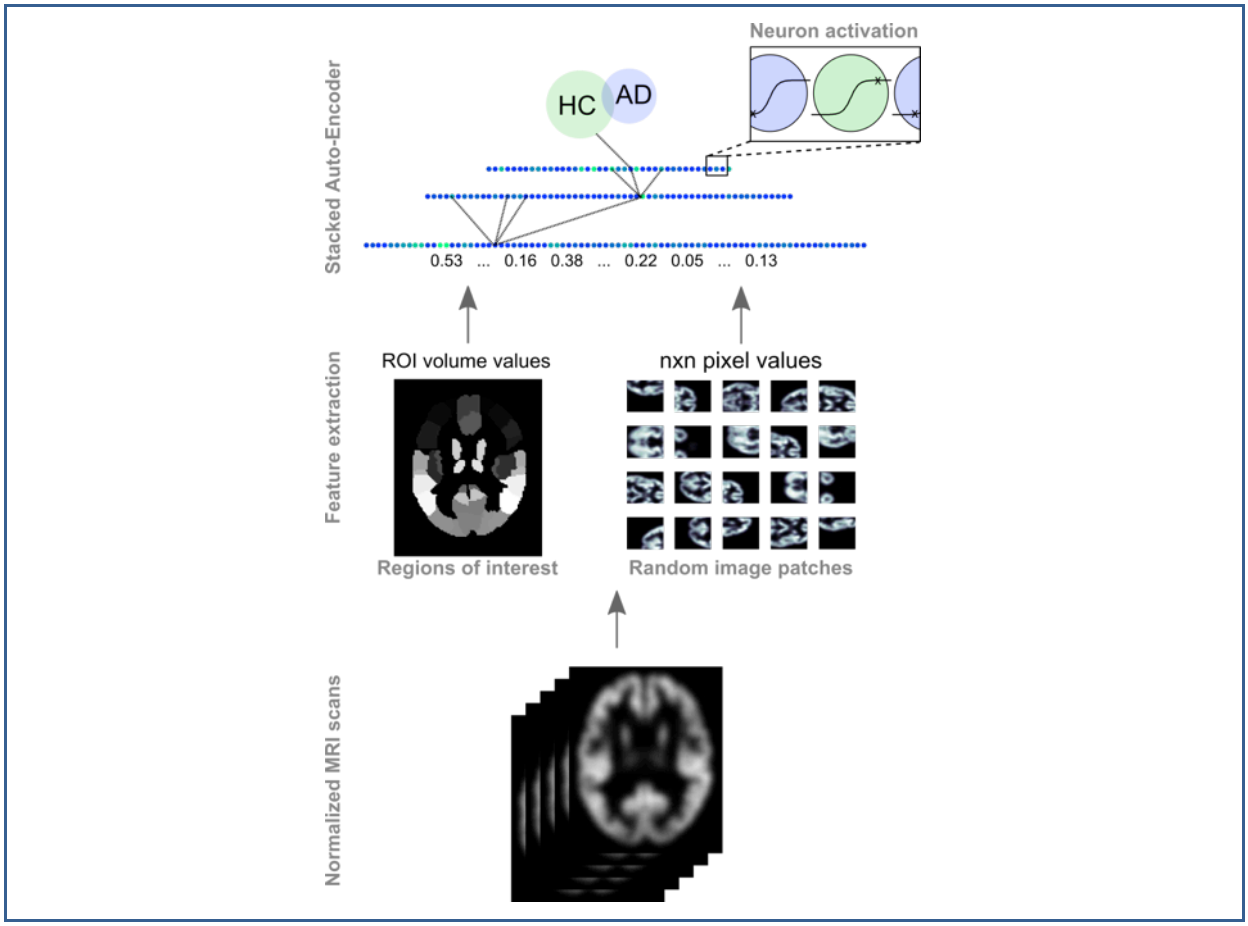

Figure 6: Feature extraction for a stacked auto-encoder containing two hidden layers

3.2.3.3 Results

Using volumes of grey matter ROI defined by a brain atlas, HC subjects were correctly classified in 75% of cases, and only 7% were misclassified. The performance accuracy is lower for AD classification (57% correct vs. 19% misclassified), but the misclassified subjects actually show structural properties of the HC subjects (almost no atrophy in temporal and hippocampal areas). Additionally, the MMSE scores of misclassified AD subjects were significantly higher than those of the true AD subjects. This could mean that we have identified a subgroup that needs a different label referring to a mild or early stage of the disease.

The alternative approach for feature extraction, using random two-dimensional patches of normalised grey matter scans, resulted in distinctive results for HC and AD subjects. The most striking of these was the fact that patches from temporal and hippocampal regions performed much better for AD classification, whereas we noticed no significant variability in classification accuracy of individual patches for HC patients. It is, however, not straightforward to put a single label on a subject, due to the classification distribution of the patches. A possible approach might be to define a threshold of the number of correctly classified patches, above which the subject is supposed to be healthy. Based on our results, a possible threshold value might be 70%, leading to an accuracy of 84% for HC subjects, and 93% for AD subjects.

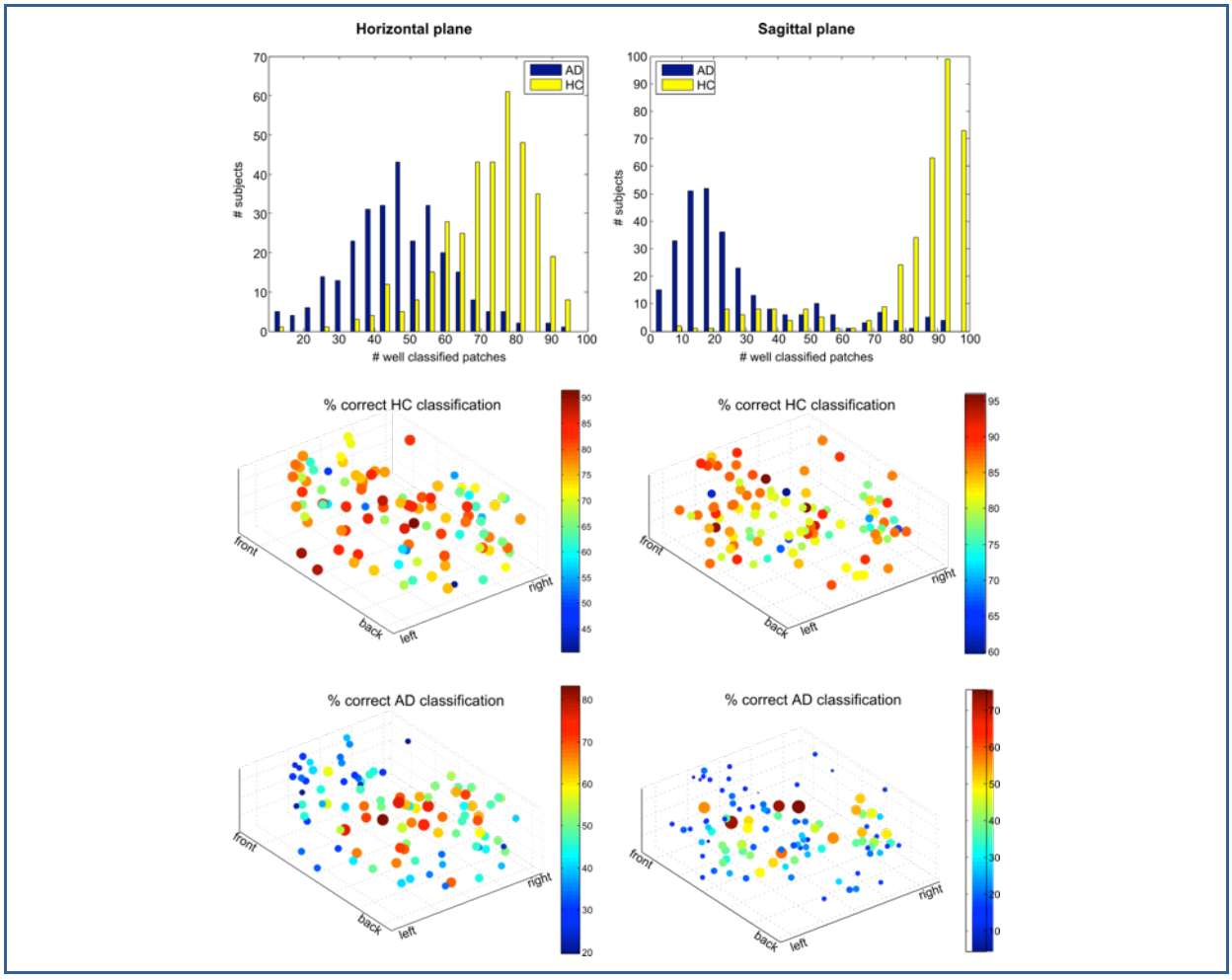

Figure 7: Results for classification based on random patches used as input features

Histograms of well classified patches (top) per group. Location of patches showing the percentage of correct classifications for HC (middle) and AD (bottom). The size and colour of the dots refer to the percentage of correct classifications.

3.2.3.4 Conclusion

The results show that low-level features extracted from the scans without any preliminary knowledge, combined with a fairly easy deep learning scheme, has promising classification potential. Apart from being able to classify single subjects (which is not possible using statistical methods such as VBM), additional conclusions such as the definition of subgroups or finding brain regions affected by a disease may be possible.

3.2.4 Informatics-based Model: Rasch model and factor analysis for learning disease severity

3.2.4.1 Objectives

Clinical diagnostic criteria for AD lack the power to reveal pathological changes in the brain, especially neurofibrillary tangles (NFT) deposition during Braak stages. The atrophy patterns detected by MR-scans correlate with NFT Braak stages. In this project, we propose to extract an index or a latent variable from the neuroimaging data to quantify the disease severity for each subject and regional vulnerability by applying factor analysis. Since the atrophy pattern correlates with the loss of neurons, this severity has a biological meaning and is independent of symptoms. We aim to test if the estimated severity significantly associates with clinical diagnosis and identify the regions highly weighted in calculating the severity.

3.2.4.2 Methods

We collected ADNI T1-weighted MRI, including 343 cognitively normals, 731 mild cognitive impairments (MCIs) and 270 ADs, preprocessed with SPM12. The 114 grey matter regional volumes were extracted based on neuromorphometrics atlas then scaled by total grey matter volume and adjusted by age. We applied a factor analysis algorithm to estimate the disease severity using. In the post-hoc analysis, we compared the probability density distributions of the disease severity of the three clinical groups pair-wise by using a two sample T-test. To identify regional vulnerability, we applied linear regression.

3.2.4.3 Results

The three clinical groups have significantly different estimated severity (Fig. 8, below). Regions affected at early stages are highly weighted (Fig. 9).

3.2.4.4 Conclusion

The clinically diagnosed groups have significantly different distributions of severity but with a big overlap between them. This is expected due to the blurry clinical diagnostic criteria.

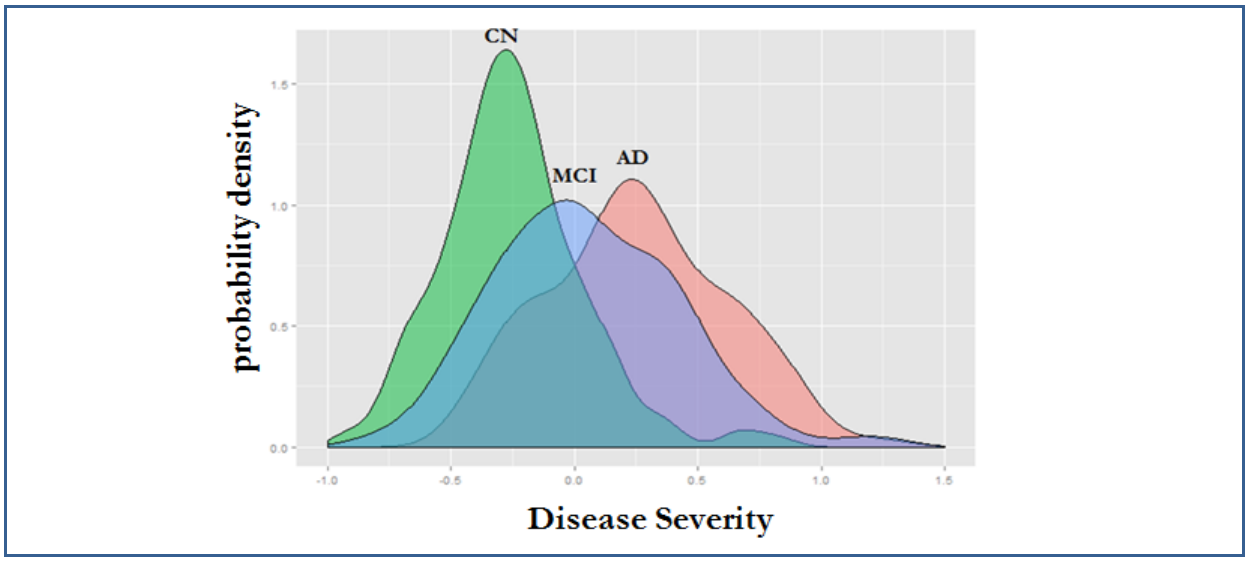

Figure 8: The probability density distribution of the three clinical groups.

In the post-hoc analysis, we applied a two sample T-test to compare each pair of groups: cognitive normals (CN) vs mild cognitive impairments (MCI), CN vs Alzheimer's disease (AD) and MCI vs AD. Three pairs of groups showed significantly different distributions with p-value < 0.001.

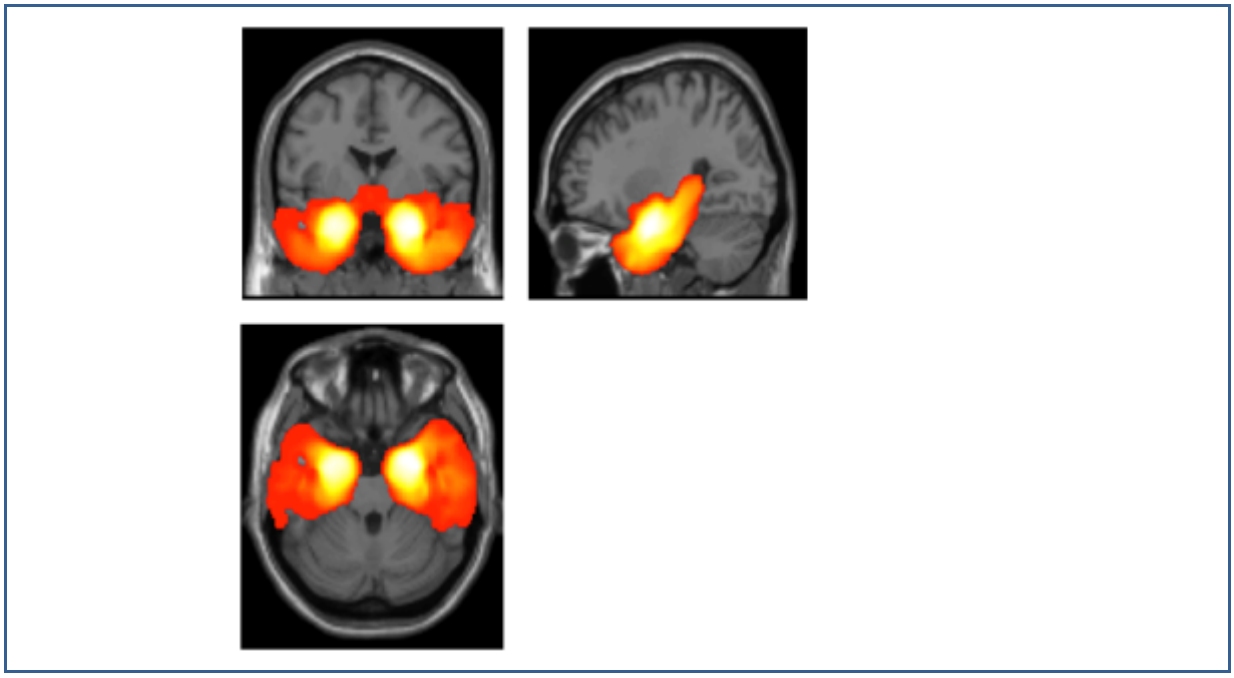

Figure 9: The regions highly weighted in estimating disease severity.

In the post-hoc analysis, we applied linear regression to identify regional vulnerability. The top weighted regions are right entorhinal area, right parahippocampal gyrus, left entrohinal area, left parahippocampal gyrus.

3.2.5 Informatics-based Model: Bi-clustering applied to gene expression and brain volumetric data

3.2.5.1 Objectives

The Ping-Pong Algorithm (PPA) is a bi-clustering method used to compare two datasets with one common dimension.[6] In our application, these datasets are the gene expression in blood, and brain volumetric data from the Alzheimer's Disease Neuroimaging Initiative (ADNI), where the common dimension is the subjects. It returns a set of co-modules, where each co-module consists of a set of genes, subjects, and brain regions, such that the subjects exhibit similar (or opposite) profiles of expression for the corresponding genes and profiles of volume for the corresponding brain regions. The aim of this study was to identify potential biomarkers in blood and gain insight into the aetiology and progression of AD.

3.2.5.2 Methods

The algorithm uses a random weighted set of genes as a starting point (called a seed). It then selects subjects in which these genes deviate from the mean across subjects. Using these subjects, it then selects brain regions whose volumes deviate from the mean in these subjects. Using these regions, it selects a second vector of subjects for which the volumes of these regions deviate from the mean across subjects. Finally, it selects a new set of genes whose expression in these subjects deviates from the norm. This process is repeated until convergence is reached (i.e. the gene, subject, and region sets do not change from one iteration to the next). A more precise mathematical definition is given in Box 1.

Box 1 The Ping-Pong Algorithm (PPA)

For a given threshold combination (t_C: condition threshold, t_G: gene threshold, t_D: drug threshold) the Ping-pong algorithm (PPA) is summarized in the following pseudocode:

- $n = 0; \boldsymbol{g}^{(0)} = \mathbf{random}(N_G) \in [0,1]^{N_G}$(initial random *seed*)
- **while** $(|\hat{\boldsymbol{g}}^{(n)} - \hat{\boldsymbol{g}}^{(n-1)}| + |\hat{\boldsymbol{d}}^{(n)} - \hat{\boldsymbol{d}}^{(n-1)}| + |\hat{\boldsymbol{c}}^{(n)} - \hat{\boldsymbol{c}}^{(n-1)}| + |\hat{\boldsymbol{c}}^{(n)}|\tilde{\hat{\boldsymbol{c}}}^{(n)}| > \varepsilon)$
  1. $\boldsymbol{c} = \boldsymbol{E}_G^T \cdot \hat{\boldsymbol{g}}^{(n)}; c_j^{(n+1)} = \begin{cases} c_j : \text{if } |c_j - \mu(\boldsymbol{c})| > t_c \sigma(\boldsymbol{c}) \\ 0 : \text{otherwise} \end{cases} (j = 1,\ldots,N_C)$
  2. $\boldsymbol{d} = \boldsymbol{R}_c \cdot \hat{\boldsymbol{c}}^{(n)}; d_k^{(n+1)} = \begin{cases} d_k : \text{if } |d_k - \mu(\boldsymbol{d})| > t_D \sigma(\boldsymbol{d}) \\ 0 : \text{otherwise} \end{cases} (k = 1,\ldots,N_D)$
  3. $\tilde{\boldsymbol{c}} = \boldsymbol{R}_D^T \cdot \hat{\boldsymbol{d}}^{(n)}; \tilde{c}_l^{(n+1)} = \begin{cases} \tilde{c}_l : \text{if } |\tilde{c}_l - \mu(\tilde{\boldsymbol{c}})| > \tilde{t}_c \sigma(\tilde{\boldsymbol{c}}) \\ 0 : \text{otherwise} \end{cases} (l = 1,\ldots,N_C)$
  4. $\mathbf{g} = \mathbf{E}_C \cdot \tilde{\hat{\mathbf{c}}}^{(n)}; g_m^{(n+1)} = \begin{cases} g_m : \text{if } |g_m - \mu(\boldsymbol{g})| > t_G \sigma(\boldsymbol{g}) \\ 0 : \text{otherwise} \end{cases} (m = 1,\ldots,N_G)$
  5. $n = n+1$
- $\boldsymbol{g}^* = \mathbf{g}^{(n)}; \hat{\boldsymbol{c}}^* = \hat{\boldsymbol{c}}^{(n)}; \boldsymbol{d}^* = \hat{\boldsymbol{d}}^{(n)}$

Starting from a random seed, the algorithm identifies weighted lists of genes, conditions and drugs as fixed points of iterating the four linear mappings followed by thresholding (which correspond to the four numbered arrows in **Fig. 1c**). The weights (or scores) are the components x_i of the gene, condition and drug vector, $\boldsymbol{x} \in \{\boldsymbol{g}, \boldsymbol{c}, \boldsymbol{d}\}$, of length N_G, N_C, N_D, respectively. $|\boldsymbol{x}|$, $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ denote their norm, mean value and the standard deviation. $\hat{\boldsymbol{x}} = \boldsymbol{x}/|\boldsymbol{x}|$ is a normalized vector. $\boldsymbol{E}_G$ and $\boldsymbol{E}_C$ refer to the expression matrix normalized across genes and conditions, respectively. Similarly, $\boldsymbol{R}_D$ and $\boldsymbol{R}_C$ denote the response matrix normalized across drugs and conditions, respectively. (See **Supplementary Notes** for more details.)

3.2.5.3 Results

Although ADNI subjects have (at most) a single gene expression analysis, most have multiple MRI scans. This allowed us to estimate the rate of change in brain volume (rather than the volume itself) and compare this to gene expression in blood. We selected subjects with at least two MRI scans and gene expression data between the first and last scans. The atrophy in each brain region was regressed to an exponential function. The slope of this function was used as an estimate for the rate of atrophy at the time when gene expression was taken. This rate of atrophy and the gene expression were then used as input for the PPA.

The PPA does not use any information pertaining to the cognitive status of subjects, and is therefore liable to pick up co-modules relating to any number of covariates, such as age and gender. In order to select co-modules relevant to Alzheimer's disease, we performed Cox's proportional hazards modelling (with age and gender as covariates) and selected co-

modules which were significantly associated with the diagnosis. P-values were corrected using false discovery rate (FDR) and only co-modules with an FDR below 0.1 were selected. These co-modules were then validated using ADNI subjects for which we have gene expression data but were not used in the first step (due to insufficient MRI data). Several co-modules were validated this way and their analysis is underway.

3.2.5.4 Conclusion

The analysis of these co-modules is mainly gene-centric. Bgee's TopAnat showed that the co-module genes were enriched for expression in various regions of the brain. GeneMANIA allowed the construction of a network whose central genes have known roles in brain functions or have previously been implicated in various neurological pathologies, such as Alzheimer's disease, Parkinson's disease, or autism. These results further confirm the relevance of the co-modules found by PPA, however more analyses are required to elucidate the role of these genes in Alzheimer's disease.

3.2.6 Informatics-based Model: Bayesian Causal Model

3.2.6.1 Objectives

We aimed at designing, deriving equations and implementing the causal Bayesian model similar to the General Linear Model (GLM) for distributed Data. GLM is one of the most used models to estimated dependences between clinical, neuropsychological and neuroimaging variables. In our case the data are distributed in different hospitals and it is not possible to move them to a unique federation node where the GLM could be computed in a classical way. Therefore special equations should be developed that allow us to have reliable GLM estimations under this condition. The Bayesian Formalism provides us the necessary armamentarium to deal with it and offers general sophisticated ways to extend to other models and manage high dimensional and Multimodal Data.

3.2.6.2 Methods

In statistics, the GLM allows us to study the relationship between a dependent variable **y** and one or more explanatory variables (or independent variables) enclosed and denoted by **A**.

The general problem is to solve the standard linear equation posed as following:

$$\mathbf{y} = \mathbf{A}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1}$$

y: vector of dependent variable, $K \times 1$

A: Design matrix, matrix of regressors, $K \times N$

$\boldsymbol{\beta}$: Vector of regression coefficients, $N \times 1$

We have the special situation in the HBP Neuroinformatics Platform where data **y** and **A** are naturally split and distributed in different 'M' Hospitals. Additionally we cannot fetch the data to a common federation site to try to solve equation (1). This is not possible due to two main reasons: 1) the data volume is big enough that moving them to one common remote site is time consuming implying high data traffic; and 2) it is mandatory to preserve clinical data confidentiality, which is not fully guaranteed if exchanging data between hospitals and federation nodes is performed. Thus we have to find a method to estimate $\boldsymbol{\beta}$ combining aggregates coming from distributed hospital data and meet the main two points above.

To determine $\boldsymbol{\beta}$ under these conditions we make use of the Bayesian formalism. Through this approach, it is naturally possible to handle the issue of having distributed data over different databases located in different hospitals. We will take advantage of the known 'Divide and Recombine' Bayesian paradigm.[7] Here, there are two main cases: 1) *Parallel*

paradigm: when the data from different hospitals arrive in parallel and an estimate of $\boldsymbol{\beta}$ is obtained with all hospital aggregates at the same time; and 2) *Streaming Paradigm*: the estimation of $\boldsymbol{\beta}$ is updated every moment a new aggregate arrives from a specific Hospital until 'M' Hospitals are covered.

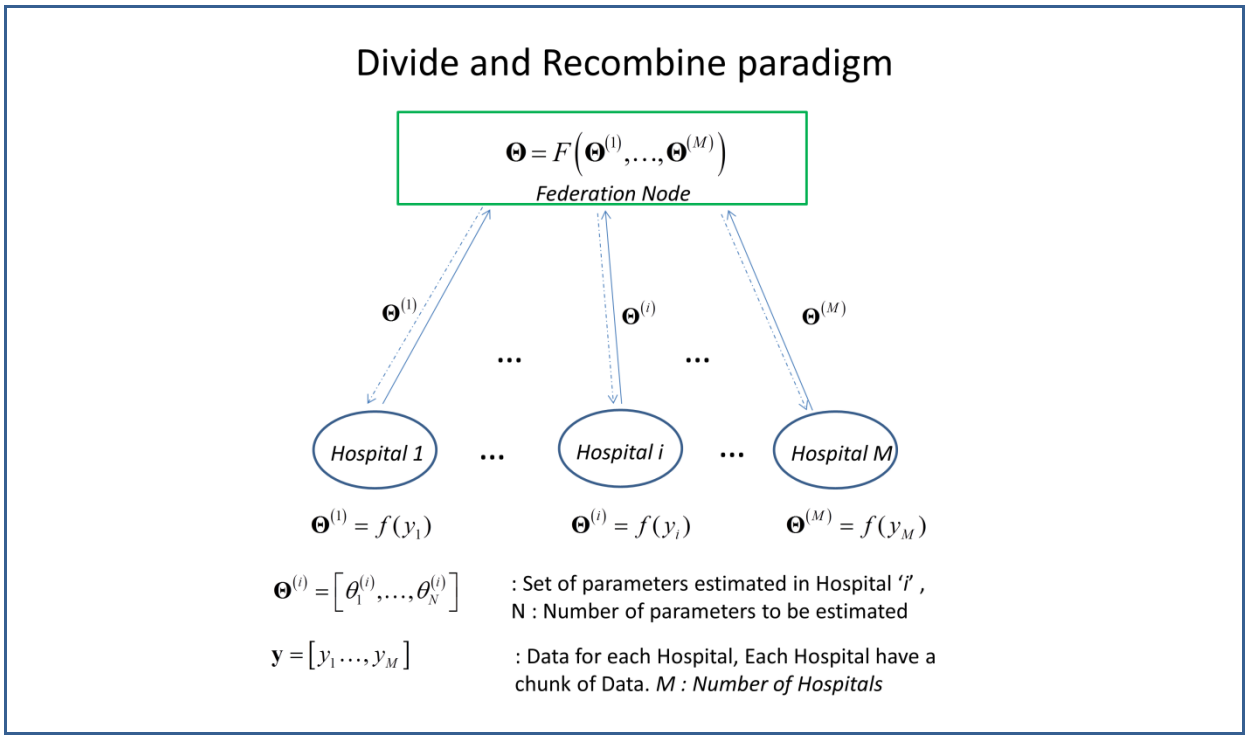

Figure 10: General 'Divide and Recombine' parallel paradigm under HBP context.

Divide and Recombine Bayesian paradigm**:** We suggest that $\mathbf{y} = \left[\mathbf{y}_1,\ldots,\mathbf{y}_M\right]$ and $\mathbf{A} = \left[\mathbf{A}_1,\ldots,\mathbf{A}_M\right]$, vectors $\mathbf{y}_i$ are $K_i \times 1$, $(i = 1,\ldots,M)$ that correspond to a design matrix $\mathbf{A}_i$ with dimensions $K_i \times N$, where $K = \sum_{i=1}^{M} K_i$.

Using the Bayesian rule we calculate that the posterior probability of $\boldsymbol{\beta}$ can be expressed as:

$$p\left(\boldsymbol{\beta}|\mathbf{y}\right) \propto p\left(\mathbf{y}|\boldsymbol{\beta}\right) \cdot p\left(\boldsymbol{\beta}\right) \tag{2}$$

where $p\left(\mathbf{y}|\boldsymbol{\beta}\right)$ is the likelihood and $p\left(\boldsymbol{\beta}\right)$ the prior probability for regression coefficients.

Assuming that chunks of data are independent, we calculate that the likelihood can be expressed as:

$$p\left(\mathbf{y}|\boldsymbol{\beta}\right) = p\left(\mathbf{y}_1|\boldsymbol{\beta}\right) \cdot p\left(\mathbf{y}_2|\boldsymbol{\beta}\right) \cdots p\left(\mathbf{y}_M|\boldsymbol{\beta}\right) \tag{3}$$

where,

$$p\left(\mathbf{y}_i|\boldsymbol{\beta}\right) = N\left(\mathbf{A}_i\boldsymbol{\beta}, \frac{1}{\sigma_i}\boldsymbol{\Theta}\right) = \frac{\sigma_i^{N/2}}{(2\pi)^{N/2}\left|\boldsymbol{\Theta}\right|^{1/2}} \exp\left(-\frac{\sigma_i}{2}\left(\mathbf{y}_i - \mathbf{A}_i\boldsymbol{\beta}\right)^T \boldsymbol{\Theta}^{-1}\left(\mathbf{y}_i - \mathbf{A}_i\boldsymbol{\beta}\right)\right) \tag{4}$$

Substituting this in equation (2), we obtain:

$$p\left(\boldsymbol{\beta}|\mathbf{y}\right) \propto p\left(\mathbf{y}_1|\boldsymbol{\beta}\right) \cdot p\left(\mathbf{y}_2|\boldsymbol{\beta}\right) \cdots p\left(\mathbf{y}_M|\boldsymbol{\beta}\right) \cdot p\left(\boldsymbol{\beta}\right) \tag{5}$$

This can be rewritten as:

$$p(\boldsymbol{\beta}|\mathbf{y}) \propto \frac{\overbrace{p(\mathbf{y}_1|\boldsymbol{\beta})\cdot p(\boldsymbol{\beta})}^{p(\boldsymbol{\beta}|\mathbf{y}_1)}\overbrace{p(\mathbf{y}_2|\boldsymbol{\beta})\cdot p(\boldsymbol{\beta})}^{p(\boldsymbol{\beta}|\mathbf{y}_2)}\cdots\overbrace{p(\mathbf{y}_M|\boldsymbol{\beta})\cdot p(\boldsymbol{\beta})}^{p(\boldsymbol{\beta}|\mathbf{y}_M)}}{\left(p(\boldsymbol{\beta})\right)^{M-1}} \tag{6}$$

Then the posterior probability of the regression coefficients will be expressed as a combination of the posterior probabilities using different chunks of data.

If the posterior probabilities of each chunk '*i*' of data is normally distributed, as follows, we obtain that:

$$p(\boldsymbol{\beta}|\mathbf{y}_i) = N(\hat{\boldsymbol{\beta}}_i, \boldsymbol{\Psi}_i) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Psi}_i|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_i)^T\boldsymbol{\Psi}_i^{-1}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_i)\right) \tag{7}$$

Assuming that the prior probability $p(\boldsymbol{\beta})$ is defined as:

$$p(\boldsymbol{\beta}) = N\left(\hat{\boldsymbol{\beta}}_0, \frac{1}{\lambda}\boldsymbol{\Psi}_0\right) = \frac{\lambda^{N/2}}{(2\pi)^{N/2}|\boldsymbol{\Psi}_0|^{1/2}}\exp\left(-\frac{\lambda}{2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_0)^T\boldsymbol{\Psi}_0^{-1}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_0)\right) \tag{8}$$

We obtain that the posterior will be expressed as:

$$p(\boldsymbol{\beta}|\mathbf{y}) = N(\hat{\boldsymbol{\beta}}_f, \boldsymbol{\Psi}_f) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Psi}_f|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_f)^T\boldsymbol{\Psi}_f^{-1}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}}_f)\right) \tag{9}$$

where:

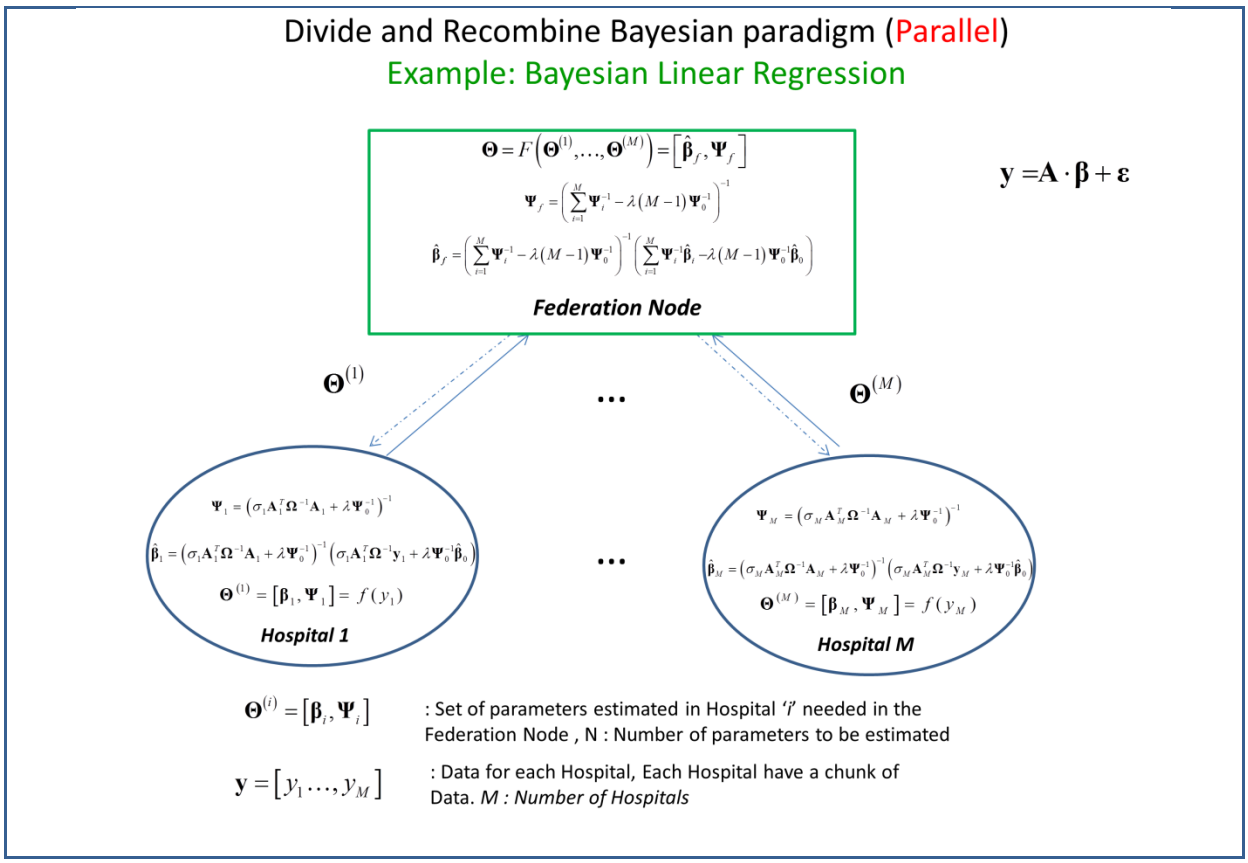

Figure 11: Divide and Recombine parallel paradigm for general linear model under HBP context.

$$\boldsymbol{\Psi}_f = \left(\sum_{i=1}^{M} \boldsymbol{\Psi}_i^{-1} - \lambda (M-1) \boldsymbol{\Psi}_0^{-1} \right)^{-1} \tag{10}$$

and

$$\hat{\boldsymbol{\beta}}_f = \left(\sum_{i=1}^{M} \boldsymbol{\Psi}_i^{-1} - \lambda (M-1) \boldsymbol{\Psi}_0^{-1} \right)^{-1} \left(\sum_{i=1}^{M} \boldsymbol{\Psi}_i^{-1} \hat{\boldsymbol{\beta}}_i - \lambda (M-1) \boldsymbol{\Psi}_0^{-1} \hat{\boldsymbol{\beta}}_0 \right) \tag{11}$$

The means $\hat{\boldsymbol{\beta}}_i$ and covariance matrices $\boldsymbol{\Psi}_i$ estimated for the '*i*' chunk of data are expressed as:

$$\hat{\boldsymbol{\beta}}_i = \left(\sigma_i \mathbf{A}_i^T \boldsymbol{\Theta}^{-1} \mathbf{A}_i + \lambda \boldsymbol{\Psi}_0^{-1} \right)^{-1} \left(\sigma_i \mathbf{A}_i^T \boldsymbol{\Theta}^{-1} \mathbf{y}_i + \lambda \boldsymbol{\Psi}_0^{-1} \hat{\boldsymbol{\beta}}_0 \right) \tag{12}$$

$$\boldsymbol{\Psi}_i = \left(\sigma_i \mathbf{A}_i^T \boldsymbol{\Theta}^{-1} \mathbf{A}_i + \lambda \boldsymbol{\Psi}_0^{-1} \right)^{-1} \tag{13}$$

For the case where data at each chunk $\mathbf{y}_i$ is considered independent and is standardised, we calculate that: $\boldsymbol{\Theta} = \mathbf{I}$, $\mathbf{I}$ is the identity matrix, and $\sigma_i = 1$. Also we assume that $\hat{\boldsymbol{\beta}}_0 = 0$. Based on these plausible assumptions (common in practice) we have that equations (12) and (13) become:

$$\hat{\boldsymbol{\beta}}_i = \left(\mathbf{A}_i^T \mathbf{A}_i + \lambda \boldsymbol{\Psi}_0^{-1} \right)^{-1} \mathbf{A}_i^T \mathbf{y}_i \tag{14}$$

$$\boldsymbol{\Psi}_i = \left(\mathbf{A}_i^T \mathbf{A}_i + \lambda \boldsymbol{\Psi}_0^{-1} \right)^{-1} \tag{15}$$

Doing the specific maths, the estimator of $\boldsymbol{\beta}$ at the Federation node is expressed as:

$$\boldsymbol{\Psi}_f = \left(\sum_{i=1}^{M} \boldsymbol{\Psi}_i^{-1} - (M-1) \boldsymbol{\Psi}_0^{-1} \right)^{-1} \tag{16}$$

$$\hat{\boldsymbol{\beta}}_f = \left(\sum_{i=1}^{M} \boldsymbol{\Psi}_i^{-1} - (M-1) \boldsymbol{\Psi}_0^{-1} \right)^{-1} \left(\sum_{i=1}^{M} \boldsymbol{\Psi}_i^{-1} \hat{\boldsymbol{\beta}}_i - (M-1) \boldsymbol{\Psi}_0^{-1} \hat{\boldsymbol{\beta}}_0 \right) \tag{17}$$

Where the local estimations (at hospital level) are estimated as:

$$\hat{\boldsymbol{\beta}}_i = \left(\mathbf{A}_i^T \mathbf{A}_i + \lambda \boldsymbol{\Psi}_0^{-1} \right)^{-1} \mathbf{A}_i^T \mathbf{y}_i \tag{18}$$

$$\boldsymbol{\Psi}_i = \left(\mathbf{A}_i^T \mathbf{A}_i + \lambda \boldsymbol{\Psi}_0^{-1} \right)^{-1} \tag{19}$$

$$\boldsymbol{\Psi}_f^{-1(1)} = \boldsymbol{\Psi}_1^{-1}$$

$$\boldsymbol{\Psi}_f^{-1(2)} = \overbrace{\boldsymbol{\Psi}_1^{-1}}^{\boldsymbol{\Psi}_f^{-1(1)}} + \boldsymbol{\Psi}_2^{-1} - \lambda\boldsymbol{\Psi}_0 = \boldsymbol{\Psi}_f^{-1(1)} + \boldsymbol{\Psi}_2^{-1} - \lambda\boldsymbol{\Psi}_0$$

$$\boldsymbol{\Psi}_f^{-1(3)} = \overbrace{\boldsymbol{\Psi}_1^{-1} + \boldsymbol{\Psi}_2^{-1} - \lambda\boldsymbol{\Psi}_0}^{\boldsymbol{\Psi}_f^{-1(2)}} + \boldsymbol{\Psi}_3^{-1} - \lambda\boldsymbol{\Psi}_0 = \boldsymbol{\Psi}_f^{-1(2)} + \boldsymbol{\Psi}_3^{-1} - \lambda\boldsymbol{\Psi}_0 \qquad (20)$$

$$\vdots$$

$$\boldsymbol{\Psi}_f^{-1(M)} = \overbrace{\boldsymbol{\Psi}_1^{-1} + \boldsymbol{\Psi}_2^{-1} + \ldots + \boldsymbol{\Psi}_{M-1}^{-1} - (M-2)\lambda\boldsymbol{\Psi}_0}^{\boldsymbol{\Psi}_f^{-1(M-1)}} + \boldsymbol{\Psi}_M^{-1} - \lambda\boldsymbol{\Psi}_0 = \boldsymbol{\Psi}_f^{-1(M-1)} + \boldsymbol{\Psi}_M^{-1} - \lambda\boldsymbol{\Psi}_0$$

Finally after *K* hospital results arrived, the equations at the federation node will be expressed as:

$$\begin{aligned} \boldsymbol{\Psi}_f^{-1(K)} &= \boldsymbol{\Psi}_f^{-1(K-1)} + \boldsymbol{\Psi}_K^{-1} - \lambda\boldsymbol{\Psi}_0 \\ \hat{\boldsymbol{\beta}}_f^{(K)} &= \boldsymbol{\Psi}_f^{-1(K)}\left(\boldsymbol{\Psi}_f^{-1(K-1)}\hat{\boldsymbol{\beta}}_f^{(K-1)} + \boldsymbol{\Psi}_K^{-1}\hat{\boldsymbol{\beta}}_K - \lambda\boldsymbol{\Psi}_0^{-1}\hat{\boldsymbol{\beta}}_0\right) \end{aligned} \qquad (21)$$

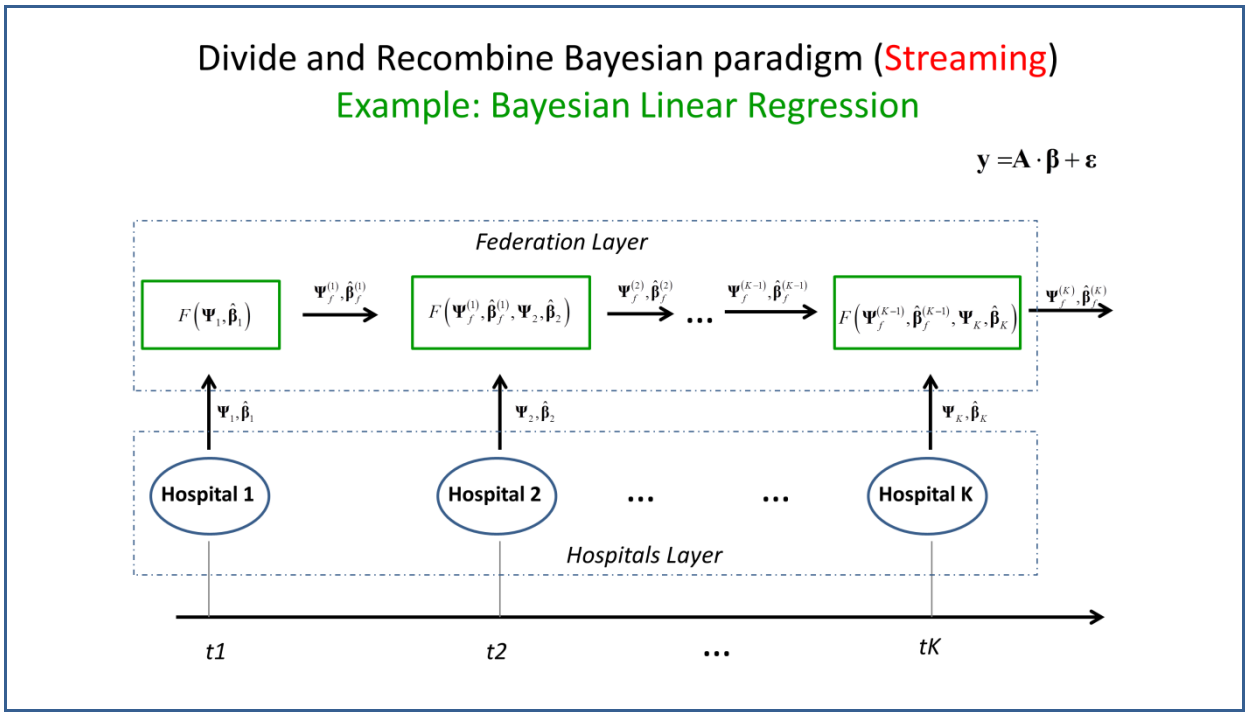

Figure 12: Divide and Recombine streaming paradigm for general linear model under HBP context.

3.2.6.3 Results

The above algorithm was implemented in scientific programming languages R and MATLAB. It was tested in the following example.

3.2.6.4 Example

During the study of AD it is interesting to know how morphometric variables (like grey matter volume, cortical thickness, etc.) covary between anatomical regions. The hippocampus is known as a key region related to AD; therefore, an interesting question is to study how grey matter volume changes in the right hippocampus induce changes in its counterpart the left hippocampus in AD patients. To answer this question, we use a linear regression model (developed above). The dependent variable will be the right

hippocampus volume in AD subjects (y variable), whereas the left hippocampus will be the main explanatory variable (in matrix A). Additionally, we add gender and age as other explanatory variables (in A) in order to minimise their influence in our results, and keep only the correlation between both hippocampi.

3.3 Implementation status of algorithms, classifiers and models with the Medical Informatics Platform (MIP)

This Task relies on data accessible through the Medical Informatics Platform (MIP) WP8.2, including data on the longitudinal study of the large cohort of “control” Alzheimer’s patients.

This Task will identify biological signatures of brain disease and, ultimately, MIP end-users will be able to query the construct. End-users will also be able to compare the derived biological signatures of disease to standard classification (e.g. ICD-10).

The next step is to create comprehensive, simple and causal models of brain diseases that can be run with live data collected by the MIP. The model will be executed locally within each hospital (preserving privacy). This step will allow us to collect additional clinical data for the definition of the subgroups (the causal models are described above).

The implementation within the MIP in the diagram below (data mining algorithms appear in red):

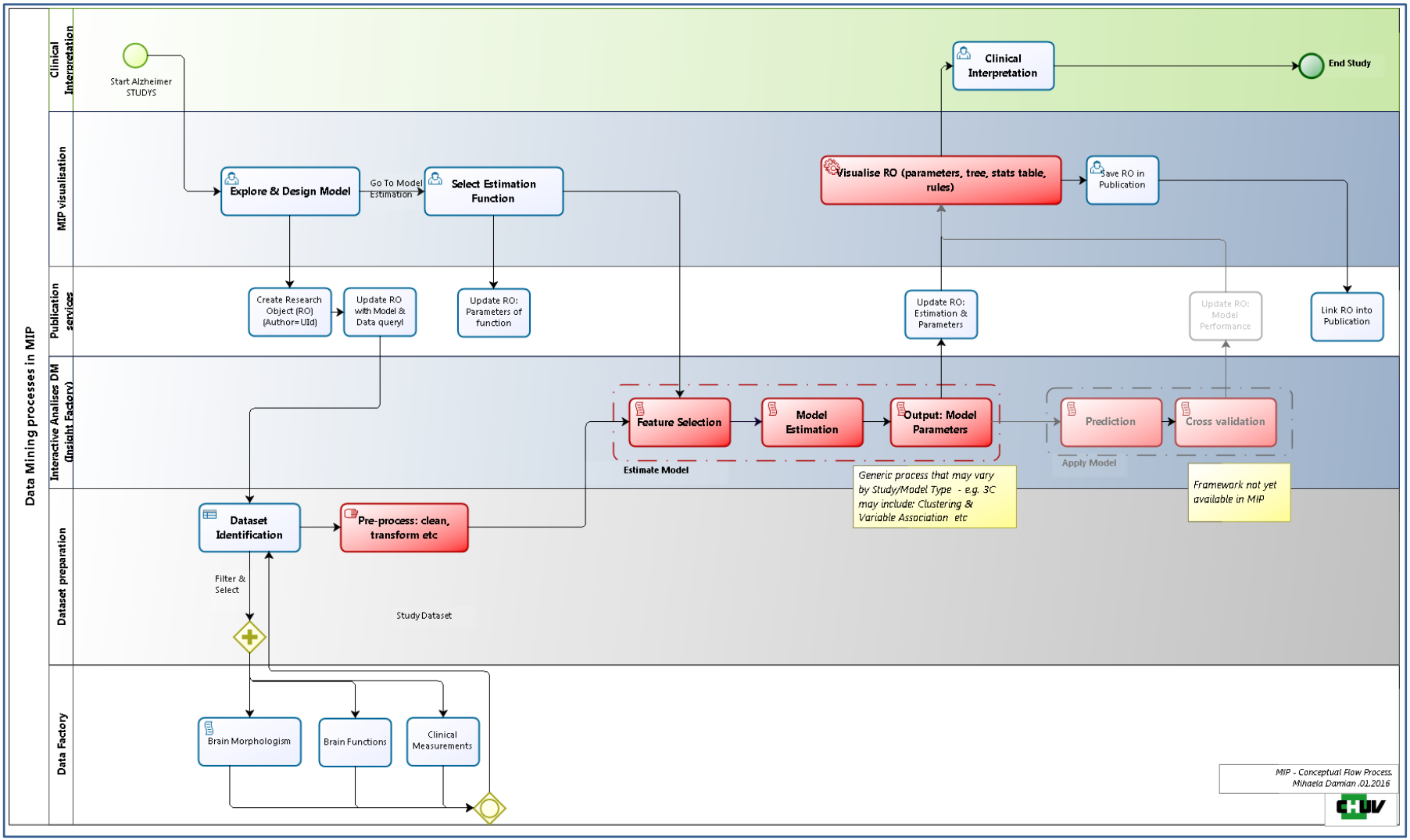

Figure 13: Implementation of algorithms, classifiers and models within the MIP

3.4 Description of major use case(s) and target users

Table 3: Major use cases and target users for the MIP

SP8-UC-006	Biological Signatures of Diseases
	Primary actor: Beth is a general user (GU). She is a **clinician in neurology.** Description: Beth is interested in taking forward personalised diagnostics using biological signatures of the disease. Preconditions: • The biological signatures of diseases produced by the data mining algorithms are available via the MIP Web Portal. • The variables that describe each disease signature cluster have been released and are available via the MIP Web Portal. Success scenario: 1) Beth logs into the Collaboratory. 2) She selects the Biological Signatures of Diseases service and uses the interface to classify her own patient by comparing his/her clinical and biological characteristics with the whole range of provided biological signatures of diseases using an optimal matching algorithm. 3) She does this by selecting variables of interest – e.g. demographic data, blood cholesterol, neuropsychological scores, genetic burden, etc. 4) She enters values for those variables. 5) She retrieves a list of disease signatures ordered according to the best match. The distribution of values of the other unselected variables is also displayed along with their uncertainty – e.g. genotype, clinical scores and cardiovascular risk factors. 6) She also retrieves a 3D brain map with highlighted anatomical regions affected by the particular disease corresponding to the optimally matched disease signature. She can compare the map with the anatomy pattern of her own patients. 7) Depending on how well the disease signature cluster matches her criteria, Beth can add new variables to determine the stability of her classification in relation to the number of criteria or Variables used. 8) She can compare the derived disease signature cluster to conventional clinical classification - e.g. ICD-10, DSM V classification. 9) If needed, she can review her patients (data) to verify the derived disease signature cluster by similarity and by differences with other patients.
SP8-UC-007	Biological Signatures of Diseases
	Primary actor: Nathalie is a GU. She is a **researcher in pharmaceutical R&D.** Preconditions: • The biological signatures of diseases produced by the data mining algorithms are available via the MIP Web Portal. • The Variables that describe each disease signature cluster have been released and are available via the MIP Web Portal. Description: Nathalie is interested in defining inclusion criteria and a set of non-invasive biomarkers for a clinical trial on a new drug for AD. Success scenario: 1) Nathalie logs into the Collaboratory. 2) She selects the Biological Signatures of Diseases service and uses the interface to retrieve the set of features of interest according to the provided disease signatures for dementia of the Alzheimer type. 3) She identifies the features leading to the creation of homogeneously stratified set of rules she would apply to create the cohorts undergoing pharmacological

intervention.
4) She uses the provided predictive tools to infer potential therapeutic targets and positive as well as adverse effects based on multi-scale information – e.g. molecular pathways, proteomics interactions, genetic profiles, etc. up to the system/behavioural level.
5) She is now in a position to specify trials using well-defined homogeneous, and therefore small, cohorts to test the effects of a drug or cocktail of drugs that modulates the targets suggested by the rules that define her disease signature of interest.

3.5 Documentation of models and related tools/applications

Documentation and tutorials about the algorithm are accessible via the MIP Knowledge Base.

Code source are available (https://github.com/LREN-CHUV/functions-repository/blob/master/Guidelines.md).

3.6 Outreach

Abstracts submitted:

- Cui, J., Muller, S., Zufferey, V., Dukart, J., Abdulkadir, A., Klöppel, S., Draganski, B., Zoltán, K., Frackowiak, R., and Kherif, F. *Computation based diagnosis reveals intermediate Alzheimer's disease phenotypes*. Joint Congress of European Neurology, Istanbul, Turkey, 31 May — 3 June 2014. Accepted: oral presentation.
- Cui, J., Muller, S., Zufferey, V., Dukart, J., Abdulkadir, A., Klöppel, S., Draganski, B., Zoltán, K., Frackowiak, R., Kherif, F. *Computation based diagnosis reveals intermediate Alzheimer's disease phenotypes: the follow up study*. Organisation for Human Brain Mapping, Hamburg, Germany, 8-12 June 2014. Accepted: poster presentation.
- Melie-Garcia, L., Ruef, A., Lutti, A., Draganski, B., and Kherif, F. *Anatomical Networks of the Brain Tissue Properties Covariance*. HBM 2016, 22nd Annual Meeting of the Organization for Human Brain Mapping, Geneve, Switzerland, 26-30 June 2016.
- Sanabria-Diaz, G., Kherif, F., Draganski, B., and Melie-Garcia, L. *ApoE-4 modulates the Anatomical Networks in Mild Cognitive Impairment converts to Alzheimer disease*. HBM 2016, 22nd Annual Meeting of the Organization for Human Brain Mapping, Geneve, Switzerland, 26-30 June 2016.
- Sanchez-Catasus, C.A., Sanabria-Diaz, G., Martinez Montes, E., Iturria Medina, Y., Samper Noa, J., Boellaard, R., De Deyn, P., Dierckx, R.A.J.O., and Melie-Garcia, L. *Hypercapnia effects on Networks of Cerebral Blood Flow covariation in Mild Cognitive Impairment*. HBM 2016, 22nd Annual Meeting of the Organization for Human Brain Mapping, Geneve, Switzerland, 26-30 June 2016.
- Melie-Garcia, L., Draganski, B., and Kherif, F. *A Bayesian model for Decoding organizational principles of the brain anatomy using high dimensional and distributed Multimodal Neuroimaging data*. International Conference on Brain Informatics and Health 2015 (BIH2015). London. United Kingdom, 30 August – 2 September 2015. (Poster, Oral presentation).
- Melie-Garcia L. *Bayesian Inference for distributed Big Data Analysis*. Human Brain Project Summit 2015, Madrid, Spain, 27-30 September 2015 (Oral presentation).

Papers (submitted/in preparation):

- Cui, J., Muller, S., Zufferey, V., Dukart, J., Abdulkadir, A., Klöppel, S., Draganski, B., Zoltán, K., Frackowiak, R., and Kherif, F. In preparation: *Computation Based Diagnosis Reveals Intermediate Alzheimer's Disease Phenotypes*.
- Van Damme, B., Cui, J., Draganski, B., and Kherif, F. In preparation: *MRI Feature Selection for Deep Learning Applied to the Classification of Alzheimer's Disease*.

Papers (published):

- Lorio, S., Kherif, F., Ruef, A., Melie-Garcia, L., Frackowiak, R., Ashburner, J., Helms, G., Lutti, A., Draganski, B. *Neurobiological origin of spurious brain morphological changes: A quantitative MRI study*. Human Brain Mapping, 2016. doi: 10.1002/hbm.23137.
- Lorio, S., Fresard, S., Adaszewski, S., Kherif, F., Chowdhury, R., Frackowiak, R.S., Ashburner, J., Helms, G., Weiskopf, N., Lutti, A., and Draganski, B. *New tissue priors for improved automated classification of subcortical brain structures on MRI*. Neuroimage 2016. **130:**157-166.
- Twomey, T., Waters, D., Price, C.J., Kherif, F., Woll, B., MacSweeney, M. *Identification of the regions involved in phonological assembly using a novel paradigm*. Brain and Language, 2015. **150:**45-53.
- Cui, J., Zufferey, V., and Kherif, F. *In-vivo brain neuroimaging provides a gateway for integrating biological and clinical biomarkers of Alzheimer's disease*. Current Opinion in Neurology, 2015. **28:**351-7.
- Maillard, A.M., Ruef, A., Pizzagalli, F., Migliavacca, E., Hippolyte, L., Adaszewski, S., Dukart, J., Ferrari, C., Conus, P., Männik, K., Zazhytska, M., Siffredi, V., Maeder, P., Kutalik, Z., Kherif, F., Hadjikhani, N., Beckmann, J.S., Reymond, A., Draganski, B., and Jacquemont, S.; 16p11.2 European Consortium. *The 16p11.2 locus modulates brain structures common to autism, schizophrenia and obesity*. Molecular Psychiatry 2015. **20:**140-7.
- Lorio, S., Lutti, A., Kherif, F., Ruef, A., Dukart, J., Chowdhury, R., Frackowiak, R.S., Ashburner, J., Helms, G., Weiskopf, N., and Draganski, B. *Disentangling in vivo the effects of iron content and atrophy on the ageing human brain*. Neuroimage 2014. **103:**280-9.
- Draganski, B, Kherif F, Lutti A. Computational anatomy for studying use-dependant brain plasticity. Front Hum Neurosci. 2014 Jun 27;8:380.

4. Future Computing (WP11.3)

4.1 Neuromorphic computing applications: state-of-the art, contributions and value of WP11.3 and future development in the HBP

State-of-the-art

Since the days of it's inception by Carver Mead around the early 1980s neuromorphic computing has seen a development from purely bio-derived analog VLSI circuits to a more general approach that mimics only certain aspects of the brain relevant for a specific computational approach. Important new developments since those early days are the introduction of more digital functionality, motivated mostly by the rapid development of manufacturing technologies in the digital world enabled by Moore's law. The current state-of-the-art in neuromorphic computing reflects this development very clearly. At the time of the end of the HBP ramp-up-phase (RUP) there are 3 large, usable systems available worldwide for experiments and applications. Ordered according to their closeness to conventional von Neumann computing these are : SpiNNaker, TrueNorth and BrainScaleS.

SpiNNaker, now also pursued as part of the Human Brain Project, is a many-core system with a packet routing architecture highly optimized for efficient spike transmission. It is effectively a massively parallel real-time simulator with 500.000 ARM cores that can be programmed by standard programming methods. Individual processors execute standard program code but their communication is event-driven and asynchronous like in the biological brain.

TrueNorth maintains the use of purely digital circuitry but gives up the concept of algorithm controlled microprocessors. As such, it is a non-von Neumann computing machine. Specific digital neural circuits have been designed to implement one particular neuron model (LIF) and very basic synapses. The advantage of this approach are a very high degree of compactness and easy scalability to advanced process nodes. The system has very low energy cost per neural operation. As TrueNorth is a non-von Neumann machine without traditional programming models it needs specific software for configuration. Also, the simple neural functions implemented in neurons and synapses are far from biological realism. The system found several high-profile users in particular among government agencies in the US.

The third approach, also pursued in the HBP, is called BrainScaleS. This system makes the largest step away from conventional digital computing towards substantial biological realism implemented as mixed-signal circuits with analog neurons and synapses communicating with binary spikes in continuous time. A 2-equation neuron model (AdEx) developed in the previous FACETS project with programmable parameters and synapses featuring short and long-term plasticity constitute the neural components of the system. Neurons can receive up to 16.000 synaptic input which makes them ideal models for cortical circuits. Like TrueNorth this system needs a custom software stack for configuration.

Currently, all 3 systems have to be trained using software algorithms running on conventional computing systems. Training (or learning) is an important time consuming factor for neuromorphic systems. If different time scales from synaptic plasticity to structural changes in developmental processes are involved, a relative factor of a Million or more has to bridged in the artificial system. This is the single most important problem for conventional computer simulations which run much slower than real-time. With a slow-down factor of 1000 or more, it is impossible to simulate just a single day-equivalent of learning. SpiNNaker and TrueNorth are real-time systems. This is an important step forward and of special importance for robotics applications which require real-time

performance. The BrainScaleS physical model implementation is special in that it has tuned all analog system parameters like conductances and capacitances to accelerate learning and emulation by a factor 10.000 with respect to biology. This is an important asset for slow learning but essentially excludes use in real time robotics applications.

Current use cases for neuromorphic hardware are mostly demonstrators for basic neural micro-circuits like winner-take-all, attractor memory, synfire-chains, neural sampling units, AI states, cortical volume models and others. Applications in machine learning benchmarks are coming up at this moment in time. IBM has recently published a paper implementing deep networks on TrueNorth but with the entire learning process carried out offline.

Neuromorphic computing in the HBP is building on 10 years of work in previous projects. Among those are the EU-FET funded projects FACETS, Brain-i-Nets and BrainScaleS as well as the national British project SpiNNaker. Most effort in the RUP was invested in ramping up the small prototype systems to full scale platforms. This was a major engineering work involving production and manufacturing of several 1000 printed-circuit boards, testing, cabling, system debugging and development of support firmware and software. In parallel, first prototype chips of the 2nd generation where designed, produced and partly tested. The 2nd generation design work received crucial inputs from SP11 (and SP4) in the HBP as described below.

Contributions and value of SP11 (WP11.3)

Subproject 11 in the HBP RUP was created to come up with very early applications of the basic research and technology work carried out in the HBP. In-line with the overall objectives of the HBP, applications have been created in the areas of future neuroscience, future medicine and future computing. Here, we give an evaluation of the application work in future computing. A detailed description of the results obtained in the corresponding tasks can be found in the following chapters.

The idea of early applications turned out to be a challenge for the following 3 reasons :

- With one exception (Task 11.3.1) all tasks started with a 6 months delay after project start because they where selected in an open call process.
- The selection in the open call was carried out strictly according to scientific and technological excellence of individual proposals and not with the goal of a broad coverage of application areas. Still, the coverage is surprisingly good with a slight overrepresentation of vision related applications.
- Application development was carried in parallel with the construction of the platform machines. As the generation 1 neuromorphic chip concepts were basically ready (as a result of previous projects), the hardware and software groups in the hosting subproject SP9 had to focus mostly on actual platform construction work rather than developing their basic design criteria.

Still, all involved groups consider the joint work of SP11 and SP9 a very good success. All video conferences and physical meeting where held together and the input from SP11 to SP9 was very effective in the following areas :

- Exploring and further developing the software tools to configure the HBP neuromorphic systems and analysing their data.
- Pointing out flaws in the workflow.
- Acting as early adopters (and multipliers) of the small-scale systems available from SpiNNaker and BrainScaleS.
- Providing inputs to the design of the two early 2nd generation prototype chips developed and partly tested during the RUP.

The inputs to the design of 2nd generation prototype chips received important contributions from SP11 in terms of the following specific requirements :

- The importance of a floating point unit in the SpiNNaker-2 ARM cores to implement more complex neuron models and plasticity rules.
- The importance of an on-chip random number generator on SpiNNaker-2 for stochastic computing.
- The decision on synapse precision required for the BrainScaleS-2 system.
- The importance of parameter stability for the BrainScaleS-2 system.
- The features and capabilities of the plasticity processing unit (PPU) on the BrainScaleS-2 chip.

The specific work of the SP11 groups was carried out in 5 areas covering key applications for future neuromorphic computing :

Data mining

The development of neuromorphic data mining systems was successfully studied in two major use cases: Predictive maintenance and the application of neuromorphic compatible classifiers to non-uniform memory access (NUMA) scheduling in a combined online transaction processing (OLTP)/online analytical processing (OLAP) In-Memory Database.

Multivariate data classification

The implementation of spiking classifier networks was based on previous work carried out on the Heidelberg spiking neural network chip Spikey. The classifier design has been modified and adapted for implementation on different neuromorphic platforms. The researchers analysed how each implementation performs differently with varying model sizes (up to 30,000 neurons, 18 million synapses), with different number of classes (2-10 digits), and when incorporating increasing numbers of “virtual receptors” (VRs) in input space (20 – 500+). The second part of the work focused on using a large model on higher capacity “large-scale” hardware.

Robotics

Porting the Cell Assembly Robot (CABot), a video game agent, to neuromorphic chips has been carried out for two major uses cases: the SpiNNaker agent; and, a cognitive model of categorisation model. The agent, though simple, works. The cognitive model is being published and categorises like the brain’s putative explicit system.

Visual system I : Convolutional networks

The exploitation of feedback in ultra-fast spiking visual architectures has been implemented as an event-driven Convolutional Neural Network architecture with optimized parameters to perform recognition of the card that has been programmed on a 4-chip Spinnaker board.

Visual system !! : Communication with spiking neurons

Asynchronous computational retina work developed a pure, event-driven visual computational approach that uses precise timing mechanisms to design new computational techniques in visual processing. The task produced a full event-driven visual processing system linking a neuromorphic retina directly to the SpiNNaker system by an Asynchronous Event Representation (AER) bus. The architecture allowed the first real-time development and implementation of new, visual, event-driven computation techniques.

Spike-based memory architectures

The construction of spiking associative networks for neuromorphic computing systems led to the development of tools (spiking associative memory (SAM) generation, benchmark

generation, PyNNLess, performance analysis, parameter optimization) that are now open to all HBP users. For the neuromorphic researchers these tools simplify the hardware access and systematic hardware testing. These tools are especially helpful for the design of the next generation of neuromorphic hardware within HBP.

Future development in the HBP with respect to applications

The HBP decided not to follow-up on the concept of a dedicated application workpackage in the SGA1 phase. Instead, applications of neuromorphic computing are now part of the SP9 subproject proper. Out of the 5 application areas described above, the following areas and groups decided to continue in the project during SGA1 :

- Multivariate data classification (Nowotny, Schmuker groups)
- Robotics (Huyck group)
- Spike-base memory architectures (Rückert group)

In order to address a wider audience the following measures are currently taken in SGA1 :

- Regular online application workshops
  https://www.youtube.com/watch?v=khRPnlDekIg
- Advertising the HBP neuromorphic platform through the open guidebook
  https://electronicvisions.github.io/hbp-sp9-guidebook/
- Alignment with the US activities, especially in the application area
  https://www.src.org/calendar/e006125/
- Strong involvement in the co-design project 5 on learning and plasticity
- Invitation of groups to work on applications as partnering projects.

Concerning the last item in the list : Partnering projects (PPs) are an important asset of the HBP. The allow to attach groups or projects with their own funding as formal partners. Applications of neuromorphic computing appear to be an ideal case for PPs as funding for applications within the core project will always be limited to a few cases. SP9 will therefore actively search and invite European partners to join HBP as PPs working on applications of neuromorphic computing.

In the SGA2 (starting April 2018) applications will be joined with benchmarking studies and a dedicated workpackage will be created within SP9, building on the successful concept of the application workpackage 11.3 in the RUP :

Workpackage 5 : Applications and Benchmarks		Michael Schmuker
Task 9.5.1:	Neuronal signal processing	Michael Schmuker
Task 9.5.2:	Agent and Agent Components	Chris Huyck
Task 9.5.3:	Object recognition	Thomas Nowotny
Task 9.5.4:	Benchmarking	Ulrich Rückert

The alignment with the technology development in SP9 will be very close and has already started in the current SGA1.

In the following paragraphs the specific work carried out in the application tasks is described.

4.2 T11.3.1 (SAP): Neuromorphic Data Mining Systems

This report describes the results achieved at the end of the 30 months ramp-up phase of the HBP project. The activities were guided by the approach to identify neural networks

able to deal with real-world problems, but also simple enough to allow an implementation on a neuromorphic hardware system. These neural networks then need to be adopted in order to be able to process spikes. This is a prerequisite for a later implementation on the hardware systems. A major decision in enabling a classical neural network to process spikes is to select an appropriate encoding scheme. Since networks of spiking neurons are dynamic systems with a complex temporal behaviour, we decided to apply temporal coding and encode the information in the precise spike time of the neurons. In contrast to this, many algorithms are currently based on rate models.

When the decision on the coding scheme is made, the next question is how to modify the synaptic weights in order to implement learning and to retrieve a previously stored weight value. The hardware systems implement the Spike-Timing Dependent Plasticity learning rule which will then be the preferred way to adjust weight values in an on-line mode. Off-line learning would also an option, but to exploit the full advantages of adaptive systems on-line learning will be preferred.

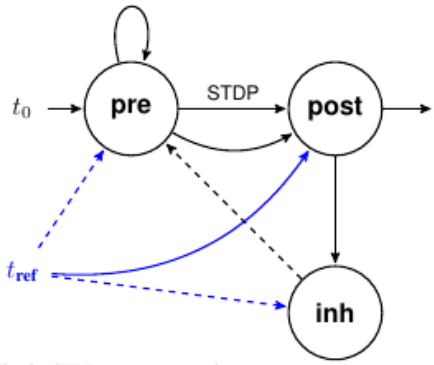

Figure 14: Minimalistic STDP memory network as described in the text.

STDP will lead to weight changes based on the observed spike patterns and change the previously learned spike times, e.g. increased weights will lead to earlier spike times. To address the topic of spike time encoded memory, we implemented a “memory cell” utilizing STDP. The memory cell is depicted in Fig.13. It basically consists of the 3 neurons. An initial start signal t_0 causes the pre neuron to spike, which in turn loops itself to keep spiking with a well-defined frequency (alternatively, a burst representing a high value could be injected in to pre). We found that this produces a short clocklike spike behaviour for a small time range.

These spikes are transmitted through the STDP synapse and excite post. The neuron post will therefore spike at different points in time, depending on the currently stored weight. A spike of post excites an inhibiting node immediately, causing inhibition of pre after a short period of time which leads to one last spike of pre after post spiked. Inhibitions are depicted by dashed lines. Details will be described in an upcoming publication.

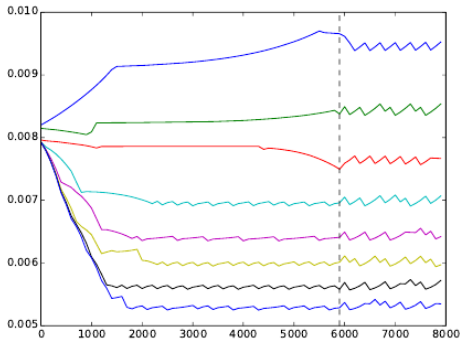

Figure 15: Simulation of a single memory network.

This figure shows the plotted weight of the single STDP synapse for 8 different training scenarios. An iteration lasts for 100 ms. Until 5900 ms or 59 iterations, a reference signal is presented for training. From 6000 ms onwards, the system is in a "retrieve phase".

This small network of neurons exhibit some interesting features as it produces stable states around certain weight values, leading to well-defined spike times. These desired spike times are induced via the injection of a training signal (t_{ref}). Updates of the weight values are performed by STDP during the training phase, STDP remains active also in the retrieval phase, leading to small variations around the stored values. The diagram above (Fig. 14), depicting the result of a simulation leading to 8 stable states.

We plotted the simulation of a single learning component over 8000ms, thus 8 seconds, using training with a reference signal an STDP. It shows the oscillation behaviour and the resulting 8 learnable intervals for storing values of (13,14, …,20) ms. Note that all target values are already reached around 2000ms, thus 2 seconds or 20 iterations (100ms each). After that, the system remains in the corresponding interval with only little oscillation. At 6000ms onwards, the system is running in retrieving mode. The oscillation variance grows, as it is expected from an unsupervised system, but still stays within small boundaries.

These memory cells can be block wise combined to form bigger units and to implement networks with a larger number of states. E.g. combining three units will extend the range of trainable spike times from (13,.., 20) to (13,..,36) ms.

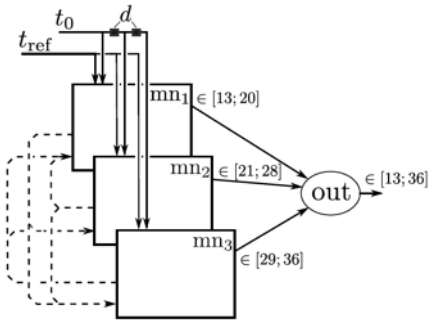

Figure 16: Blockwise composition of three memory networks denoted as mn1 to mn3.

Once these memory cells become available, they can be used to construct larger networks like the one mentioned in the previous report, the CMAC network. The classical (non-spiking) model can, with the concepts described above, be transferred to a spiking version of the network.

Implementations on the neuromorphic hardware systems are ongoing. In the meantime we managed to implement a preliminary version of the memory cell on the physical-model system and could identify stables states leading to well-defined spike times.

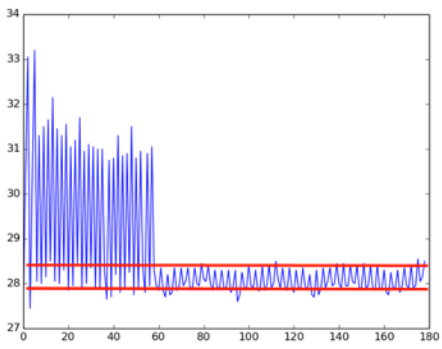

Figure 17: Preliminary results of the memory network implemented on the Spikey chip. Shown is the "retrieve phase" for a training scenario leading to a stable state after a "burn-in" time.

In the experiments depicted in the diagram (Fig. 1), we created a stable state leading to a spike time of about 28 ms with fluctuations < 1ms. An implementation leading to more states, similar to the simulation described above, is underway. This will be the prerequisite for the implementation of larger networks.

4.2.1 Description and validation of software models for network architectures in neuromorphic hardware

The cerebella model articulation controller (CMAC) has been introduced as a computational model of the mammalian cerebellum. Developed as an associative memory, it is capable of approximating a continuous nonlinear function from samples of the function's input and output similar to a multi-layer perceptron (MLP) however only a small fraction of the network determines the actual output. Its focus lies on local generalisation: similar inputs produce similar outputs while distant inputs produce nearly independent outputs. This leads to very efficient and fast learning making it suitable for real-time adaptive control applications.

The network described here implements a spiking version of the traditional CMAC. It applies two different encoding schemes, population coding and temporal coding, respectively. The non-linear transformation, f, is performed similar to the classical CMAC algorithm leading to a spatial representation of the input signal in the hidden layer. The main characteristic of the transformation remains the same, i.e. similar input signals will generate similar representations in the hidden layer. These connections are static and will not change. In the case of random connections, a soft winner-take-all (sWTA) circuit could be applied to subsets within the hidden layer selecting the L cells with the highest activation potential.

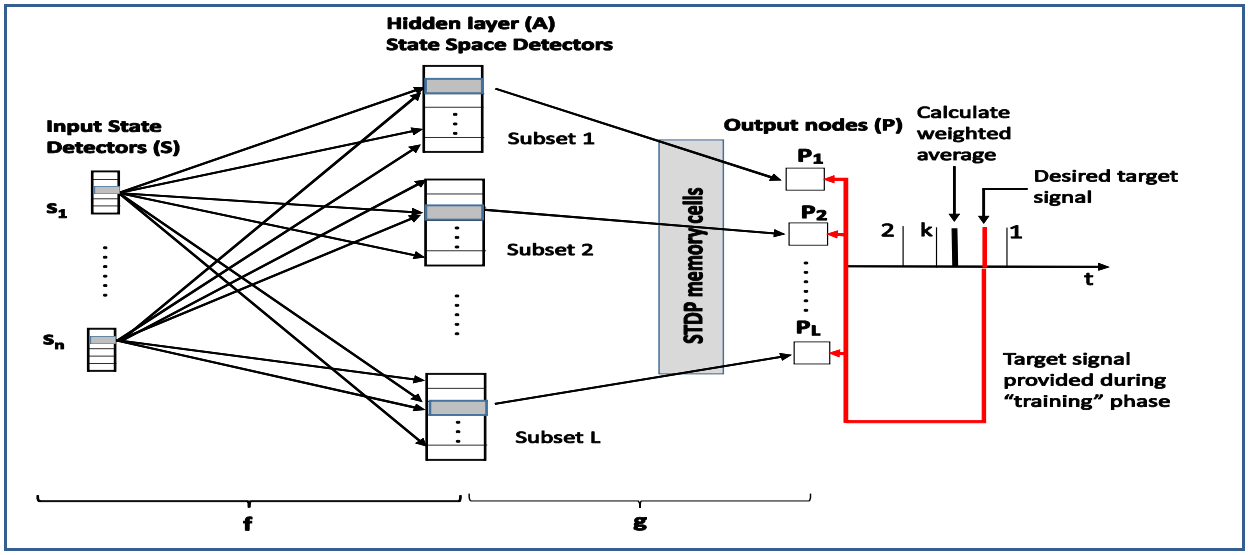

Figure 18: Proposed spiking CMAC network

The proposed network has temporal coding in the output layer g, while the transformation f is similar to the original CMAC wiring. The total output of the network is the average of all intermediate values Pj of the (at most) L active hidden neurons for a given input. Note that we use the previously described memory primitive for storing and training the connection of the hidden layer to the output cells.

A state in the hidden layer is represented by L active cells, one in each of the sub-layers. Due to a fixed delay between the input and hidden layer, the activated cells will all spike at the same time, t_0. This defines a reference point for the temporal coding which will be used to encode the information for the linear transformation between the hidden layer and

the output layer. In contrast to the classical CMAC network, the temporal CMAC has L output nodes, each fully connected to a subset of the hidden layer. Every subset triggers a single spike in the output layer and the specific spike time depends on the corresponding weights.

We now apply two different training rules to adjust the weights and the corresponding spike times: one is using STDP, the other is based on the calculation of a "trend signal". For every training iteration, the spike time t_j - t_0 ($j \leq L$) of the output cell of each sub-layer is shifted towards the target. In spiking models, this is encoded as synapse weight adjustments, as heavier weights lead to earlier spikes of the post connected neuron. The final output of the network is the weighted average of the spike times of active cells in the output layer: Result $= \sum_j^L t(w_j * a_j)/L$.

Cells which are involved in the training of a single target value will approach that value after a number of training iterations. Due to the generalisation property, some of the cells will participate in the training of more than one target value and the learning rule will lead to a spike time representing the weighted averages of those target values as they occur in the training data. The summands will therefore either be precise target values or weighted averages of the target values of neighbouring states avoiding extreme contributions as can be the case for the original CMAC. This will ensure good generalisation compromising the approximation accuracy. These models have been tested as part of the milestone MS211 in extensive software simulations as a candidate for an implementation on the neuromorphic hardware. Results of the PyNN simulation are shown in Fig. 19.

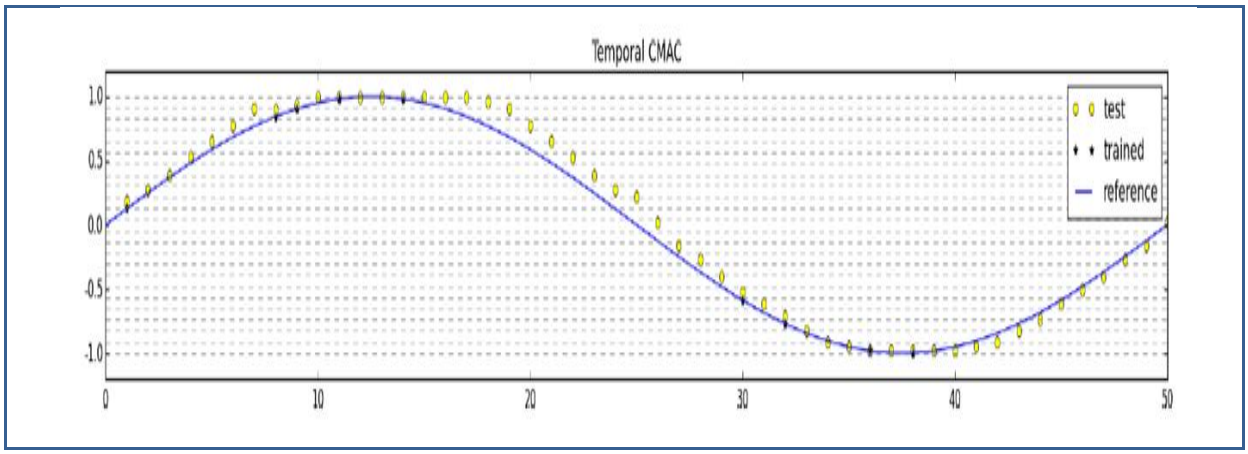

Figure 19: Single prediction simulation of the proposed temporal CMAC.

Dashed lines mark the 16 distinguishable values by the memory primitive. The blue line depicts the target values, a sinus. Randomly sampled training points are depicted by stars. Finally, every possible point is tested, indicated by yellow dots.

Parts of the network could be implemented on the Spikey system during activities leading to MS212 (Implementation of test networks on the HBP Neuromorphic Platform).

4.2.2 Implementation of network architectures in the NCP and defined benchmarks

Both use cases are selected to exploit the benefits of Neuromorphic Computing. The predictive maintenance use case builds on the implementation of a neural network in hardware with the goal to enable real-time processing of data streams using the intrinsic parallelism of neural algorithms and corresponding hardware implementations. An important factor (at least for some of the application areas) will be the low power consumption. A benchmark for the software implementation is the data set described in the Prognostics Data Challenge Dataset (PHM08 / NASA) and the results of the competition published on the respective web site. The software model based on an implementation of

the CMAC as described above achieved results comparable to other (more complex) methods. A hardware implementation based on this data set is at the current stage beyond the scope of the Spikey system. As a proof-of-concept we implement a regression problem (approximating a non-linear function) in PyNN.

The NUMA aware scheduling in the in-memory data base will only be possible with a very fast classifier, implemented in HW and located in proximity to the CPU. The benchmark is the performance achieved by a set of heuristic rules which works well under "normal" operational situation, however performance decreases significantly under higher system load. Experiments are ongoing to benchmark the performance of a classifier against the set heuristics under varying load conditions.

4.2.3 Description of major use case(s) and target users of your application

We looked at two different use cases, (1) Predictive Maintenance and (2) NUMA aware scheduling in an OLTP/OLAP in-memory data base.

Predictive Maintenance

The previous science report mentioned the activities to identify an application of neural networks in the area of predictive maintenance. It could be shown that a classical neural network, based on model described by Albus in 1975, is on the one hand able to deal with complex real world data sets, but on the other hand simple enough so that it can be implemented on the neuromorphic hardware systems of the NCP.

The following diagram shows the result of the simulation, based on a publicly available data set (Prognostics Data Challenge Dataset, PHM08 / NASA). The results are comparable with results achieved by other (more complex) methods. An implementation in low-power hardware could potentially allow to move such prediction functionality into the devices. IDC (IDC FutureScape: Worldwide Internet of Things 2016 Top 10 Predictions) predicts that by 2019, 45% of all data created in Internet-of-Things scenarios will not be processed and stored at backend systems, but rather analyzed by devices at the edge of the network and directly acted upon locally. This could lead to significant benefits in terms of latency, reduction of data transmission, required network bandwidth and data storage.

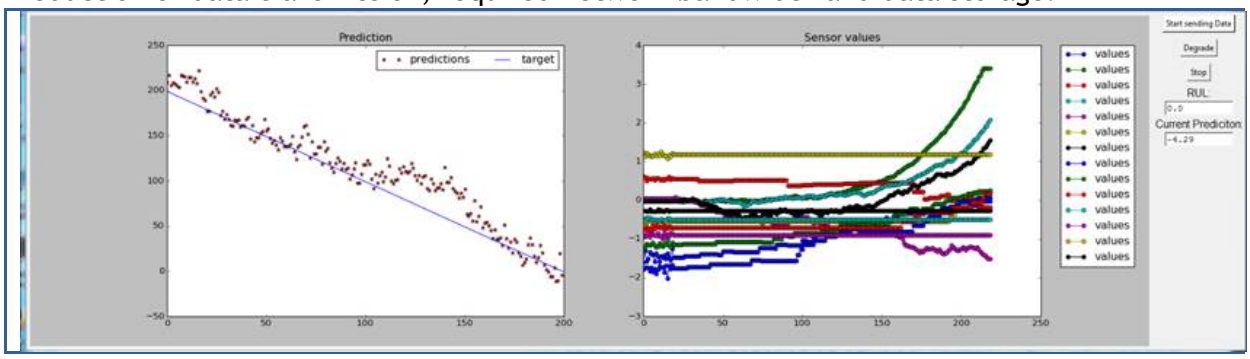

Figure 20: Simulation of a predictive maintenance scenario.

The left diagram shows the target function (remaining useful life) in days and the prediction by the CMAC network. The diagram on the right shows the behaviour of the selected 14 sensor values.

The goal of the activity was to model the CMAC algorithm as a spiking neural network suitable for an implementation on the NCP. The selected approach is to take conventional neural network structures and adopt them to the spiking methodology. This is a prerequisite for a later implementation on the hardware systems.

Application of neuromorphic compatible classifiers to NUMA scheduling in a combined OLTP/OLAP In-Memory Database

The SAP HANA in-memory database system is a very high performance system underlying most of SAP's newer solutions. To achieve consistently adequate performance without

having to design special purpose data access paths, clever low-level implementation techniques are employed that use robust, mostly brute-force algorithms that are fairly insensitive to variations in the data or effectiveness of traditional query processing optimisations. Queries are split into small jobs that are executed in parallel wherever possible. Many of these jobs are bottlenecked on access to data rather than computation, which makes relative placing of data and query processing jobs in machines where processors and memory are segregated into multiple interconnected clusters a difficult optimization problem.

Dynamically scheduling jobs onto a modern machine with hundreds of cores and thousands of concurrent jobs is challenging (optimal scheduling is NP-hard and actual hardware exhibits dynamic behaviour that introduces uncertainty and noise into the optimisation parameters) and is infeasible to perform in software at the required speeds.

“Optimal” scheduling in this context simultaneously tries to achieve best throughput (by utilizing as many of the resources in the machine in parallel as possible) as well as limiting the waiting time for completion of individual queries (which translates to the user-perceived response times). Traditional databases split workloads of long-running OLAP (online analytical processing) queries and the short-running, but frequent OLTP (online transaction processing) queries between separate systems (OLTP database optimized for transactions, Business Warehouses for analytics). This approach not only doubles the systems count, but also leads to delays and inconsistencies between the systems. HANA tries to handle both types of query in a single database. For efficient resource utilization for OLAP, one wants close to 100% resource utilization, and for good interactive performance traditional OLTP databases have to limit utilization to around 40%.

A “good” scheduler for such a mixed workload system will try to limit the latency impact of the resource hungry, longer queries on the shorter ones. For our experiments, we selected a set of nine “solvable” workload traces and computed an optimal schedule (i.e. the best one could do if one knew the future outcomes of current scheduling decisions as well as the future queries coming in). “Solvable” means that schedules exist that let all queries meet latency requirements and offer good throughput. Of course, completely overwhelming a machine with too much work means that work eventually becomes impossible to schedule and has to be solved by admission control. The goal of a good scheduler is to push the cut-off point where admission control has to shed workload from the typical 40% resource utilization point to as close to 100% as possible.

Known heuristic schedulers either optimize for fair resource usage (round robin) or try to quickly get rid of queries (greedy), which lead to good minimal response times but sometimes to terrible worst-case latencies and eventual starvation, even when a better schedule could have avoided the problem.

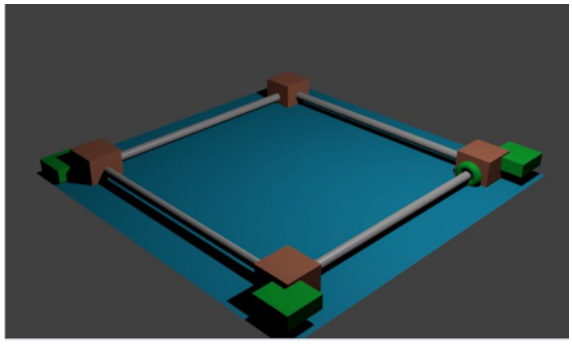

Figure 21: Illustration of CPUs with attached memory modules and interconnects

Our neural scheduler approach uses supervised training of classifiers that are implementable on neuromorphic hardware to approximate optimal scheduling decisions in real-time.

The scheduling problem is reformulated as a classification problem by presenting the dynamic state of the system (load factors, queue lengths, etc.) together with the parameters of each job to a classifier. Supervised training happens under the guidance of a full system simulator that can calculate dynamic system parameters from first principles and can also simulate future outcomes of job placement decisions accurately.

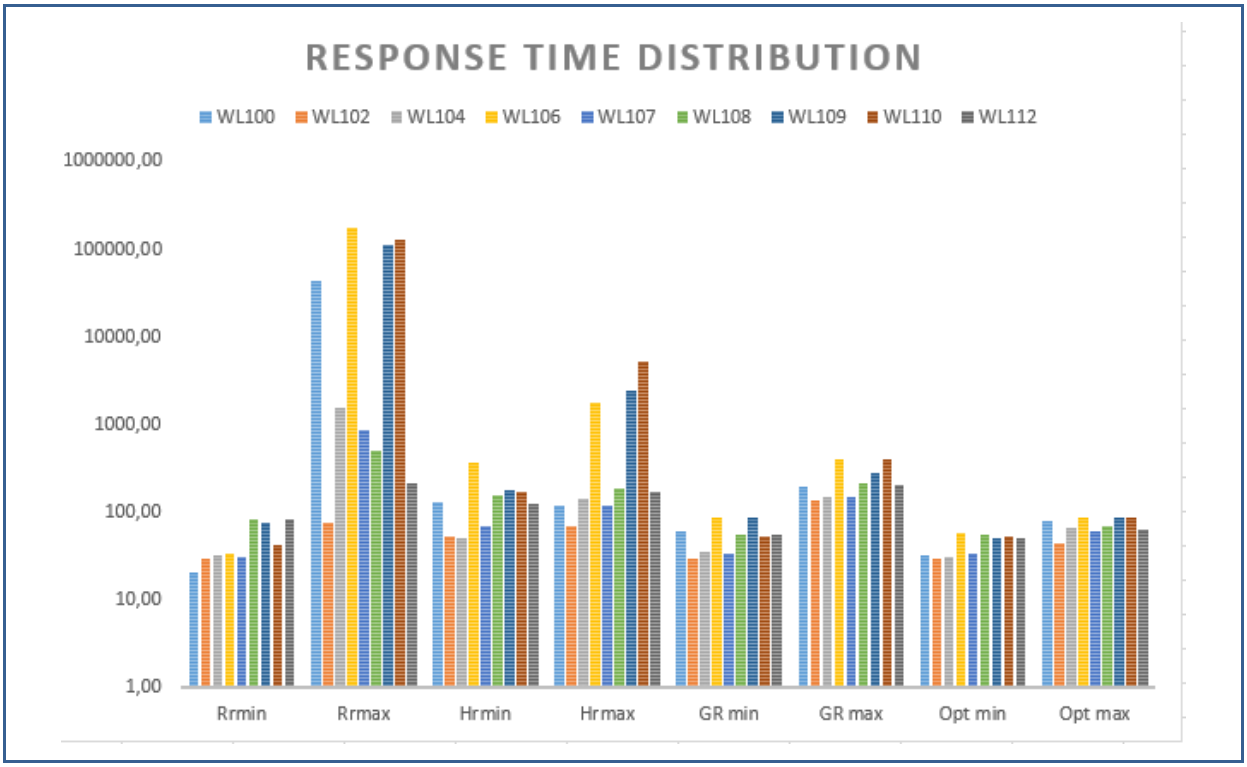

Figure 22: Distribution of minimum and maximum query response times across 9 different workload traces (WL100-WL112).

WL traces are based on the same queries and contain the same amount of overall work, but differ significantly in inter-query arrival times and clustering of query types. In particular, WL100, WL106, WL109 and WL110 have significant clustering of queries towards individual resources. The four different schedulers (*Rr* for naïve round-robin, *GR* for greedy, *Hr* for our neural scheduler, and *Opt* for the omniscient "teaching" scheduler used to train the *Hr* scheduler. Note the huge worst case latencies for both the round-robin and greedy schedulers.

We compared the classifier-based scheduler on workload traces of 100,000 jobs each with different degrees of skew and saturation of bottleneck resources. Performance on workloads far off saturation or with even load distribution was indistinguishable (i.e. marginally better or sometimes worse) than either a naïve round robin heuristic or a somewhat more sophisticated "greedy" heuristic scheduler. Performance on heavily skewed workload (i.e. "hotspots" in resources) typically come within a factor of two of the (unachievable in practice) optimal schedule calculated by the "omniscient" scheduler, vastly outperforming naïve round-robin scheduling. Performance relative to the "greedy" heuristic scheduler was comparable for many of the workloads, but the classifier-based scheduler does a better job in managing the workload for all cases without exhibiting "long-runners". More follow-on work will be required to either better characterize the workload or engineer features to be better able to capture the relevant dynamic aspects of the workloads.

The workload traces used for the experiments are available publicly at

https://www.dropbox.com/s/a22dkvn9epj0735/HBP_4.1_NUMA_scheduling.zip

The application Work Package WP11.3 ends with the completion of the HBP Ramp-Up Phase and therewith SAP's involvement. Bi-lateral collaboration models are currently being discussed.

Table 4: Status of use cases for the NCP

Use Case	Data Set	Classification and clustering	Algorithms and Benchmarks	Implementation status on NMPs	Comments
Predictive Maintenance	PHM08/NASA for the software simulation, sinus curve for the PyNN implementation	Regression	CMAC	Model implementation in PyNN, parts of the memory cell on the neuromorphic HW system	Full data set to complex, therefore sinus curve selected for regression
NUMA scheduling in in-Memory DB	Generated by software model	Classification	CMAC and others	Only software simulation at this stage	Preliminary evaluation finished

4.2.4 Documentation of software models and network architecture

The software model and network architecture will be described in an upcoming publication (Online Trainable Spiking CMAC Network, by Björn DEISEROTH, Ulf BREFELD, Christian DEBES, and Frank GOTTFRIED).

4.2.5 Outreach

SAP is currently preparing a paper describing the work in detail, to be submitted shortly. The results were presented during a talk as part of the "Gesprächskreis Rhein-Neckar" which took place on 28.01.2016. SAP participated in all meetings of the Neuromorphic Computing Division (SP9/WP11.3) within the reporting period.

4.3 T11.3.2 (MU): Port CABot3 to neuromorphic chips and extend

Table 5: Status of model types and networks in WP11.3.2

Model Type/ Network size	Simulation Software	Implementation Status (NM-PM1, NM-MC1)	Parameters evaluated	Comments
Validate Models	PyNN, Nest, SpiNNaker	Tested existing models on all platforms. Developed new neural and synapse models on Nest and SpiNNaker	Spikes appropriately. Simple learning works.	Initial work was slow. Later work has been slow.
Virtual Agent	PyNN	NM-MC1	Full CABot3 agent	Closed Loop
Virtual Agent	PyNN	NM-PM1	Code runs remotely but no agent.	

4.3.1 Describe and validate neural and synaptic models for neuromorphic systems

Initially we needed to get things running in PyNN and SpiNNaker. We needed to use the standard neural models. Initial work with Izhikevich neurons failed because they could not

be inhibited without spiking. Subsequent work with a range of IF models worked, along with use of the standard plasticity model (STDP). Unfortunately, we could not bind with STDP, needing a form of short-term potentiation; consequently, we could not bind during Natural Language Parsing.

In later work (end 2015), we developed our own neural and synaptic models to run on Nest and SpiNNaker. We have developed a fatiguing leaky integrate-and-fire (IaF) model, a short-term plasticity model, and a compensatory long-term plasticity model.

CABot4 (MS309, M30) has not been completed. While a neural implementation of a cognitive model of learning has been developed, this has not been integrated into a new SpiNNaker agent.

4.3.2 Implementation and performance of virtual agent and environment on neuromorphic systems

The virtual agent runs on SpiNNaker. It is a simple agent in a simple environment, but can explore the environment, remember the coarse structure, respond to natural language commands, view the environment, and execute simple plans. It runs on both 4- and 48-chip systems. For the full test, remembering the environment, current performance is around 75%.

We wrote the virtual environment using Python and Tcl/TK. The environment had an agent that could perform four primitive actions: move forward, move backward, turn left, and turn right. The environment consisted of four rooms connected by four corridors. Each room had a unique shape in it: a red or blue pyramid or stalactite (up facing and down facing tetrahedrons).

The agent, implemented on SpiNNaker in simulated neurons, responded to natural language commands typed in by a user. The agent was implemented by several subsystems, with the coarse topology shown in Figure 23. Subsystems could be tested in isolation, and used in the full agent. The Natural Language Processing (NLP) subsystem was implemented by a Finite State Automata to parse the predefined commands. This subsystem always succeeded both in isolation and in the full agent.

The vision subsystem takes inputs from the virtual environments camera as pixels. This can vary in size, and has been tested on 20x20, 30x30, 40x40, and 50x50 as input. A simulated retina converts the pixels to a series of on-off and off-on receptors with 3x3, 6x6 and 9x9 receptive fields. The outputs of these are passed to four angle, and eight edge detectors. All of the receptors and detectors are the same size as the input. So, with larger inputs performance is better, but the number of neurons increases rapidly. Finally, there are three 20x20 colour detectors (red, blue and green). These inputs are then passed on to object recognisers for recognising the four shapes and the corridors. These then activate vision facts, indicating that the appropriate object is in the visual field, and whether it is left, centre or right. It is difficult to measure performance with this, since it is not clear if (for instance) a one pixel pyramid is actually present in the environment. This largely works correctly with 20x20 input, and there is improved performance as the picture gets bigger. We have not noticed any false positives.

The planning component is a spreading activation net derived from Maes work. There are goals (set by the NLP subsystem), modules, facts and actions. Actions are single neurons and the virtual environment interprets a firing of one of those neurons as an action.

The cognitive map component memorises the configuration of the rooms. It associates the shape with the room. This supports the most complex command the system executes. There are four including “move to the room before the red pyramid.” The system must navigate between the rooms, and look for the room before the room that has the red pyramid.

The control subsystem supports the interplay between the NLP subsystem and the goal subsystem. It is quite simple, and has not been seen to fail.

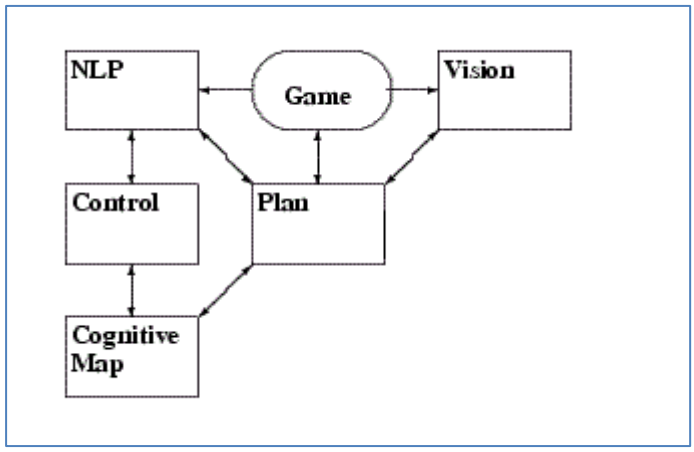

Figure 23: Gross Architecture of Agent. Boxes represent subsystems instantated in simulated neurons. Arrows represent synaptic connections between subsystems.

One of the major challenges was the communication between the agent and the SpiNNaker board. We developed systems for input to the board for vision and natural language and an output system for actions. Input relied on pixelating the image, and sending spikes if the pixel was over a threshold; this yielded a black and white map that was tested in several sizes. A parallel colour system was set up that thresholded the image for red, green and blue, but always in 20x20. The natural language parsing caused one neuron on the board to fire for starting the parse, and then one for each word. The output just read the spiking behaviour of the four action neurons. A real challenge for this was the timing. There was different behaviour if an input vision neuron fired once every 5 ms, to if it fired once every 10 ms.

The overall system executes thirteen commands. There are four primitive commands, two compound commands, two simple closed loop commands, the explore command, and four cognitive map testing commands. The four primitive commands include one for each primitive action, e.g. “move forward”. These performed perfectly every time we tested them. There are two compound commands: “move left” and “move right”. These require the agent to turn and then move forward. These also perform perfectly. Note that these six commands do not require an environment, and can function without a closed loop. The simple closed loop commands are “turn toward the pyramid” and “turn toward the stalactite”. If the vision system has recognised the shape on the left or right, these always perform correctly. The “explore” command causes the system to fill in the cognitive map. This is quite complex as the agent has to move between the rooms. The communication problem with the board causes some problems, and this works about 90% of the time, when the agent starts facing the shape with the door on its right. The tests of the cognitive map are similarly, but slightly less effective. These four commands are executed correctly about 80% of the time when the map is correctly filled in. An example of one of these runs can be found in the figure below.

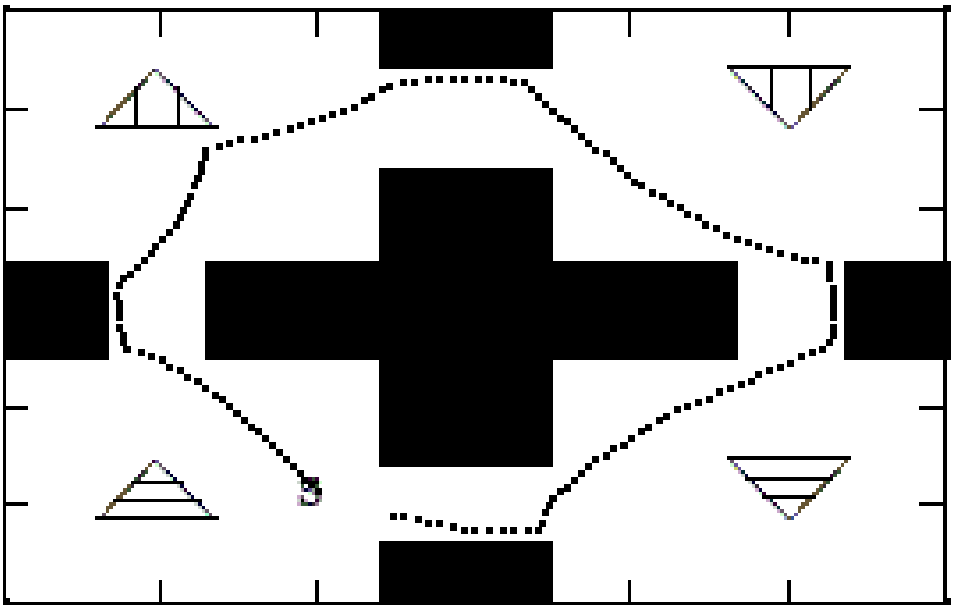

Figure 24: Top down view of the path of agent through the virtual environment while learning the cognitive map.

The agent starts at S, and moves are marked by the dots.

The full system runs on the 4-chip SpiNNaker board with 20x20 visual input, but needs the larger 48-chip board for the larger visual fields. Aside from vision, the other subsystems have a constant number of neurons.

NM-PM1 use was restricted to accessing chip over the internet. We managed a persistent cell assembly, which is the basis of most of the agents formal processing. A complete agent was not viable.

4.3.3 Description of major use case(s) and target users of your application

There are two major uses cases: The SpiNNaker agent, and a cognitive model of categorisation model. The agent, though simple, works. The cognitive model is being published and categorises like the brain's putative explicit system.

The cognitive model was for classification, and was based on classic psychological work from Shepard, Hovland and Jenkins (Learning and memorization of classifications, 1961). The system was developed in our own neural simulator with fatiguing leaky integrate and fire (FLIF) neurons and using compensatory (Hebbian) learning. It was a binary categorisation task, with each of eight items represented by three binary features, for example, black vs. white, large vs. small, and triangle vs. square. Each category consisted of four items. While there are 70 possible categories, these can be broken down into six types. For example, one type would categorise based solely on one feature (e.g. black vs. white or large vs. small), and another type was based on two features (e.g. black triangles and white squares vs. white triangles and black squares). Human subjects learn different types of categories more rapidly and more effectively than others. Our cognitive model mimicked some of this behaviour, though not the full task. It was proposed that this echoed the brain's putative explicit system. This work is now in review at the journal *Connection Science*.

Both the compensatory learning rule and a FLIF neural model have been implemented in SpiNNaker and Nest. We hope to translate this cognitive model to these platforms shortly.

4.3.4 Documentation of models and related experiments

Website for the NEAL project is http://www.cwa.mdx.ac.uk/NEAL/NEAL.html. Code can be found there. A related DIC was filled in the Dataset_Information_Catalog/ Collaboratory.

4.3.5 Outreach

- Huyck, C., and Mitchell, I. *Building Neuromorphic Embodied Cell Assembly Agents that Learn*, International Conference on Biologically Inspired Cognitive Architectures, Cambridge, MA, USA, 7-9 November 2014.
- Huyck, C., Evans, C. and Mitchell, I. *A Comparison of Simple Agents Implemented in Simulated Neurons*. Biologically Inspired Cognitive Architectures, 2015. **12:** pp 9-19.
- Harris, J., and Huyck, C. led working group on Neuromorphic Natural Language Processing at the Telluride Neuromorphic Cognition Engineering Workshop 2015.
- Huyck, C., talk on Better Cell Assemblies at BICA 2015.
- Huyck, C. (2015) *Neural constraints and flexibility in language processing*. Behavioral and Brain Sciences, **38.**
- Mitchell, I., Huyck, C., and Evans, C. *PlaNeural: Spiking Neural Networks that Plan*. (2016). International Conference on Biologically Inspired Cognitive Architectures, NY, NYU, USA.
- Evans C., Mitchell, I., and Huyck, C. (2016). *Programming with simulated neurons: a first design pattern*. Conference on Psychology of Programming, Cambridge, UK.
- Huyck, C., Kulkarni, R., *A Spiking Half-Cognitive Model for Classification* (Connection Science, in review).

4.4 T11.3.3 (CSIC): Exploitation of Feedback in Ultra-fast Spiking Visual Architectures

Table 6: Status of feedback exploitation in ultra-fast spiking visual architectures

Model	PyNN Implementation	Implementation Status (NM-PM1, NM-MC1)	Benchmarks evaluated	Comments
ConvNets optimised event-driven	Finished	NM-MC1 (SpiNNaker)	Yes (see Section see Section 4.4.1.1)	
Completely event-driven	Finished	NM-MC1 (SpiNNaker)	Yes (see Section see Section 4.4.1.2)	
Standard SpiNNaker using event-driven model	Finished	NM-MC1 (SpiNNaker)	Yes (see Section see Section 4.4.1.3)	
Optimisation model	Finished	NM-MC1 (SpiNNaker)	Yes (see Section see Section 4.4.2.2)	
Mismatch evaluation model	Finished	NM-MC1 (SpiNNaker)	Yes (see Section see Section 4.4.2.3)	

4.4.1 Description and validation of the neuron models for symbol recognition

Unlike frame-driven neurons, time has a crucial role in an event-driven neuron model. The state of the neuron evolves with time. In our work we have used signed event-driven spiking neurons. In such neurons the state of a neuron x_j can take positive or negative values, and will trigger a positive output event when a positive threshold is reached, or a negative output event when a negative threshold is reached.

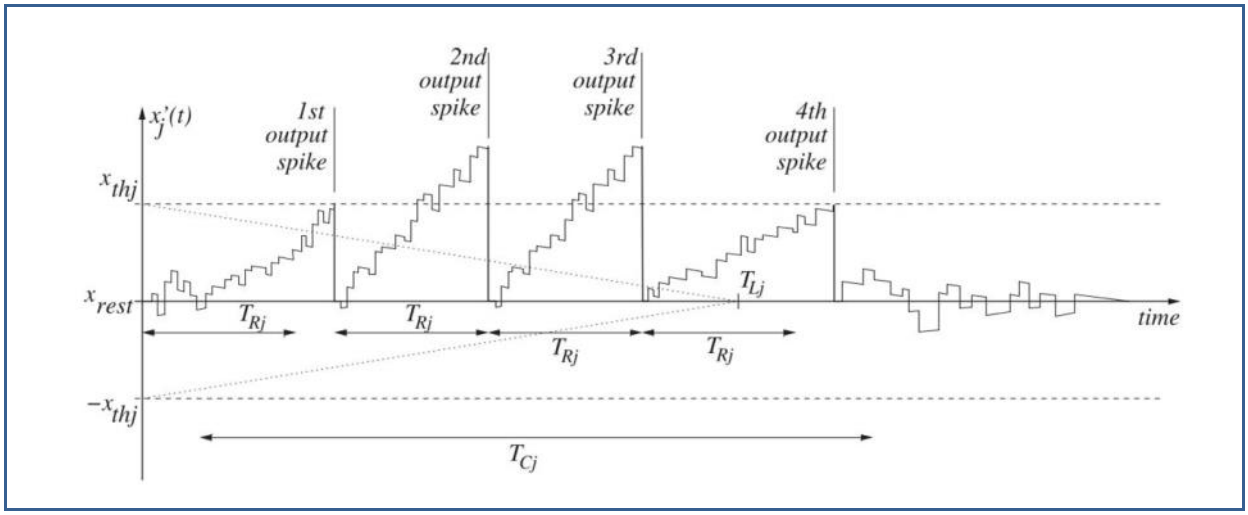

Figure 25 (a)Typical state evolution and spike production sequence for a signed spiking neuron with leak and refractory period

A positive output event will be produced when the positive threshold x_{thj} is reached, and a negative output event will be produced when the negative threshold $-x_{thj}$ is reached.

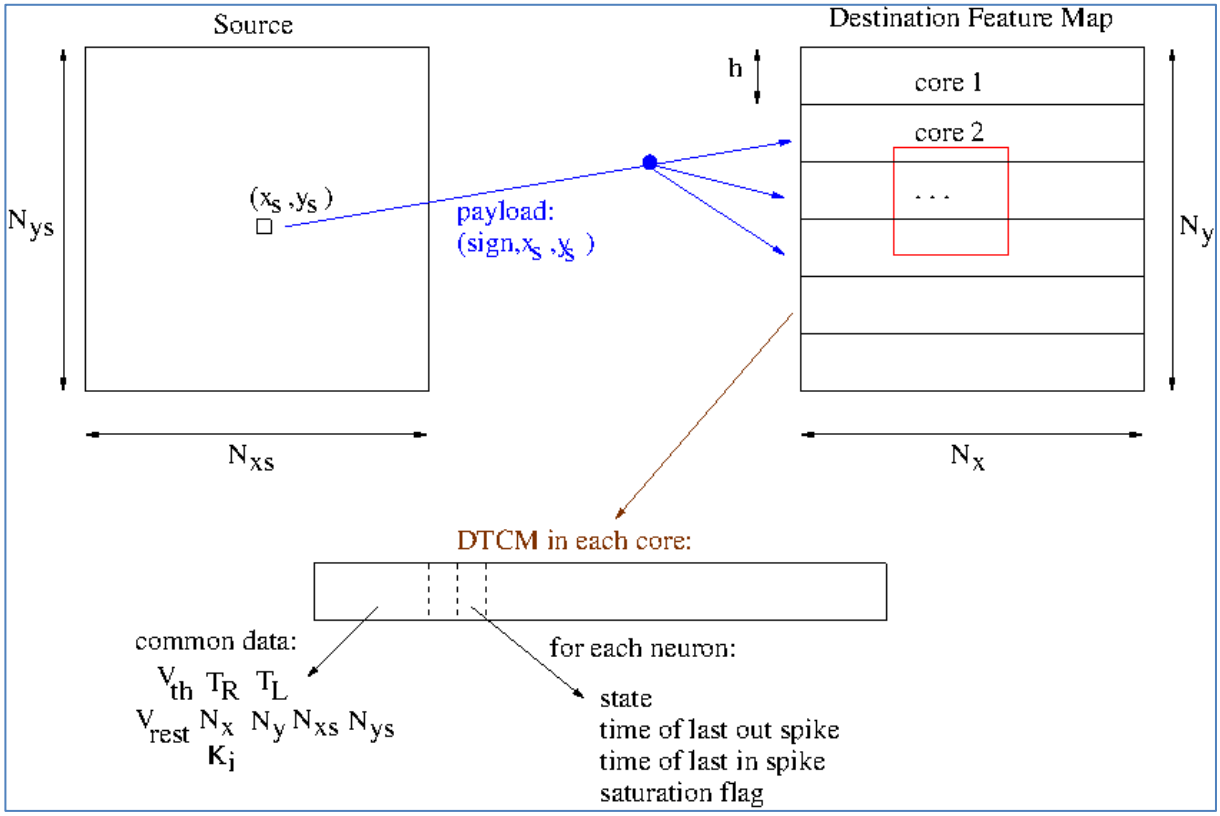

Figure 25: (b) Detail of compact ConvNet implementation on SpiNNaker.

Figure 25(a) shows an example of neural state evolution and spike production. Based on the presented neuron model, we have developed three different implementations using the SpiNNaker platform. In all the systems, the description of the multi-layer vision systems has been done in PyNN matching the **Function number 11.3.3.1**.

4.4.1.1 ConvNets optimized implementation

One of the peculiarities of a ConvNet (Convolutional Neural Network) architecture is the weight sharing property. The weights of the kernels that connect two neurons in two different feature maps do not depend on the absolute neuron position but just on the relative positions of the two neurons in the origin and destination feature maps. Because of that "weight sharing" property the number of synaptic weights that must be stored for a neuronal population is highly reduced compared to populations with full connectivity and independent non-shared weights. To optimize the processing speed, the original SpiNNaker support software PACMAN has been modified to admit a special "convolution connector". The "convolution connector" is shared by all the neurons belonging to the same convolutional feature map population and contains the kernel weights which are stored in the local data tightly coupled memory (DTCM) of the corresponding population. This solution avoids the reading of the kernel weights from the external RAM each time an event arrives to the convolution module. Each time an event arrives to a convolution module, depending on the source population of the incoming event, the corresponding kernel is read from DTCM memory and the neuron states of the neighbour pixels are updated correspondingly. If any of the updated neurons overcomes the firing threshold an output event is generated and sent to the next processing layer without waiting for any timer, as is done in the conventional implementation of neural systems on Spinnaker. That way, the implemented ConvNet is truly event-driven.

Another characteristic of the ConvNets is that most of the neuron parameters (such as neuron voltage thresholds, voltage reset level, leakage rate, and refractory time) are shared by all the neurons in the same population. Only the particular neuron state and firing times are individual for each neuron. In the original PACMAN 103 version, all the

neuron parameters are replicated and stored individually for each neuron in the SpiNNaker DTCM. Thus the DTCM capacity sometimes limits the number of neurons that can be implemented per core. The PACMAN 103 version was modified to distinguish between the parameters that are individual for each neuron and the parameters shared by the whole population. With this approximation, we are able to implement 2048 convolution neurons per core, where this number is determined by the maximum number of addressable neurons by the implemented routing scheme.

Figure 25(b) shows schematically how the modified PACMAN implements a convolution module spread over multiple cores in a SpiNNaker system. In a ConvNet, a source population of size N_{xs}xN_{ys} is projected onto one or more destination Feature Maps. Each event (x_s,y_s) produced in the source population is projected onto each destination Feature Map by projecting the corresponding kernel (shown as a red box in Figure 25(b)) around the destination coordinate. All neurons inside the red box will update their state by adding the corresponding kernel weight. In order to parallelize this process in SpiNNaker, each destination Feature Map (of size N_xxN_y) can be split into segments (of size hxN_x). Each core will hold, in its DTCM, data common to the full Feature Maps (such as threshold V_{th}, refractory time T_R, leakage time T_L, resting potential V_{rest}, size of full Feature Map N_xxN_y, size of source population N_{xs}xN_{ys}, and projection kernel $\boldsymbol{K}_j$), and specific data for each neuron (such as state, time of last output spike, time of last input spike, and state flags). The parallelization degree is determined by parameter '*h*' in Figure 25(b), and results in a design compromise: the smaller *h* is, the more cores operate in parallel to implement the same Destination Feature Map, and the more events per unit time can be processed, but this consumes more cores (SpiNNaker resources) and a smaller ConvNet can be mapped onto a given number of SpiNNaker chips; on the other hand, by setting *h* at the maximum possible number such that a core can host a maximum number of neurons, the most compact mapping with minimum cores is achieved, at the cost of limiting event throughput. Consequently, parameter '*h*' should be set keeping in mind the compromise between SpiNNaker resources (cores) consumption and throughput (speed).

4.4.1.1.1 Experimental set-up and results

The event-driven Convolutional Neural Network architecture with optimized parameters to perform recognition of the card has been programmed on a 4-chip Spinnaker board. To test the recognition rate we have used a test sequence of 40 32x32 tracked symbols obtained from the events recorded with a DVS. In an attempt to achieve real time recognition while at the same time reproducibility of the recordings, we loaded the events sequence in a data player board. The data player board stores the addresses and timestamps of the recorded events in a local memory and reproduces the events through a parallel AER link in real time (see Figure 26). A splitter board reproduces the event-flow in two separate AER links. One of them enters a commercial FPGA prototyping Raggestone Board. The Raggestone board is programmed to convert the parallel AER events coming into its input connector to the Spinnaker input accepting events coded in a 7-to-10 bit protocol. In a similar way, the board converts output events generated by the SpiNNaker hardware to the parallel AER protocol and sends them through a separate AER output link. The output of the SpiNNaker board coming through the Raggestone board is combined with the replica of the input stimuli (generated by the splitter board) into one single AER event flow using a merger board. That way, we have the input stimuli synchronized with the SpiNNaker classifier outputs. As can be observed in Figure 26, events coming out of the merger board are fed into a jAER board that timestamps the arriving events and sends them to a computer through a USB-2.0 connection.

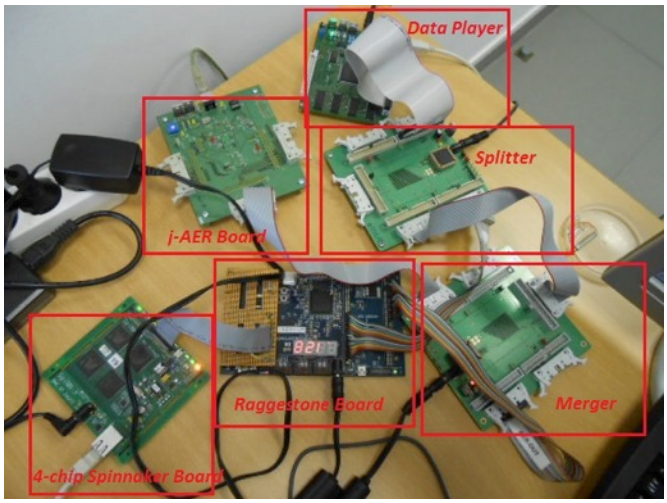

Figure 26: Experimental set-up.

We first tested the correct functionality of the ConvNet for card symbols classification programmed on the SpiNNaker board at low speed. This is done by introducing a **slow-down** factor at the recorded stimulus events timestamps. For this experiment, we multiplied by a factor of 100 the timestamps of all the events of the sequence that was reproduced by the data player board. In order to maintain the same classification capability as the ConvNet architecture optimized for card symbols recognition, we had to multiply all the neuron timing parameters (refractory time and leakage time) by the same factor of 100.

The classification is considered successful if the number of output events of the correct category is the maximum. The mean of the success classification rate was 97% for the 30 repetitions of the experiment, with a maximum 100% and a minimum of 93% in the success classification rate. Once we had checked that the SpiNNaker ConvNet classifier functionality was correct (with a conservative slow-down factor of 100), we tested its maximum operation speed. For that purpose, we repeated the experiment for different, less conservative slow-down factors. We repeated the classification of each test sequence 30 times, measuring the classification success rate as explained above. We tested the following slow-down factors: [1, 2, 5, 15, 20, 25, 30, 50, 100, 200]. A '1' slow-down factor means real time operation. For slow-down factors higher than 25, the mean successful classification rate was higher than 90%. However, for slow-down factors lower than 25, the recognition rate suffered from a progressive degradation (down to about 20% for real time). The main reason behind this loss of performance was the loss of events due to communication saturation.

*Note: Extended documentation of this section can be found in the paper "*Serrano-Gotarredona, T.; Linares-Barranco, B.; Galluppi, F.; Plana, L.; Furber, S., "*ConvNets experiments on SpiNNaker*," in Circuits and Systems (ISCAS), 2015 IEEE International Symposium on Circuits and Sytems, vol., no., pp.2405-2408, 24-27 May 2015".

4.4.1.1.2 Feedback Exploration

The symbol recognition spiking ConvNet topology used throughout Task T11.3.3 is a purely feed-forward topology, like any ConvNet in general (spiking or non-spiking). In T11.3.3 the central objective (besides developing all corresponding ConvNet infrastructure and sensor interfacing) was to explore and exploit the impact of introducing feedback in this type of topology. Two types of feedback mechanisms were explored:

a) Lateral feedback within a Feature Map

b) Returning feedback from a Feature Map to its prior Population(s).

The starting feedforward spiking ConvNet, although capable of very high speed in simulations, suffered from an explosion of event number in the 2nd and 3rd layer. This hampered the speed in hardware realizations. By introducing lateral inhibitory feedback in a Feature Map, the total output event rate in that Feature Map was reduced, while at the same time the contrast sensitivity of the produced outputs was increased. However, this was only possible by suppressing all negative output events, and having the network only produce positive output events. Consequently, this implies the need to modify the starting signed neuron model and corresponding benchmarking ConvNet topology used throughout Task T11.3.3. This is beyond the scope of T11.3.3, although this goal is being pursued as a continuation of this work in a separate project.

We also investigated the impact of projecting back events from a destination Feature Map to a prior Feature Map by using the same kernel weights as a backward projection, although attenuated. This would implement a kind of attentional mechanism by enhancing the activity in a prior layer that gave rise to the activity in the destination layer. We tested this approach by progressively increasing the retroprojecting weights, starting from zero. However, similarly to the case of lateral inhibitory feedback, this retroprojecting feedback only shows benefits (reduction of event rate and increase of signal contrast) if the negative output spikes are suppressed. Consequently, the starting ConvNet topology using signed neurons needs to be modified to use only unsigned output events. This requires the network to be redesigned and retrained under the unsigned neuron model. This again is beyond the scope of T11.3.3, although it is presently being pursued in a separate project.

4.4.1.2 Completely event-driven neuron implementation

In this approach, we modified the standard partition and configuration manager (PACMAN) release from the University of Manchester to make a spike-driven PACMAN for the SpiNNaker platform.

In the original PACMAN, there is a millisecond time-step so that each neuron is updated every millisecond. There are two major disadvantages for this millisecond time-step. First, temporal precision will be limited to 1ms. Second, even without any incoming activity, neurons continuously update every millisecond. If someone wants more accuracy (like very fast visual processing and object recognition) or if someone cares more about power efficiency, a neuron that just becomes updated immediately after receiving a spike might be preferable. We are especially interested to work with micro-second precision because the output of our DVS retina sensor has micro-second precision.

Note: Extended documentation of this section is under preparation for a paper

4.4.1.3 1ms time step implementation using the standard SpiNNaker software

This implementation was done using the latest available SpiNNaker software version (20015.005 – Arbitrary). A new neuron model compatible with the standard models was created to implement the event-driven neuron model.

The new neuron model is inspired in the IF_curr_exp model with some particularities. Our Event-driven neuron fires positive and negative spikes.

4.4.1.3.1 Experimental set-up and results

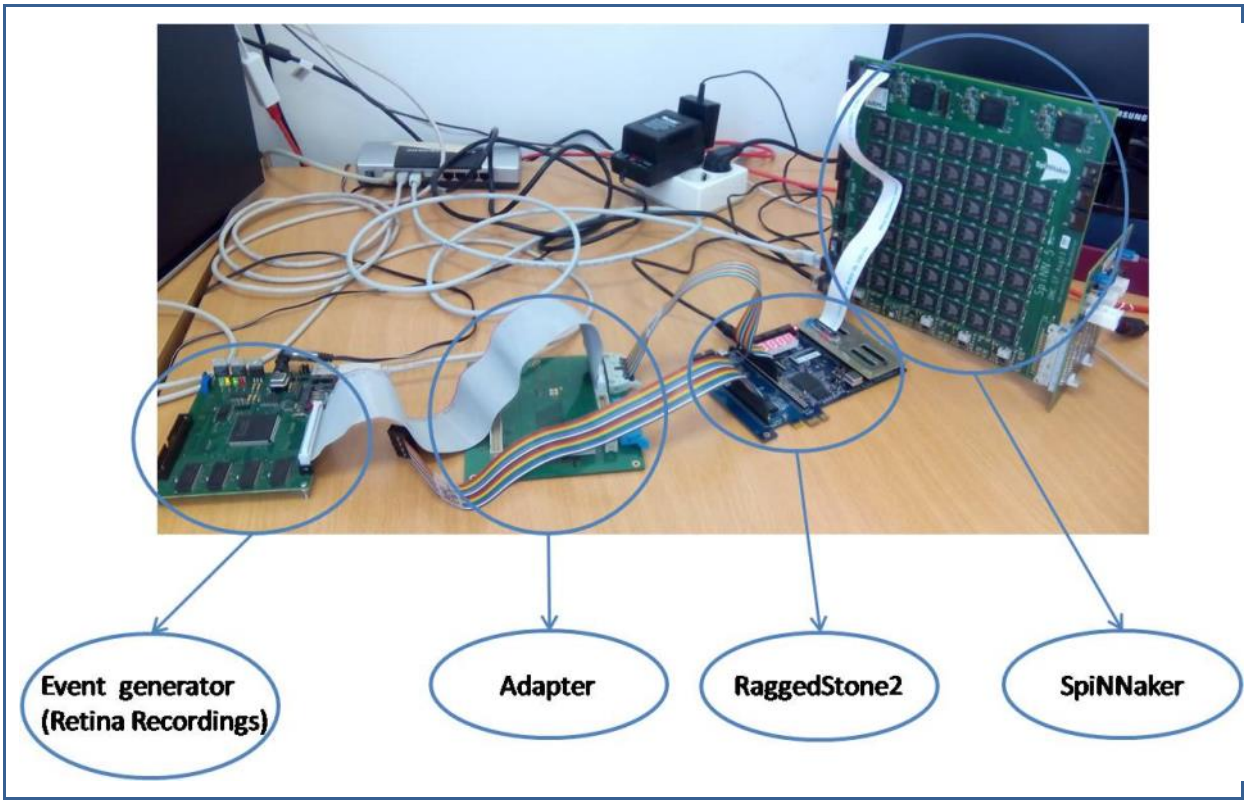

Figure 27: Experimental set-up of the system.

Figure 28 shows the recognition performance at different slow rates. It can be seen that with slow-rate 10 it already has an optimum recognition performance, almost 100%. With lower slow rates (1 and 2) the recognition performance decreases abruptly.

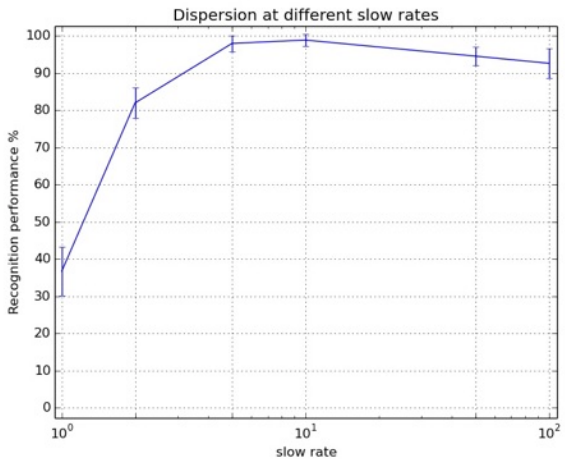

Figure 28: Recognition capability in different slow rates.

4.4.2 Implementation of benchmark architecture on the NCP

4.4.2.1 Interfacing between AER sensors and SpiNNaker (Functions 11.3.3.4 and 11.3.3.5)

The aim of this task is to connect neuromorphic systems to our AER (Asynchronous Event Representation) sensor and pre-processing hardware. This pre-processing hardware is dedicated hardware for communicating through the AER protocol. We developed an

interface for importing and exporting spikes from SpiNNaker chips and visualising them on a computer using jAER software and AERmini2-boards. This task can be divided into the following subtasks:

4.4.2.1.1 SATA communication between AER-Node board and Spinn-FPGA

We developed a firmware that communicates the field-programmable gate array (FPGA) of the AER-NODE board and the FPGAs of the SpiNNaker platform using the serial advanced technology attachment (SATA) communication link. The firmware is capable of sending the incoming spikes from the AER sensor to the SATA link and receiving spikes from the SATA link at 3 Gbps.

4.4.2.1.2 Speed improvement between Spinn-FPGA and SpiNNaker chips

Present day commercial FPGAs operate typically in synchronous mode, thus making it necessary to incorporate synchronisers when interfacing with asynchronous chips. This introduces extra latencies and precludes pipelining, deteriorating transmission speed, especially when sending multi-symbols per unit communication packet. We present a technique that learns to estimate the delay of a symbol transaction, thus allowing a fast pipelining from symbol to symbol. The technique has been tested on links between FPGAs and SpiNNaker chips, achieving the same throughput as fully asynchronous synchroniser-less links between SpiNNaker chips. The links have been tested for periods of over one week without any transaction failure.

Note: Extended documentation of this section can be found in Yousefzadeh et al. (2016)[8].

4.4.2.1.3 Receiving spikes form SpiNNaker and interfacing with AERmini2 board to visualize in jAER

As mentioned before, the FPGA of the AER-Node board receives the output spikes from the SpiNNaker platform. The AER-Node board merges the incoming spikes from the AER sensor with the incoming spikes from the SpiNNaker platform, converts the packets into pixels and sends them to the AERmini2 board.

Displaying both sources helps to verify the correctness of the neural network implementation in SpiNNaker platform and gives us an idea of the system delay.

4.4.2.2 Improvement in the optimisation process (Function 11.3.3.2)

The aim of this task is to perform exhaustive parameter optimizations by exploiting the real-time and accelerated-time capabilities of the two hardware platforms.

Our approach to change the neuron parameters during the simulation time is to send a message from a host computer to the neurons telling the new value for the parameter to be changed. To do this, we use a “config neuron” interfacing with the host computer using the UDP protocol and connected to a subset of neurons.

The host computer sends an MC packet with PAYLOAD using the KEY of the “config neuron” and the message arrives to all the neurons connected to the “config neuron” with that KEY. When a neuron receives a message with PAYLOAD, it decodes the parameter to change and the value to assign and changes the value of its parameter. The PAYLOAD can be a 32-bit field; therefore, if four parameters are going to be changed, we need 2 bits for the parameter codification and 30 bits for the value of the parameter.

Figure 29 displays a summary of the time and number of iteration results extracted from running the optimization process at different slow-rates. The number of iterations varies depending on the initial values used for the optimisation algorithm. The initial values were generated randomly between specific limits. From the obtained values with the proposed approach, the time that the standard SpiNNaker software will need to do it is calculated (last column of Figure 29). We can see a significant time reduction in the optimization process with the proposed approach. The time reduction is more noticeable when the relation between the loading time and the simulation time is bigger. As it was mentioned

before, using more complex neural networks increases the loading time, and therefore the optimisation time with the standard software will be extremely increased. However, the increase in the loading time is insignificant for the proposed approach.

Slow_rate	Number of iterations	Time with the proposed approach in s (in h)	Time with standard software in s (in h)
1	2160	32749 s (9 h)	334800 s (93 h)
2	1584	23852 s (6 h)	245520 s (68 h)
5	1130	23008 s (6 h)	180800 s (50 h)
10	138	3689 s (1 h)	22770 s (6 h)
50	1500	128068 s (35 h)	330000 s (91)
100	1440	203678 s (56h)	381600 s (106)

Figure 29: Optimization process time and number of iteration results.

Note: Extended documentation of this section is under preparation for a paper

4.4.2.3 Results of the mismatching effect (F11.3.3.7, F11.3.3.8, F11.3.3.9)

The aim of this task is to artificially add controlled variability into the SpiNNaker platform in order to analyse natural mismatch impact of other platforms implemented using analog circuit techniques, like the HBP PM-NM. This will allow us to study the impact of variability and analyse the degradation of the system.

Figure 27 shows the hardware setup used for the measurements. The event generator reproduces the recording of the symbols sequence. This event generator is connected through an adapter to the RaggedStone2 board that maps the AER events to SpiNNaker packets and sends them to the Spinn-5 board using the parallel port. The Spinn-5 board is connected to the network, so that it can interface with a host computer using Ethernet for gathering the results or modifying the neuron parameters.

The SpiNNaker platform, as a digital platform, stores the values of the kernels and neuron parameters in a binary representation. To analyze the effect of the mismatch in a hardware network, a random variation is applied to kernels and neuron parameter values in the SpiNNaker platform. A relative coefficient (RC), from 0 to 100, was applied to each kernel and parameter value as a standard deviation to obtain the random variation using a Normal distribution.

The following results show the mean (with an '*') and the standard deviation (with an error bar) for each parameter and kernel value for 5 measurements using different RC values.

All the experiences are done with a slow-down factor of 50. This factor showed a good recognition performance in the optimization process.

4.4.2.3.1 Parameter mismatch

The mismatch of three neuron parameters is analysed: v_threshold, refractory time and leakage rate. These parameters have been identified as the most significant parameters for the recognition of the poker card symbols.

- *V_threshold*

Figure 30 shows the effect of increasing the RC in the poker card recognition performance. "All layers" line represents the result of applying random variations with the specific RC in

all layers. It can be seen that with an RC of 20% the mean of recognition falls under 80% and with an RC of 30% the recognition mean is less than 50%. “Layer 1”, “Layer 2”, “Layer 3” and “Layer 4” lines represent the result of applying random variations only to layer 1, 2, 3 and 4 respectively.

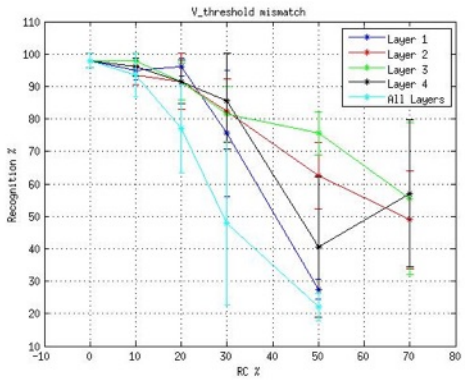

Figure 30: V_threshold parameter mismatch by layers and compared to all layers.

- *Refractory time*

Figure 30 shows the effect of the refractory time mismatch. It can be seen that for RC higher than 20% it starts to decrease the recognition percentage. SpiNNaker software has an inherent 1 ms time step, therefore depending on when the event happens, it can have an inherent maximum refractory time of 1 ms. This is a reason why for low RC values there are small changes in the recognition percentage.

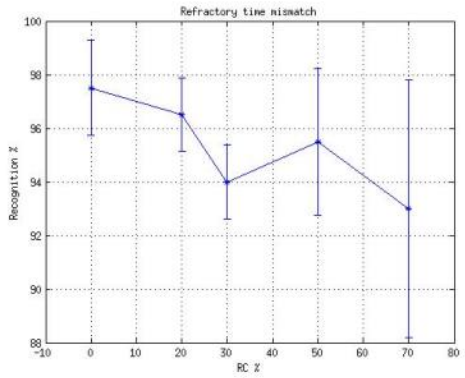

Figure 31: Effect of refractory time mismatch.

- *Tau_m (Membrane Time Constant)*

With the increase of RC the recognition percentage decreases. However, with the largest RC value (70%) the recognition is higher than 90%. Therefore, this parameter mismatch does not have much impact on the recognition performance.

4.4.2.3.2 Kernel values mismatch

This section describes the results of applying random variations on the kernel values. The measurements are done layer by layer, applying the variations only to one layer, and leaving the others without the variation, and for all the layers, applying the variations to all the layers at the same time.

Figure 32 shows the effect of increasing the RC when generating the random variations. It can be seen that all the layers behave in the same way with the increase of RC. Up to an

RC value of 10%, the mean values of the recognition percentage are over 90%. This means that the kernels are robust to mismatch.

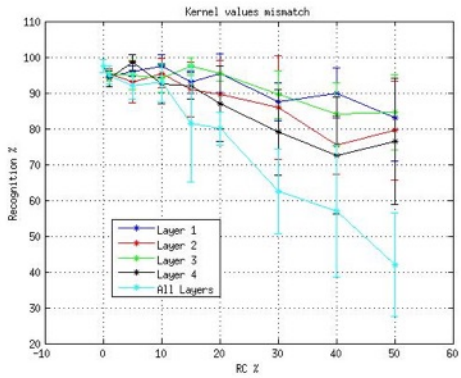

Figure 32: Effect of kernel mismatch with the increase of RC.

With these results, we can say that the most critical parameter is v_threshold. The system tolerates an RC of 10% in the kernel values without a decrease in the recognition performance. Therefore, the poker card convolutional neural network (CNN) shows a robust behavior to parameter and kernel value variability. Furthermore, with this study we have proved the capability of SpiNNaker to perform mismatch analysis.

4.4.3 Description of major use case(s) and target users of your application

Our experiments with SpiNNaker relate to vision based applications using special event-driven cameras, also called DVS. In principle, our experiments have focused on single-SpiNNaker-PCB cases, which easily allows for scenarios with portable robotic systems. However, our ConvNets based approach extends immediately to Deep-Neural-Networks, as ConvNets are a particular case of the latter. Therefore, we can expect to scale up our poker-card ConvNet prototyping systems to highly scaled-up versions which require the extensive use of both HBP remote NCP platforms. Note that biological visual systems use a large percentage of the neurons in the brain. Visual processing is extremely neuron-hungry. Consequently, we foresee that our studies can be used as a starting point for for elaborate neuro-scientific studies of visual systems, as well as real-world artificial applications on mobile platforms.

4.4.4 Documentation of models and related experiment set-up

As mentioned in Section 4.3.1 a new event-driven model is implemented in SpiNNaker using three different approaches. and Figure 25 shows a possible block diagram and sate evolution of the event-driven model.

Sections 4.4.1.1, 4.4.1.2 and 4.4.1.3 describe respectively the three implementations of the event-driven model using SpiNNaker. Each implementation has advantages and disadvantages. Depending on the aim of the experiment, e.g. recognition performance, latency, software version compatibility and so on, the implementations show different results.

Using this event-driven model, the poker card convolutional neural network is implemented to analyse its performance. Figure 33: shows the network design used for the poker card recognition CNN.

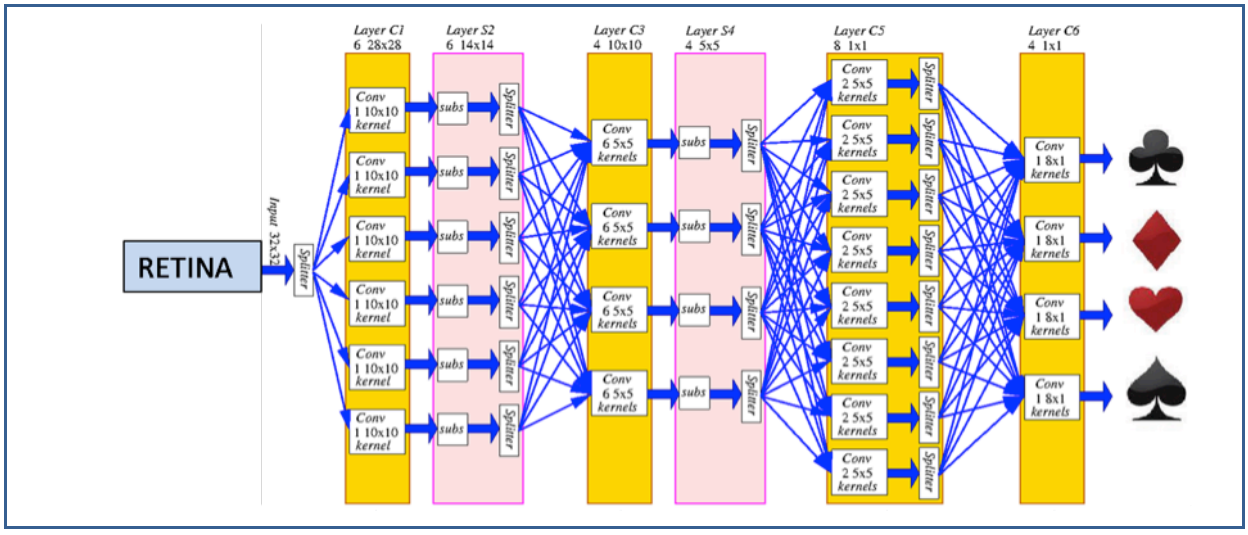

Figure 33: Poker card convolutional neural network.

The experimental set-up used for each experiment is shown in Figures 26 and 27. All the experimental set-ups are equivalent, in some cases we use our own dedicated hardware AER-NODE, and in others we use the Raggestone board. For the first case the connection to the SpiNN-5 board is through the SATA link and for the second case the connection is through the parallel port.

By the end of the Project the source files of all the implementations will be delivered along with the dataset and the instructions to configure and run the experiments.

4.4.5 Outreach

During the development of the work described, the CSIC group has had extensive research interaction and exchange of visitors between the groups of Prof. S. FURBER at Manchester Univ. and of Prof. R. BENOSMAN at UPMC, Paris, France, both part of the HBP consortium. The work developed has contributed to the outreach of participating in two new H2020 projects and two new industrial contracts, one with Samsung and one with a new UPMC spin-off Chronocam.

Papers, Conference Presentations or Posters

- Yousefzadeh, A.R., Plana, L.A., Temple S., Serrano-Gotarredona, T., Furber, S.B., and Linares-Barranco, B. Fast Predictive Handshaking in Synchronous FPGAs for Fully Asynchronous Multi-Symbol Chip Links. Application to SpiNNaker 2-of-7 Links," IEEE Transactions on Circuits and Systems II: Express Briefs, 2016. **PP:**1-1.
- Serrano-Gotarredona, T., and Linares-Barranco, B. "Poker-DVS and MNIST-DVS. Their History, How They were Made, and Other Details," *Frontiers in Neuromorphic Engineering*. Frontiers in Neuroscience. 2015. **9:**481. doi: 10.3389/fnins.2015.00481.
- Serrano-Gotarredona, T., Linares-Barranco, B., Galluppi, F., Plana, L., and Furber, S., "ConvNets experiments on SpiNNaker," in Circuits and Systems (ISCAS), 2015 IEEE International Symposium on Circuits and Systems, 2015. pp.2405-2408, 24-27 May 2015.

4.5 T11.3.4 (UNIBI): Spiking Associative Networks for Neuromorphic Computing Systems

Table 7: Status of spiking associative network models for neuromorphic computing systems

Model	Implementation status	Benchmarks evaluated	Comment
SAM-PyNN	SAM simulations PyNN/Nest	16	Done
SAM-ESS	SAM emulations with ESS	4	Done
SAM-Spy	Runnig on Spikey NM-hardware	4	Done
SAM-PM	Running on NM-PM-1	2	Ongoing
SAM-SNN	Running on SpiNNaker	4	Done
SAM-MC	Running on NM-MC-1	2	Ongoing

4.5.1 Implementation of benchmark architecture on the NCP

Within the Project we successfully implemented a generic software prototype of an SAM in PyNN based on leaky IaF neurons. The SAM has a single layer structure with feedback and the synapses of the IaF neurons are binary or analogue. This prototype is now used as a benchmark network for the neuromorphic hardware platforms and to experience the effects of parameter variations on the behaviour of SAMs. We specified 16 different versions of SAMs with PyNN and simulated them with NEST. The simulation results act as the desired SAM behaviour and will be compared later on with the results from the hardware implementations.

As a next step we simulated SAMs (e.g. 224 neurons with 224 synapses each and 1024 neurons with 1024 synapses each) with the Heidelberg hardware emulator (ESS: Executable System Specification). With the help from the Heidelberg and Dresden groups we improved the ESS simulations so that they are now closer to the expected results of the real hardware. The implementation on the virtual hardware is the first and necessary step to implement SAMs on the Heidelberg HBP hardware NM-PM-1 (Function 11.3.4.4). As a second step we mapped the simulated SAMs (PyNN/Nets, ESS) on all SP9 hardware platforms (Spikey, NM-PM-1, SpiNNaker, NM-MC-1) and studied the effects on SAM behaviour quantified by the selected performance measures storage capacity, retrieval time, energy consumption, and robustness (Functions 11.3.4.5 and 11.3.4.6).

We started with the simulation of connected neuron populations (cell assemblies) on NM-MC-1. As a first approach we adapted the spike-counter model from Knoblauch ("Synchronization and pattern separation in spiking associative memories and visual cortical areas" (2003), PhD-Thesis). Assisted by the Manchester group, we start to run larger numbers of populations on NM-MC-1.

We achieved our first milestone "SAM implementation on virtual hardware (ESS)" on schedule (M19). The defined KPIs have been fulfilled as well: 16 SAMs were implemented in PyNN, 2 benchmark parameter sets (one for each neuromorphic platform) generated, and 8 SAMs successfully tested on HBP neuromorphic platforms. The final milestone "SAM implementation on HBP hardware" is almost done and will be reached in time (M30).

4.5.2 Description and usage of implemented SAM models to compare different execution environments (benchmark development)

All 16 different versions of the generic SAM model were implemented in PyNN and simulated with NEST first (Function 11.3.4.1). For systematic testing we used two automatically generated test pattern sets (> 1 million sparsely coded patterns), one for NM-PM-1 and one for NM-MC-1. We developed a software tool for automatically analysing simulation outputs (spike trains) and for comprehensive reporting of the performance evaluation (Function 11.3.4.2). For a more detailed analysis of the parameter influences

on SAM behaviour we implemented a specified program package for an automatic generation of SAM benchmark data sets (Function 11.3.4.3). With these tools we can compare conveniently the different execution environments for the neuromorphic platforms as well as their performance.

4.5.3 Description of major use case(s) and target users of your application/tools

The developed tools (SAM generation, benchmark generation, PyNNLess, performance analysis, parameter optimization) are open to all HBP users. For the SP9 researchers these tools simplify the hardware access and systematic hardware testing. These tools are especially helpful for the design of the next generation of neuromorphic hardware within HBP.

Also, performance and memory bottlenecks in the software stacks of the current HBP target platforms have been identified that hinder scalability of the SAM benchmarks. In this way, the developed benchmarks and tools have already provided valuable feedback to the developers of software stacks and are used as test beds for proposed and prototyped countermeasures. Finally, the benchmarks and tools target the designers of software abstractions for the modelling of spiking neuronal networks, because these abstractions need to scale to future larger neuromorphic hardware platforms and meaningful evaluation requires dependable reference test cases.

4.5.4 Documentation of models (and tools) with related experiment set-up

Based on our generic implementation of spiking associative networks in PyNN we are now able to simulate different architectures with currently up to 10,000 neurons in order to analyse the effect of parameter variations on associative memory system performance. In order to simplify systematic simulation runs on all platforms we developed the Software Abstraction Layer PyNNLess as a unified simulation backend. PyNNLess supports the easy development of SAM experiments that transparently work on HICANN physical model (NM-PM-1), its ESS emulation, SpiNNaker (NM-MC-1), and the NEST software simulator (version 2.2 or 2.4). PyNNLess forms a software layer above PyNN. This layer hides the differences between the PyNN versions 0.7 and 0.8 (as required by the different HBP target Platforms) and handles current limitations in hardware backend binding. A tried and tested stable version of PyNNLess has been released (GitHub repository hbp-sanncs/pynnless). This version has already been used to run test cases and the several representative neural associative memory (NAM) experiments remotely on hardware backend.

The accompanying AdExpSim tool (GitHub repository hbp-sanncs/adexpsim) was implemented for parameter optimisation and for faster design space exploration. The first phase of the NAM parameter space exploration has to cope with objective functions that are step functions, like the number of desired neuron spikes. In this phase, promising parameter constellations have to be selected quickly based on a simplified idealised model of a small slice of the full NAM, as are starting points for the second phase. A reformulation of this piecewise constant functions as continuous functions has been developed. Such continuous functions avoid that the optimisation algorithm gets caught in the plateaus of the original step function. The continuous reformulation of the step function is based on the lowest interference voltage required at the most critical point in time to trigger an additional spike. It is thus a continuous completion of a discrete objective that uses robustness against interference to grade parameter constellations with the same number of neuron spikes. Now we can automatically compute an adequate parameter set for SAM implementations on Spikey and HICANN.

4.5.5 Outreach

Bielefeld University (UNIBI) participated in all video conferences of the Neuromorphic Computing Division (SP9/WP11.3) within the reporting period. We attended the HBP Summit in Madrid (September 2015) and contributed to the presentations of SP9/SP11. Together with the groups from KTH Stockholm and UHEI Heidelberg we contribute to the definition of benchmark sets for spiking associative memories. Project results were presented at the second HBP Education Summer School (Obergurgl) and the 3rd HBP Education Workshop (Manchester).

4.6 T11.3.5 (UMPC): Asynchronous Computational Retina

Table 8: Internal milestones for T11.3.5

Internal Milestone	Name
11.3.5.1	Interface to connect one ATIS camera into SpiNNaker
11.3.5.2	Interface to connect two ATIS cameras into SpiNNaker
11.3.5.3	Hardware implementation: Stimulation platform
11.3.5.4	Hardware implementation: Database platform
11.3.5.5	Computational model: Visual motion
11.3.5.6	Computational model: Retina model
11.3.5.7	Computational model: Stereovision

This task developed a pure, event-driven visual computation approach that uses precise timing mechanisms to design new computation techniques in visual processing. The task produced a full event-driven visual processing system linking a neuromorphic retina directly to the SpiNNaker system by an Asynchronous Event Representation (AER) bus. The architecture allowed the first real-time development and implementation of new, visual, event-driven computation techniques. We implemented event-driven early vision models and 3D stereovision in the SpiNNaker board using a precise timing mechanism.

The neuromorphic retina (Asynchronous Time-based Image Sensor, ATIS) used in this work is an event-based time-domain, developed by members of the team, which encodes the visual scene with 304x240 pixel resolution. It contains an array of fully autonomous pixels that combines a luminance change detector circuit and a conditional exposure measurement block. These sensors are novel vision devices that, like their biological counterparts, are driven by “events” occurring within the scene, as opposed to conventional image sensors that are driven by artificial timing and control signals (e.g., a frame clock) with no relation to the source of the visual information. The ATIS output fits the SpiNNaker massively parallel multi-core architecture. It benefits from SpiNNaker’s ability to simulate millions of neurons, and makes use of its computing power to develop and map event-driven computation architecture.

4.6.1 Hardware implementation: Interface to connect one ATIS camera into SpiNNaker (Milestone 11.5.3.1)

The ATIS camera used in this work is composed of two electronic boards. The first, denoted the *sensor board* contains the ATIS chip itself with all its required components.

The second is an Opal Kelly XEM3010 FPGA board housing a XC3S1500-4FG320 Spartan-3 FPGA from Xilinx.

We designed an interface board, which can be plugged in between these two existing PCBs (see Fig. 33) in order to add the driving electronics and connectors required to interface the ATIS camera to the SpiNNaker system. This interface board contains the connector required for the SpiNNaker link[9,10] and the linking logic. A view of this PCB is presented Fig. 31, showing the different components sitting on the board.

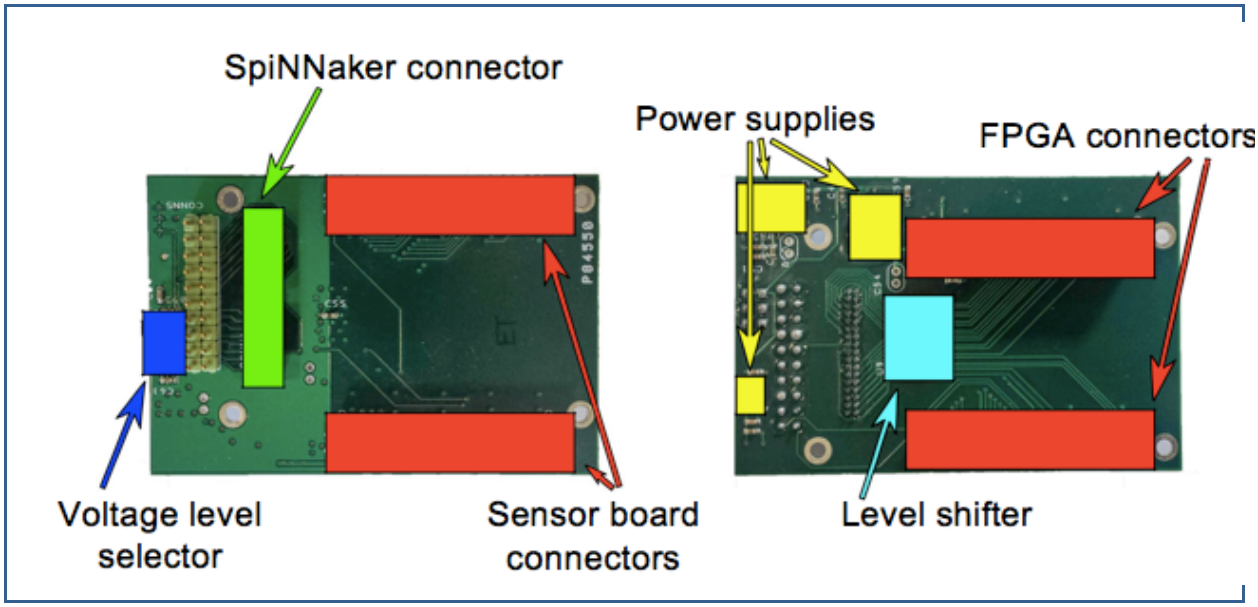

Figure 34: Interface board used to feed data from an ATIS camera to SpiNNaker

The interface board sits in between the sensor board and the FPGA board driving the camera.

ATIS events occupy a routing space of 9 + 8 + 1 = 18 bits, as the x coordinate can be mapped in the 0-304 range, the y=coordinate can be mapped between 0-239, and the polarity bit can assume values in the 0, 1 range accordingly to the light intensity change direction.

An original SpiNNaker event on the other side is represented by a 32-bit routing key/address (8+8 for the x,y chip coordinates, 5 bits for the core coordinate and 11 bits for the neuron id within a core). ATIS events need therefore to be projected in the 32-bit routing space of SpiNNaker. This is done using the techniques described in this report which derive from Plana (2013).[10]

Both solutions rely on the idea to represent the spiking sensor as a virtual SpiNNaker chip, external to an original physical mesh and to convert the addresses to the same format used by SpiNNaker. By doing so the sensor can seamlessly be integrated in the network simulated on SpiNNaker; the sensor itself, using the board described in Section 1 emits spikes which are no different than the ones produced by other SpiNNaker chips; its interconnection is therefore transparent to the system. In other words, spiking sensors are assigned fictional, virtual chips, which are not physically present on a SpiNNaker board. Sensors directly feed their spiking data into the SpiNNaker interconnect through the bespoke SpiNNaker link or through the SATA interface.

The mapping mechanism and the SpiNNaker packet structure are shown in Figure 35 (after Plana, 2013).[10]

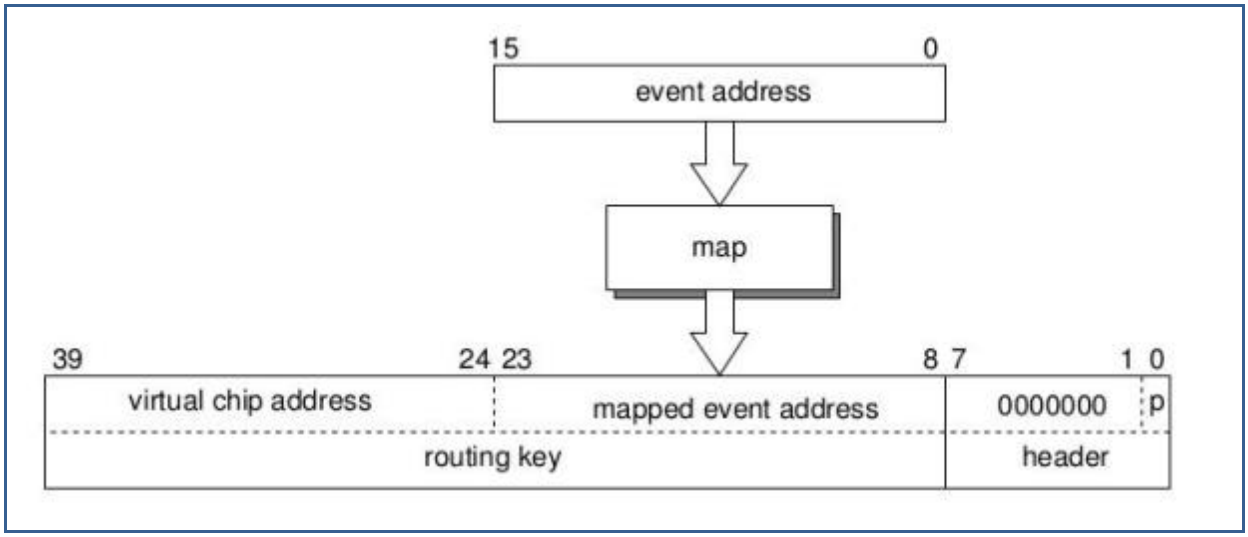

Figure 35: SpiNNaker packet structure and event to MC packet mapping.

Injecting spikes from a single ATIS silicon retina into SpiNNaker follows the same technique presented in Galluppi et al. (2014),[11] they both rely on using the link provided in chip 0,0 for both boards. A region of interest of 128 × 128 pixels is selected from the original ATIS sensor space, and are assigned a virtual chip 254,254 and injected in chip 0,0. During the mapping process extra neural applications, called *Proxy*, are responsible for the translation from the virtual routing key (254,254) to a key which is physically present in the same chip where the *Proxy* application is loaded, generally chip 0,0. After the packet translation the AER packet containing the address of the event can be routed in the SpiNNaker system as any other MC packet.

Figure 36: Designed interface board plugged between an ATIS camera and the SpiNNaker system.

4.6.2 Hardware implementation: Interface to connect two ATIS cameras into SpiNNaker (Milestone 11.3.5.2)

Connecting two ATIS cameras on the 4-chip SpiNNaker board was developed with the same approach: the two available SpiNNaker links on the 4-chip board were used to inject spikes.

Figure 37: Designed interface boards plugged in two ATIS cameras feeding data into the SpiNNaker system.

Moving to the Spin-5 platform (42 chips) required to develop a new interface for the two cameras: this board does not have a dual SpiNNaker link. Work from Yousfadeh et al., (2016)[8] was used in order to achieve this: an AERNode board receives events from both cameras, merges them and sends them to the SpiNNaker board through the SATA bidirectional connection (Xilink High Speed inter FPGA connection AURORA[12]. The AER protocol defined in section 1 was used to inject events through SpiNN5 FPGAs.[10] The F1-L01 link (chip 0, 0, link West from PFGA F1 - Figure 36) was used to inject the spikes in the platform.

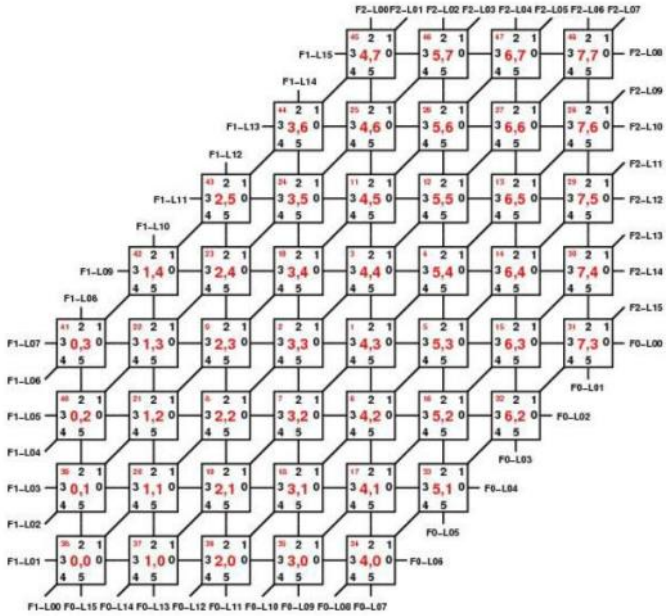

Figure 38: SpinnakerSpiNN5 hardware description and FPGAs links.

Such a bus reaches up to 2Meps (events per second).

As the ethernet outcoming link was limited to 64 keps (one UDP packet per event per millisecond), the same link was used to send the events back to the AERNode Board,[13] and then back to the computer through an AER-USB mini2 board[14].

4.6.3 Hardware implementation : stimulation platform (Milestone 11.5.3.3)

The aim of this platform is to create automatically a database for shape recognition with the ATIS sensor. Due to its high temporal resolution, this platform should be able to move the desired shape at a high speed (typ. 3 $m.s^{-1}$). The shapes are displayed on an 6" electronic ink display from Visionect, [15] which presents the advantage, compared to standard displays, to be continuous and thus match the high temporal resolution of the ATIS sensor, thuis avoids avoiding flickering.

Two high-dynamic axis were used, powered by two NEMA 23 (1.8deg per step, 24-48V nominal @ 100W) stepper motors.

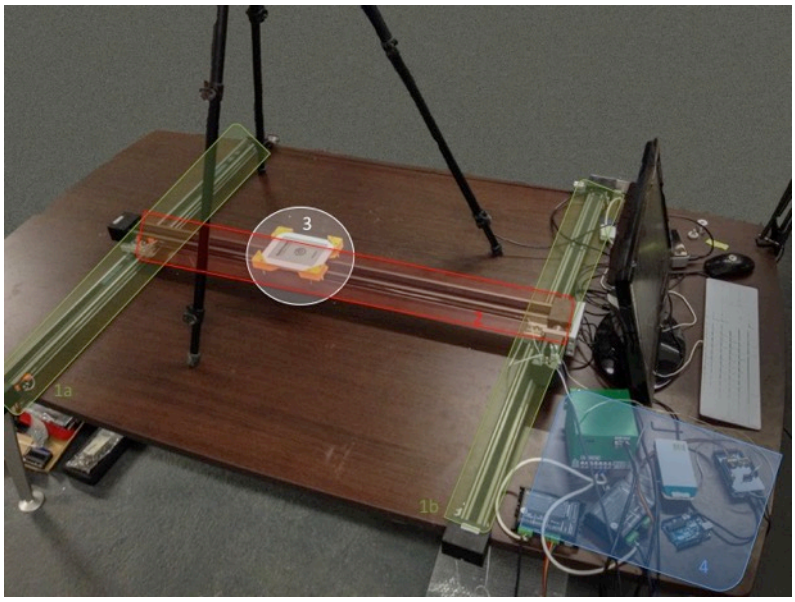

Figure 39: XY platform - 1a) & 1b) Y axis. 2) X axis. 3) Eink display. 4) Electronics.

As the maximal frequency for the stepper drivers was 200kHz, the possible speeds were :

Steps per Revolution	200	400	800	1600	3200	6400	12800
$V_{max}(m.s^{-1})$	62.83	31.42	15.71	7.85	3.93	1.96	0.98
A_{min} to reach $V_{max}(m.s^{-2})$	4934,8	1233,7	308,4	77,1	19,3	4,8	1.2
V_{max} with $A_{max} = 15m.s^{-2}(m.s^{-1}$	3.46	3.46	3.46	3.46	3.46	1.96	0.98

4.6.4 Generate datasets for tests and evaluation of computational models (Milestone 11.3.5.4)

With the stimulation platform described in Section 3, a database was recorded for the testing and evaluation of computational models.

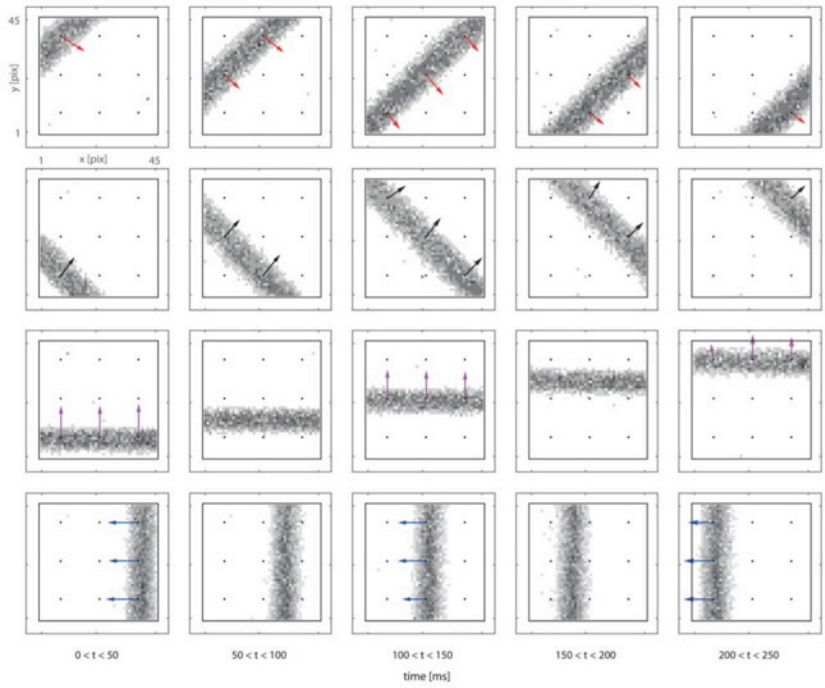

Figure 40: Capture of a moving bar in the database.

4.6.5 Computational model: Visual motion (Milestone 11.3.5.5)

For computing the optic flow we use a slightly modified version of the model presented in Giulioni et al. (2016) [16]. The original model, inspired by the neural circuitry found in the rabbit's retina by Barlow and Levick (1965) [17] is based on inhibition-based direction-sensitive (DS) units, combined in motion detectors. A single DS unit provides information only if the object moves in its preferred direction (see Figure 42), otherwise it stays silent. To extract a 2D time-of-travel vector we combined four DS units together in a single motion detector (see Figure 43, upper panels). The four DS units share the same *start* (orange) neuron while they have four different *stop* neurons mapped onto the retina macropixels such that they are selective to upwards, downwards, leftwards or rightwards movements. Correspondingly, each 2D motion detector has four output *counters*. The number of spikes emitted from a counter neuron is inversely proportional to the detected speed. Inhibitory currents are set so as to immediately shut down the neuron's activity, so as to relay speed information. Our complete model comprises 64 × 64 × 4 direction sensitive units for a total of 64 × 64 motion detectors. Each motion detector receives input from a 2 × 2 macropixel of the original retinal resolution (128 × 128) through a subsampling population which acts as a robust edge detector.

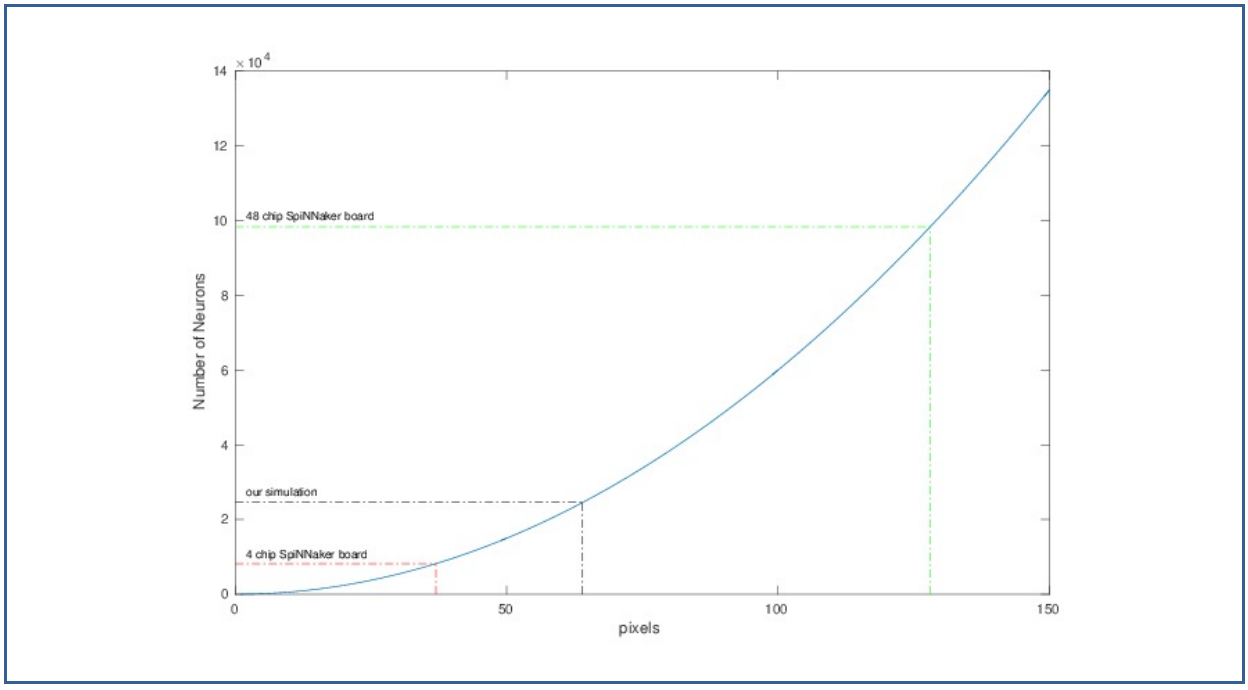

Figure 41: Number of neurons needed for a simulation versus the number of pixel

Two limits are represented : the small 4 chips SpiNN3 board, and the big 48 chips SpiNN5 board.

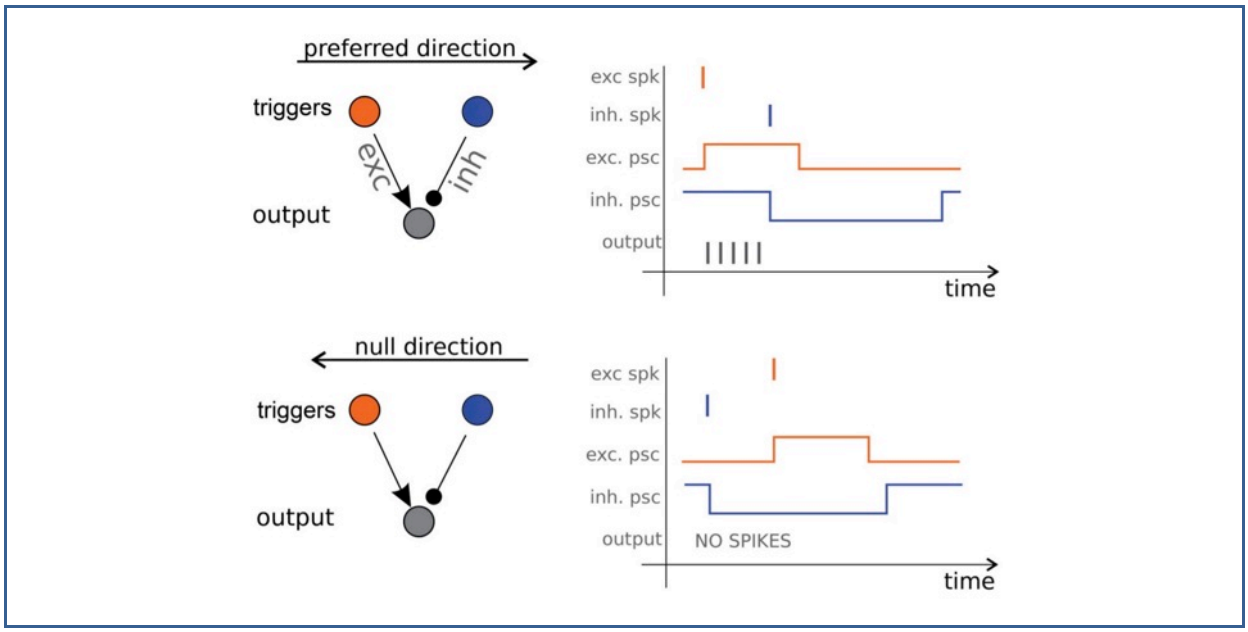

Figure 42: Simplified principle

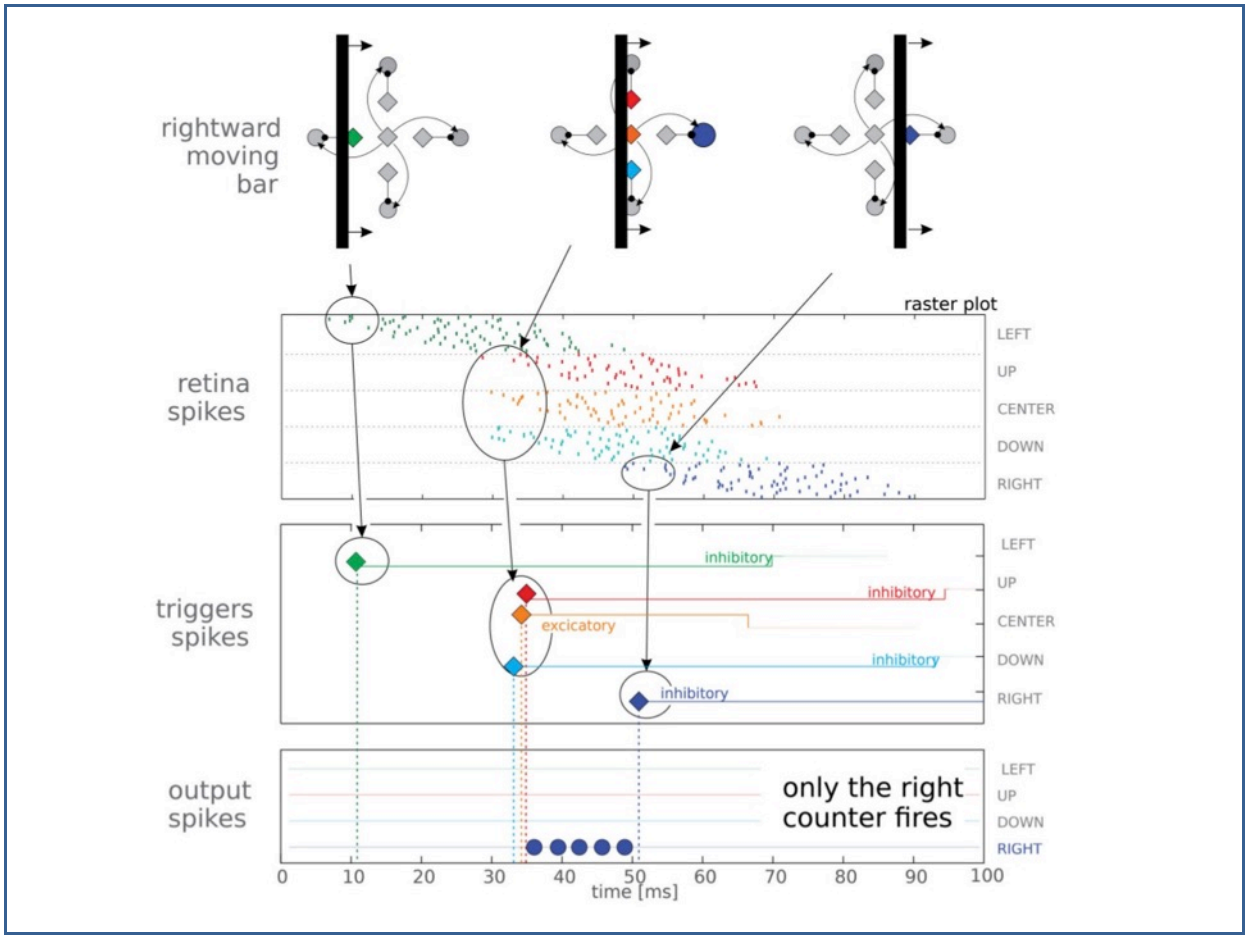

Figure 43: On a moving bar

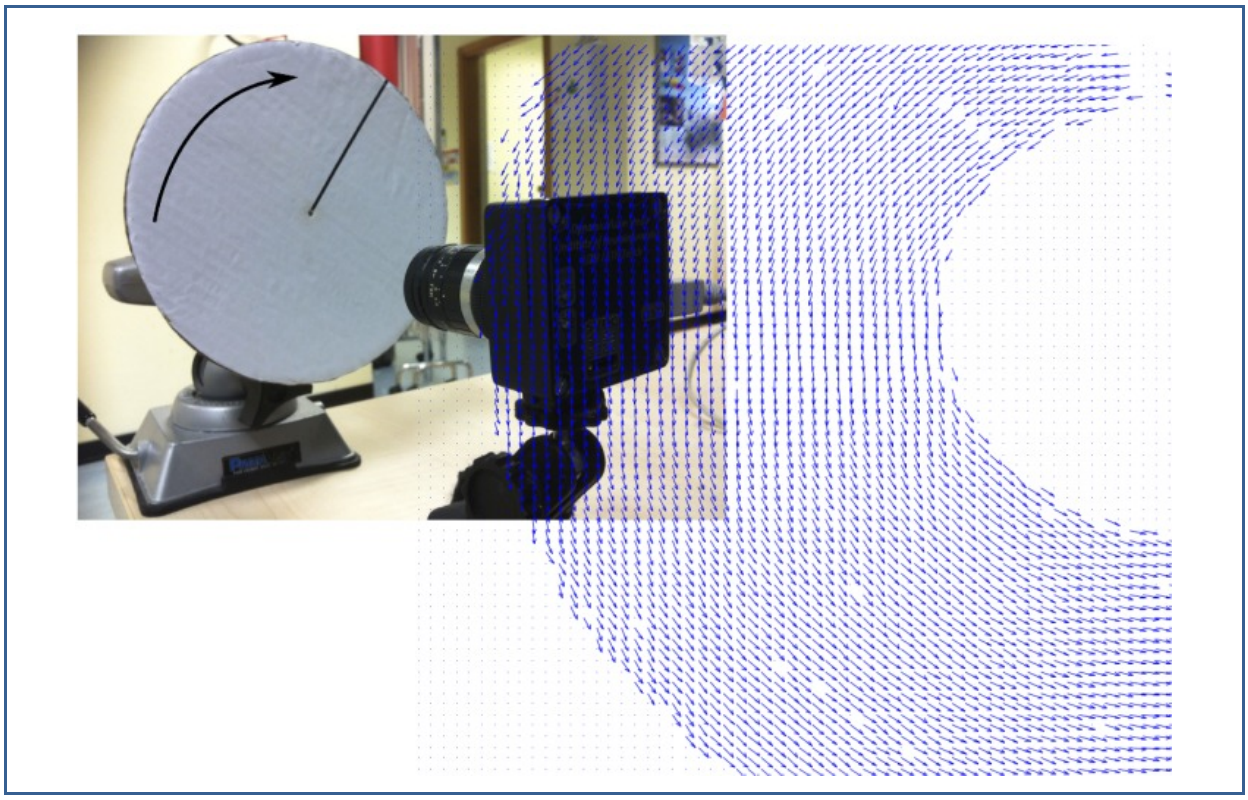

Figure 44: Moving bar and optical flow

Only 32x32 pixels were implemented on the small board. On the SpiNN5 one, we scaled this up to a 64x64 ROI, leading to the use of N cores and M neurons.

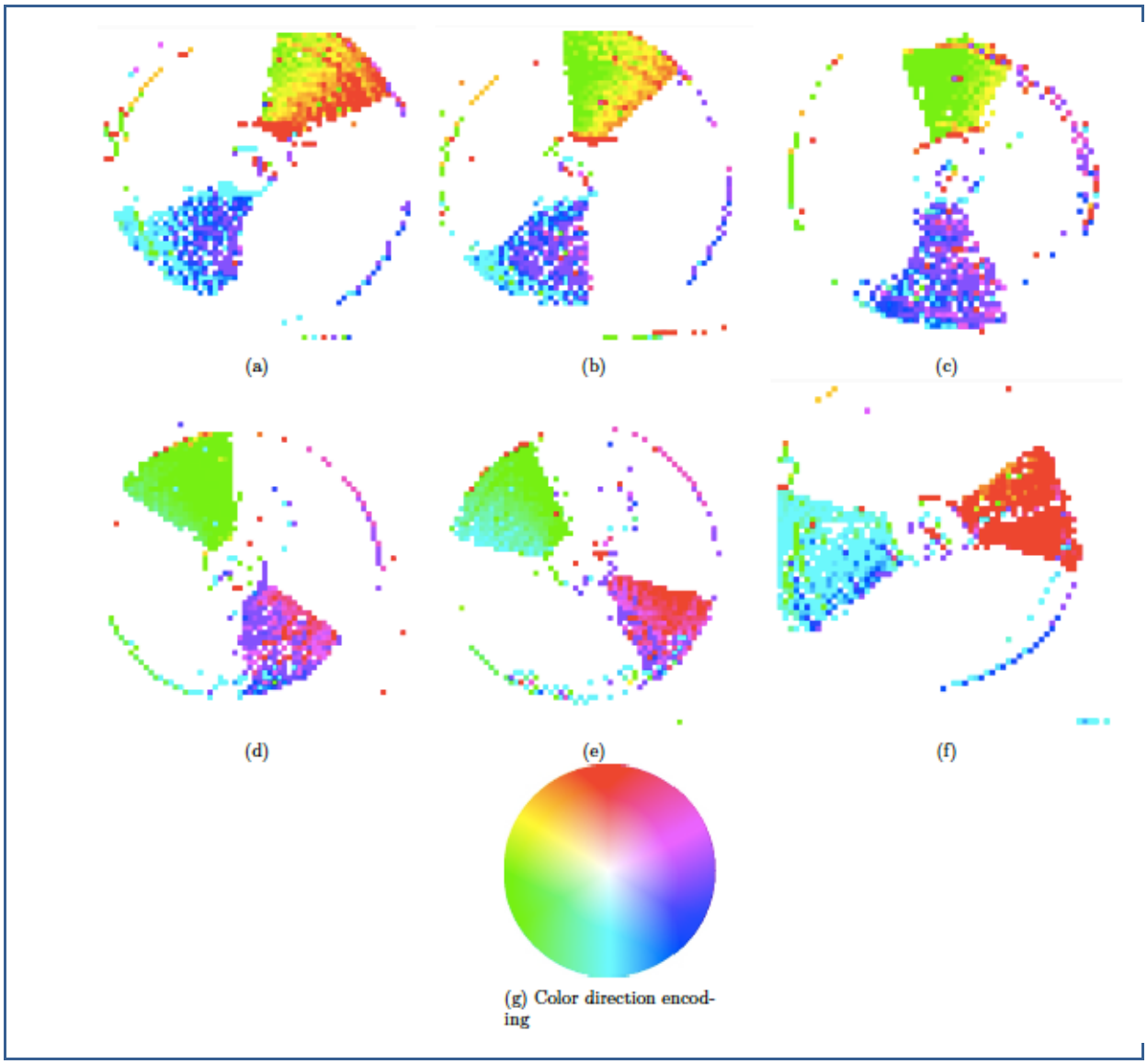

Figure 45: CounterClockwise rotation of a pen, speed and direction are encoded among the HSV map.

Computational model: Retina model (Milestone 11.3.5.6)

In SpiNNaker software available neural models are implemented in a mixed time/event-driven fashion. These models use the Timer event to periodically update the state of the simulated neurons with a given timestep. Parallel to that update process, incoming spikes to the implemented neural population are processed through the Packet Received Event. This event looks up the different synaptic weights and delays relative to each connection. When the synaptic delay has been retrieved, the future contribution of the spike to the membrane potential of a given neuron is stored in its associated Post-Synaptic Potential buffer (PSP buffer). To implement the actual delay, the PSP buffers are ring buffers comprising one cell per simulation timestep. This introduces a big memory footprint when modelling delays bigger than a few timesteps. In order to circumvent this memory limitation we decided to implement these delays independently from the rest of the neural simulation We have therefore proposed a dendritic delay model which uses a proxy neural population to implement larger delays. [18] One dendritic delay core implements one particular delay value. When a packet containing a spike is received, it is stored in a ring

buffer in DTCM (local memory of the core). Then, a second process schedules events to dispatch these spikes after the given delay of the core has elapsed. This allows very compact and efficient code because events are output in their order of arrival.

Because there are a large number of cores available in a typical SpiNNaker machine, using one core per delay value is not troublesome. Moreover, one could configure a network where a spike goes several times through the same delay core to implement multiples of a base delay: if one core implements a delay of 100 ms, it can be used to realize a delay of 300 ms by routing events three times through the core before delivering the spike to its target neuron.

We use these dendritic delays populations in the retinal model so as to cope with the temporal delays used to implement the alpha-functions characterizing the current response of the cells to an event. These delays can range from a few milliseconds to several hundred milliseconds on SpiNNaker – see Table 1 in Lorach et al. (2012)[19] and Lagorce et al. (2015)[18].

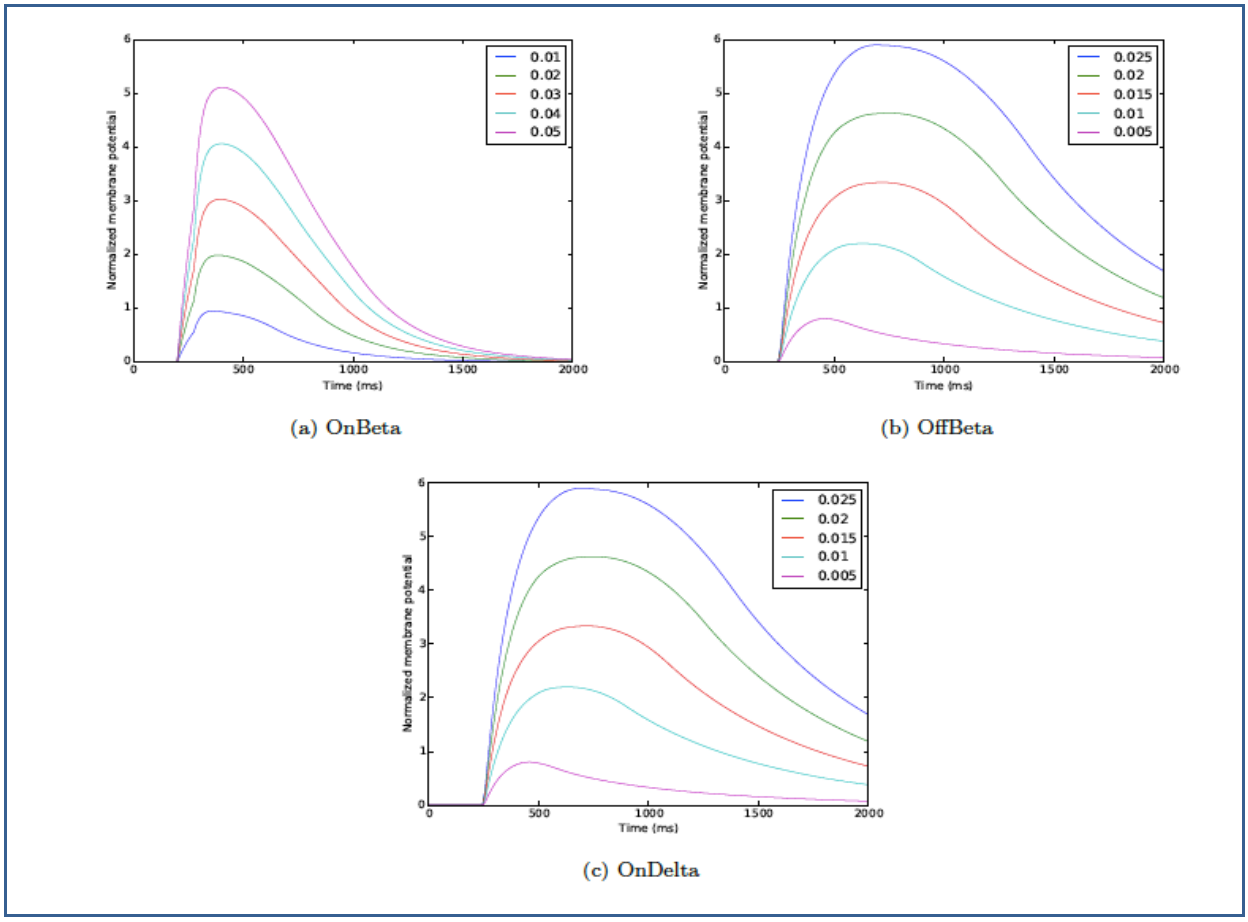

Figure 46: Normalized membrane potential responses for (a) OnBeta Cells, (b) OffBeta Cells, (c) OnDelta cells for an input spike at *t* = 200ms for different weights.

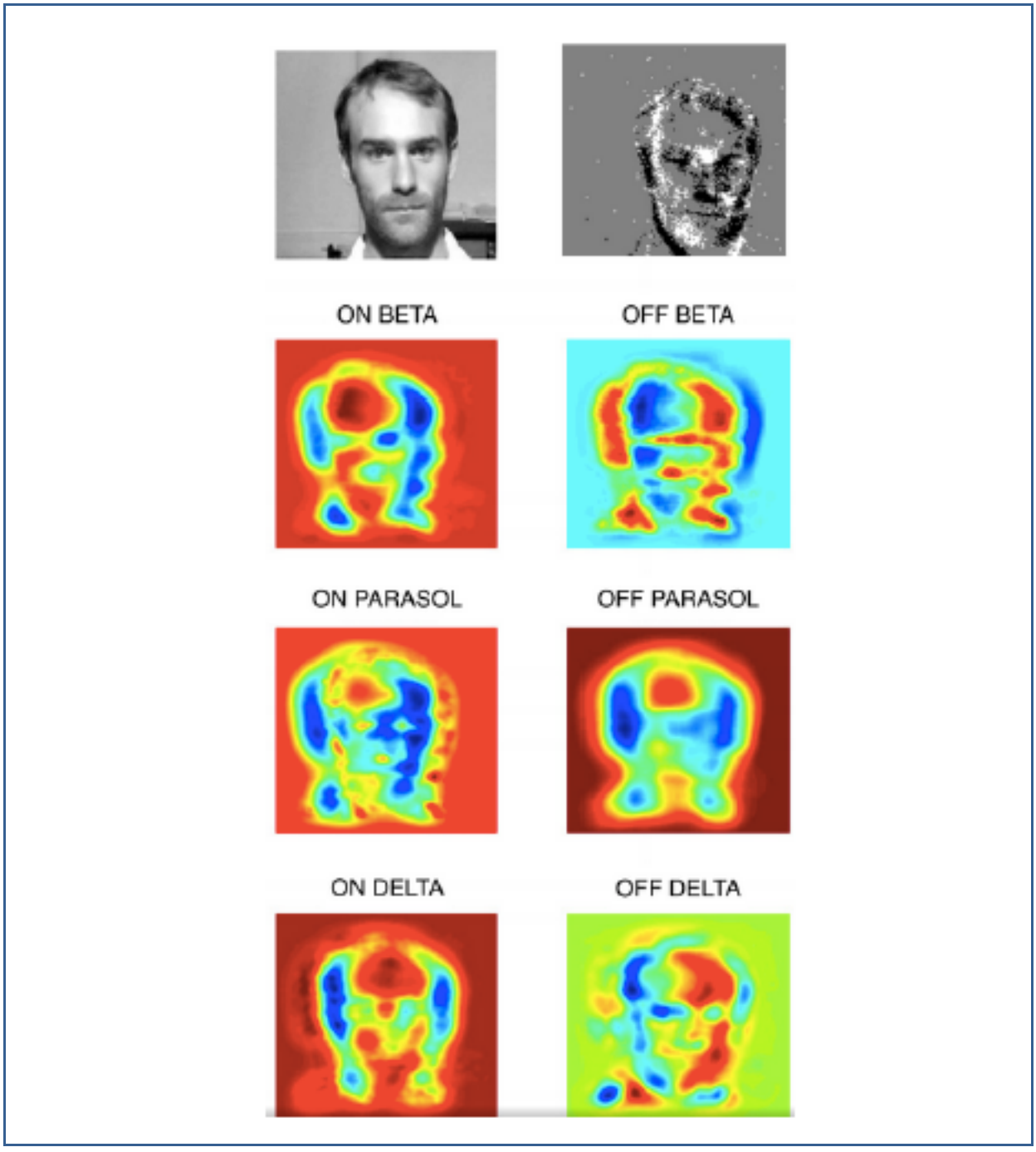

Figure 47:

4.6.6 Computational model: Stereovision (Milestone 11.3.5.7)

4.6.6.1 Temporal coincidence

Detecting temporal coincidence between two spikes is a widely used feature in spiking neural networks. As a consequence, we decided to implement a dedicated core for this task instead of using standard IaF neurons which would introduce an unnecessary overhead. Each neuron simulated by this core has two types of synaptic input and a time window. When an incoming spike is received on one input, the core will output a spike if another spike was received on its second input in the given time window. We added a refractory period to this process to limit the maximum firing rate of the neurons if required.

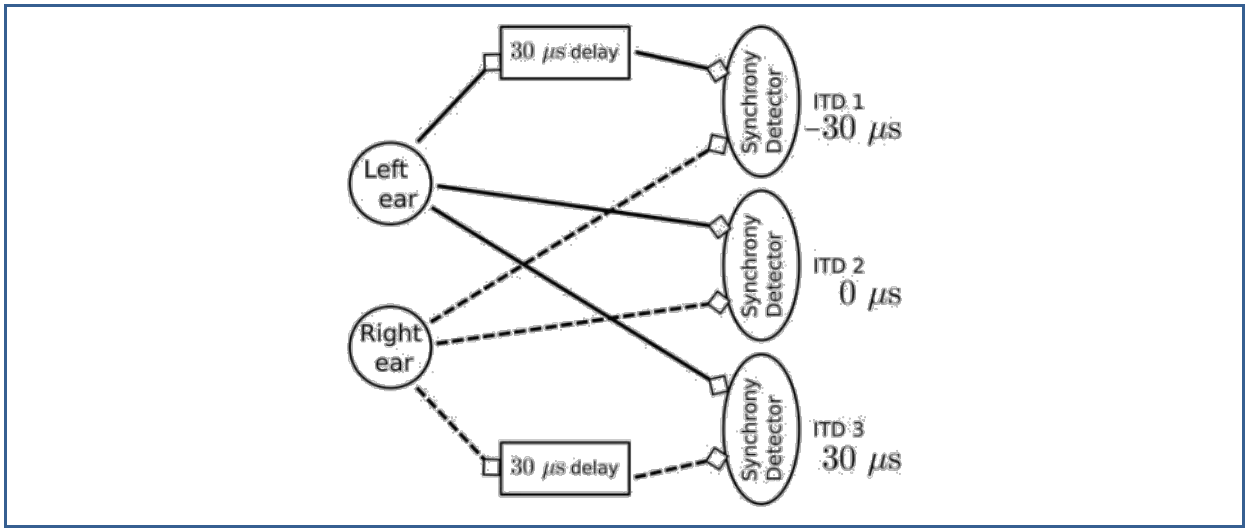

Figure 48: Model used to detect sub-millisecond spike synchrony for sound localisation.

The model consists of three synchrony detectors where each is activated by a different input interaural time difference (ITD). To achieve this result, each detector is directly connected to one ear whereas the second input comes from a dendritic delay population. For positive ITDs, spikes from the right ear are delayed. For negative ITDs, spikes from the left ear are delayed. In this experiment, spike trains from the two ears are simulated for three sound sources localized at positions producing expected ITDs.

To test the dendritic delay and synchrony detector models, we simulate a standard network used for sound localisation. This model is presented in Figure 48 and results are presented in Figure 49. For each ear, we consider a population of neurons representing 10 different frequency channels (Panel (a) in Fig.49). We start by generating spike trains with an interspike interval (ISI) of 100 μs for each of these channels and we feed them in the right ear (red dots). Then, this simulated sound is shifted in time according to the input interaural time differences (ITDs) corresponding to values compatible with human hearing: −30 (phase (1)), 0 (phase (2)) and 30 μs (phase (3)) to generate the input spikes for the left ear (blue dots). Some noise is then added independently to spikes from each ear and each channel by jittering each spike randomly between −5 and 5 μs to get the actual input presented in Figure 49(A). Each ear is then input in delay lines and synchrony detectors such as to detect the corresponding ITDs, synchrony detectors are, because of their associated delay lines, centred around −30, 0 and 30 μs with a window of 15 μs. These detectors are colour coded in Fig. 49(B) with detectors for ITDs −30, 0 and 30 μs respectively corresponding to red, blue and green dots. We can see that the different input ITDs are correctly extracted by the architecture for each phase of the input pattern. This model, while being simple and related to auditory information processing, shows how synchrony detection can be exploited in a biologically inspired model to compute information; it is worth noting that the same architecture hereby used is also used as the basis of the stereovision model to compute disparity.

4.6.6.2 Stereovision

We perform stereovision using the asynchronous high temporal resolution properties of the ATIS camera as developed in Rogister et al. (2012).[20] The combination of spatial and temporal constraints fully uses the high temporal resolution of neuromorphic retinas. It allows us to produce an optimal stereo algorithm that is able to perform stereo computation by detecting coactive pixels laying on the same epipolar lines. We illustrate how the model works through an example on a single epipolar line. We first characterize the model running a simple example on synthetic data, modelling a moving bar. In the first experiment the bar moves horizontally, parallel to the plane containing the two cameras. This corresponds to pixels (neurons) in the epipolar lines being successively activated with a certain speed and offset. In particular in our simulated experiment we chose a speed of 1 pixel every 10 ms and an offset of 10 pixels. Stimulation starts at t = 100 ms, and spans the

whole length of an epipolar line (64 pixels). Figure 50 shows the raster plots for the neurons in the two epipolar lines, where neurons get activated at the same time but with a spatial offset of 10 pixels. Figure 51 shows the activation of the neurons in the disparity plane. As the line is moving parallel to the cameras, the disparity does not change; this corresponds to the activity in the disparity plane moving in a diagonal shifting position, but not disparity. In order to evaluate the disparity, we plot therefore the difference between the coordinate of each epipolar line as represented in the disparity plane or, in other words, the difference between the X and Y coordinate. Figure 52 shows the results of this computation, with a constant disparity of -10.

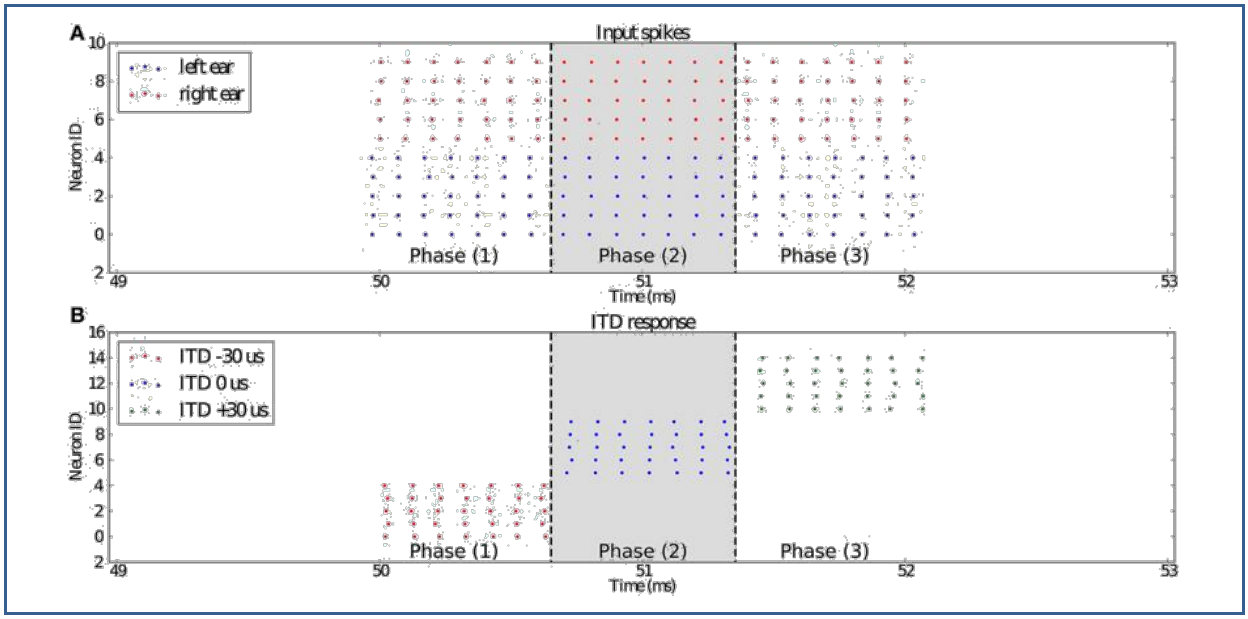

Figure 49: Input and output plots

Top plot (a) shows the input spikes to the system. It is comprised of the spikes (5 channels, red dots) from the right ear and the spikes (5 channels, blue dots) from the left ear. They correspond to a sound source positioned at ITD −30 μs for the first third (phase (1)) of the input stimulus, then ITD 0 μs for the second third (phase (2)) and 30 μs for the last part (phase (3)). Bottom plot (b) presents the outputs of the three synchrony detectors of the network configured to respond to ITDs −30 μs (red), 0 μs (blue) and 30 μs (green). We can see that for each channel, the detector with its preset ITD fired correctly in each phase of the experiment.

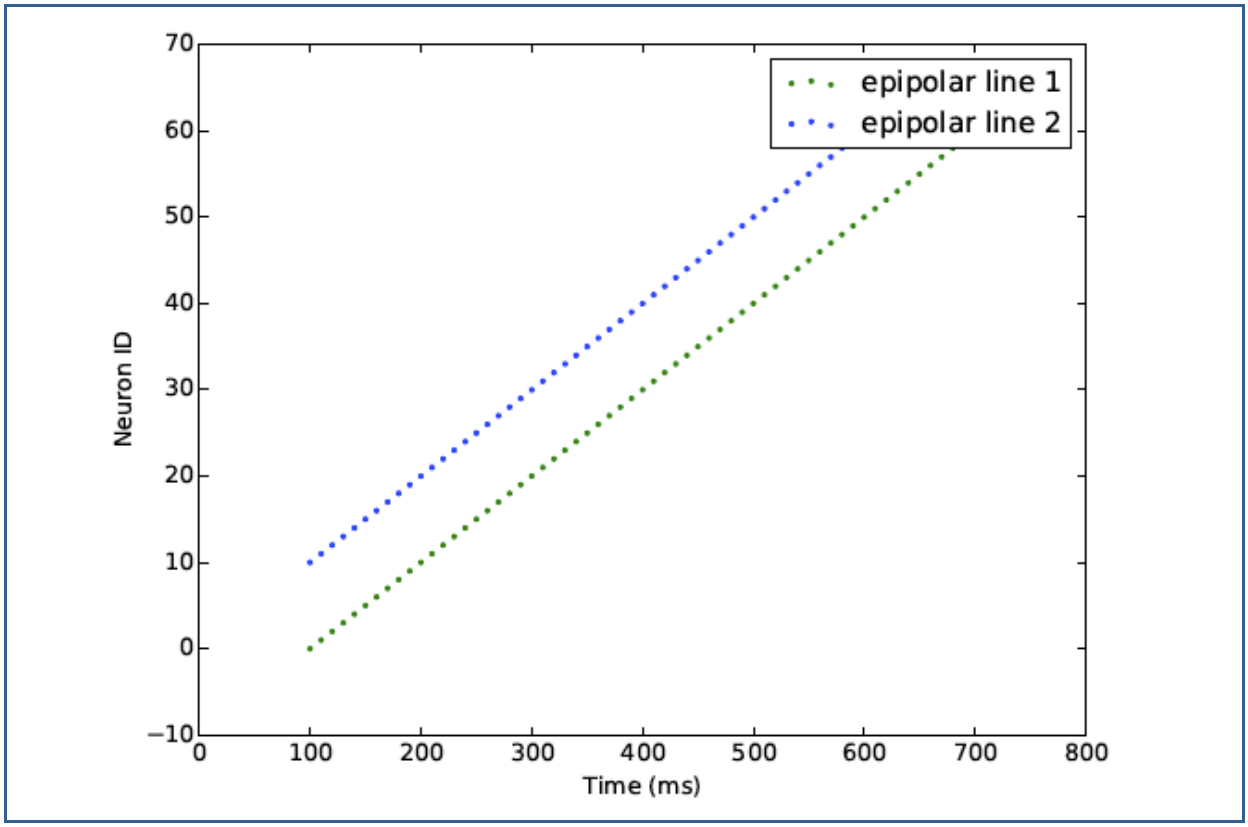

Figure 50: Stimulation starts at t = 100 ms, and spans the whole length of an epipolar line (64 pixels).

The figure shows the raster plots for the neurons in the two epipolar lines, where neurons get activated at the same time but with a spatial o set of 10 pixels.

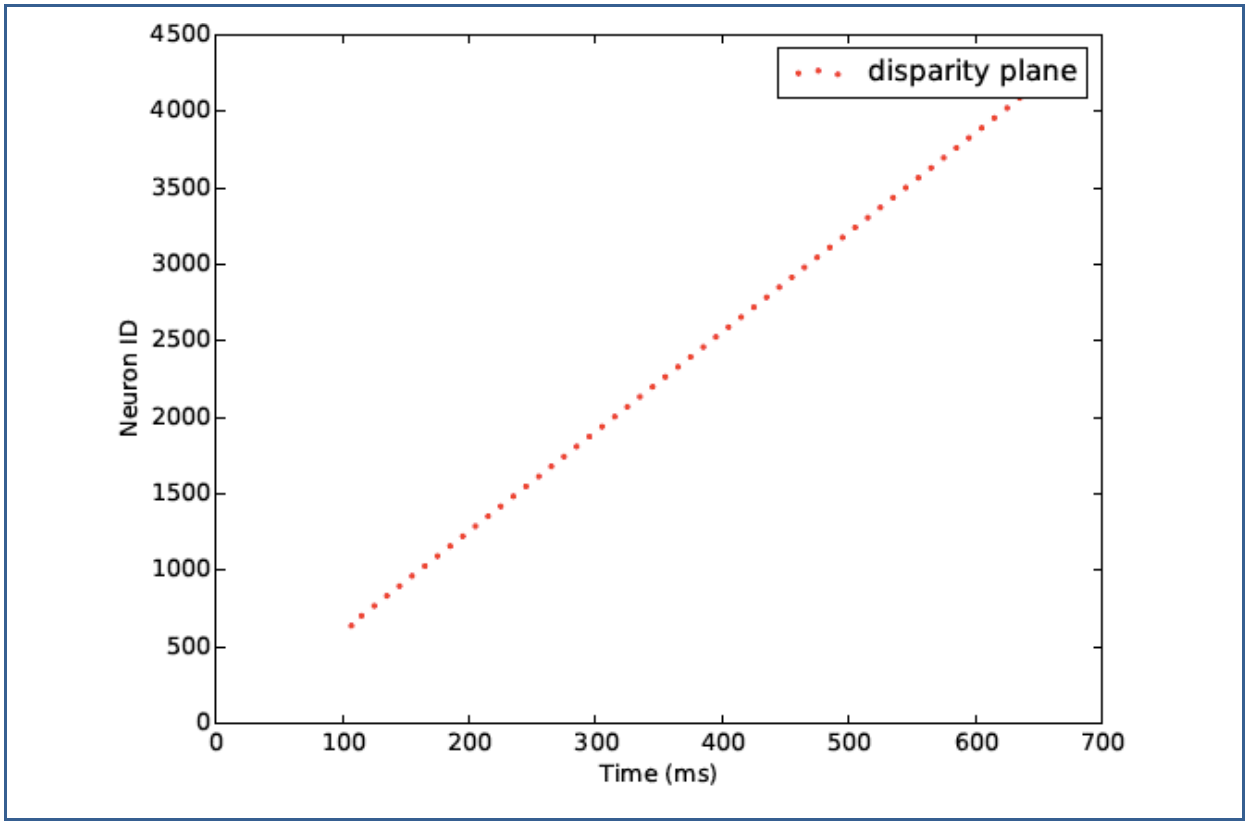

Figure 51: Raster plot of the neurons in the disparity plane.

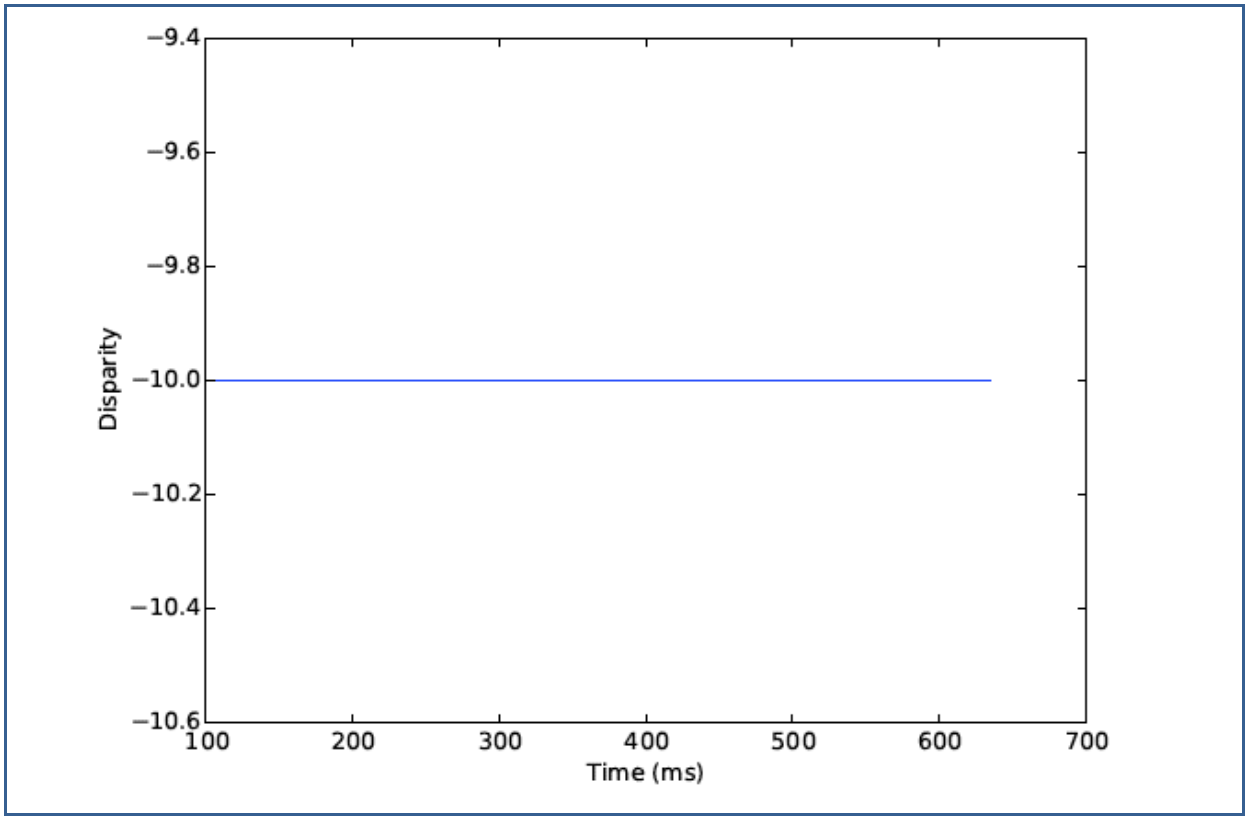

Figure 52: Disparity computed as the difference between the coordinate of each epipolar line as represented in the disparity plane.

In other words, the difference between the X and Y coordinate. Disparity is constant as the bar moves parallel to the cameras.

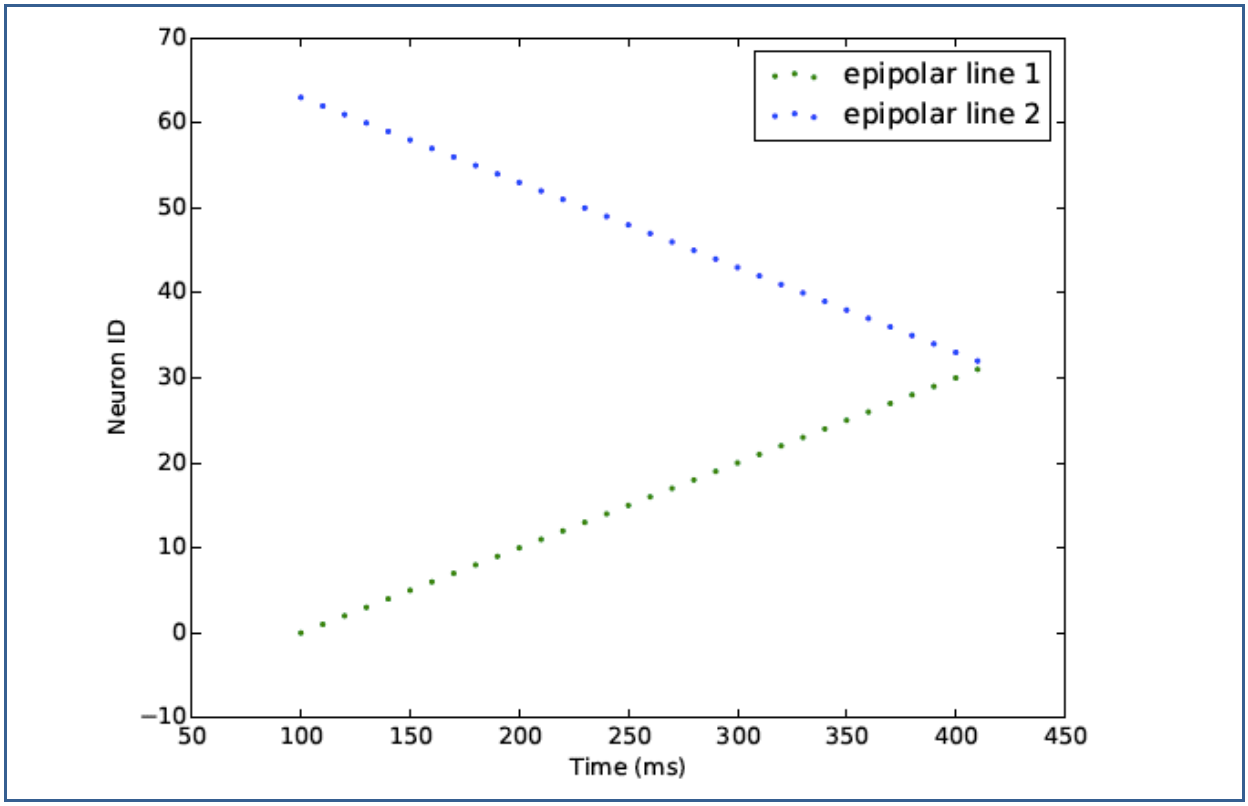

Figure 53: Raster plot of the epipolar line populations.

At the beginning of the stimulation the bar activates the left and right edge of the two epipolar lines respectively, corresponding to a maximum disparity; the bar starts moving away from the camera, hence moving towards the centre of each epipolar line, where they meet at the end of the stimulation; this corresponds to a location where disparity is equal to zero, when the bar is far away from the cameras (theoretically at infinity).

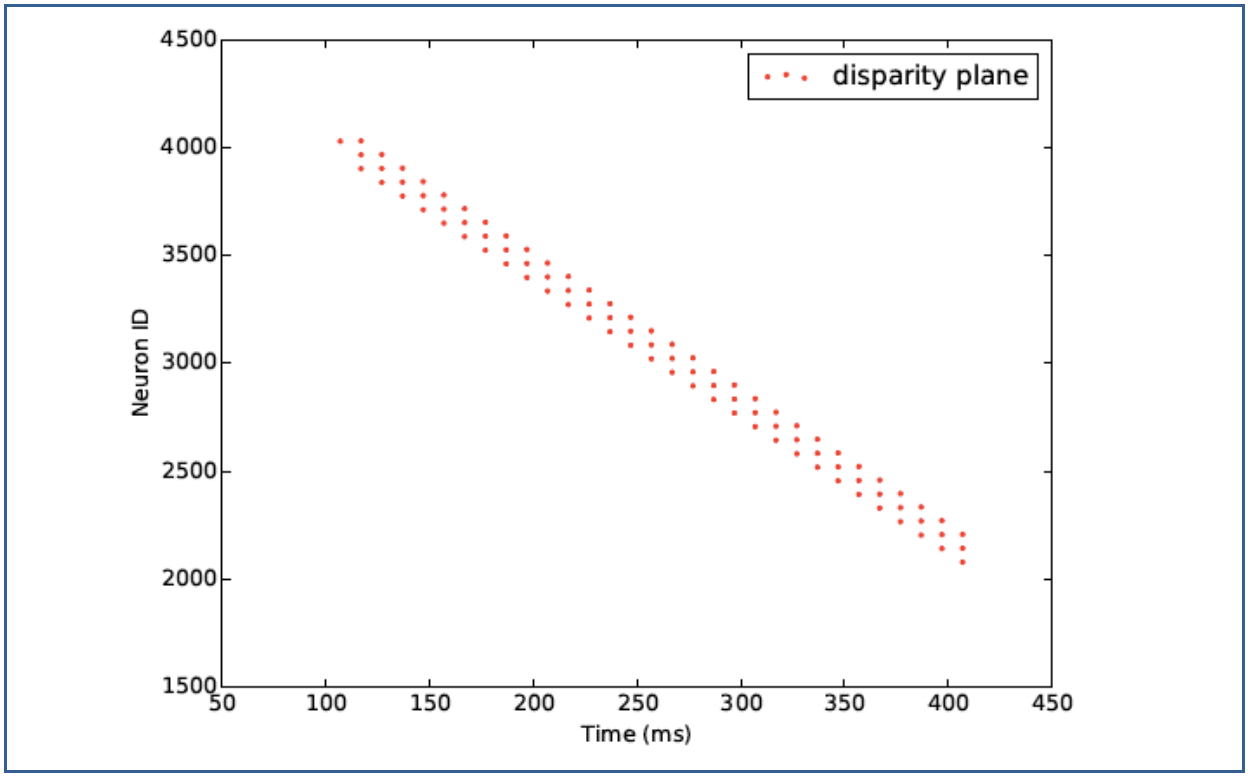

Figure 54: Raster plot of the neurons in the disparity plane.

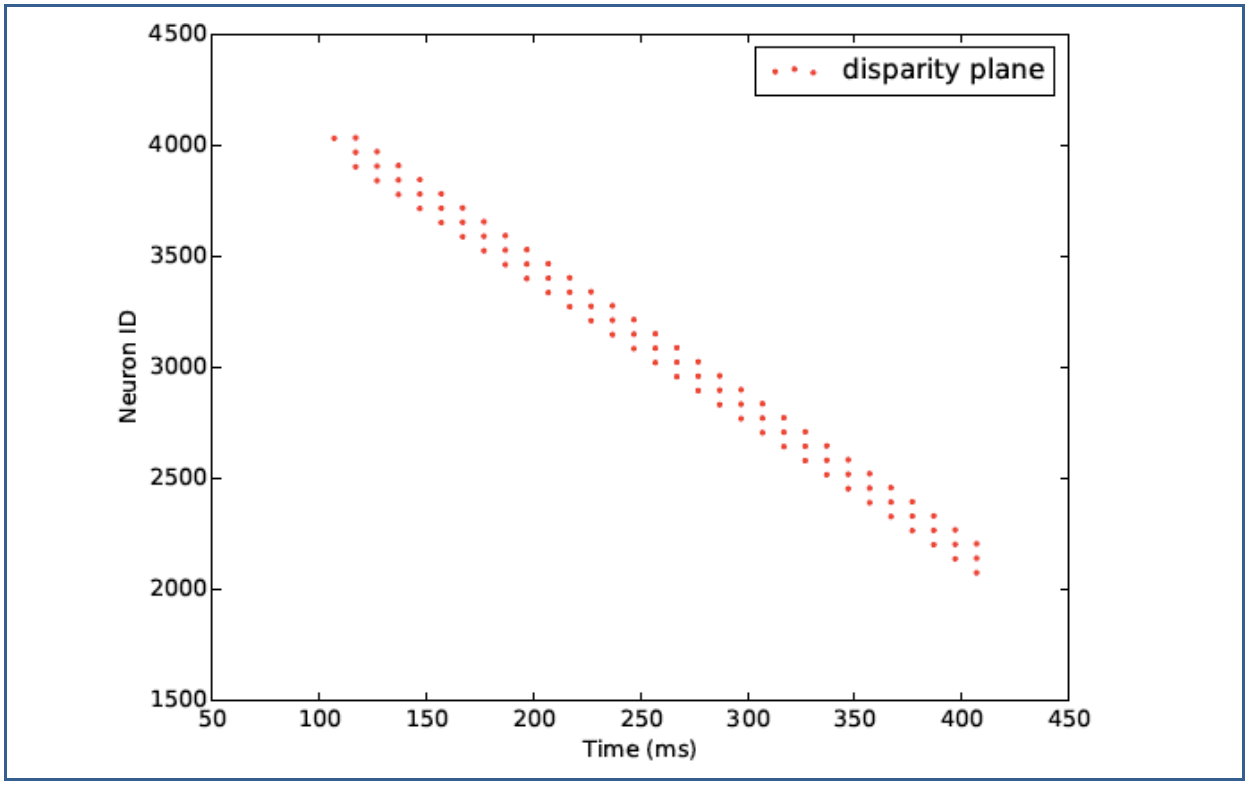

Figure 55: Disparity computed as the difference between the coordinate of each epipolar line as represented in the disparity plane.

In other words, the difference between the X and Y coordinate. Disparity decreases as the bar moves further away from the cameras.

We then repeat the experiment, this time simulating a line moving perpendicularly to the camera plane, spanning the whole overlapping visual field, starting close the cameras and moving further away. This corresponds to an opposite activation in the two epipolar lines: at the beginning of the stimulation the bar activates the left and right edge of the two epipolar lines respectively, corresponding to a maximum disparity; the bar starts moving away from the camera, hence moving towards the centre of each epipolar line, where they meet at the end of the stimulation; this corresponds to a location where disparity is equal to zero, when the bar is far away from the cameras (theoretically at infinity). The temporal activation of the two epipolar lines can be observed in Figure 53 showing their raster plot. For one epipolar line the first neuron is activated at the beginning of the simulation, and as time passes by activity shifts towards the center. The second epipolar line population shows the opposite pattern of activation; the last neuron fires at the beginning of the simulation, and then activity shifts towards the middle neuron. The same central neuron is activated at the end of the simulation, when the disparity is equal to 0. Figure 54 shows the activity in the disparity plane, Figure 55 shows the calculation of the disparity computed as in the previous case. It can be observed from the Figure that disparity starts from a maximum and then decreases until 0, in line with our expectations and the stimulus design.

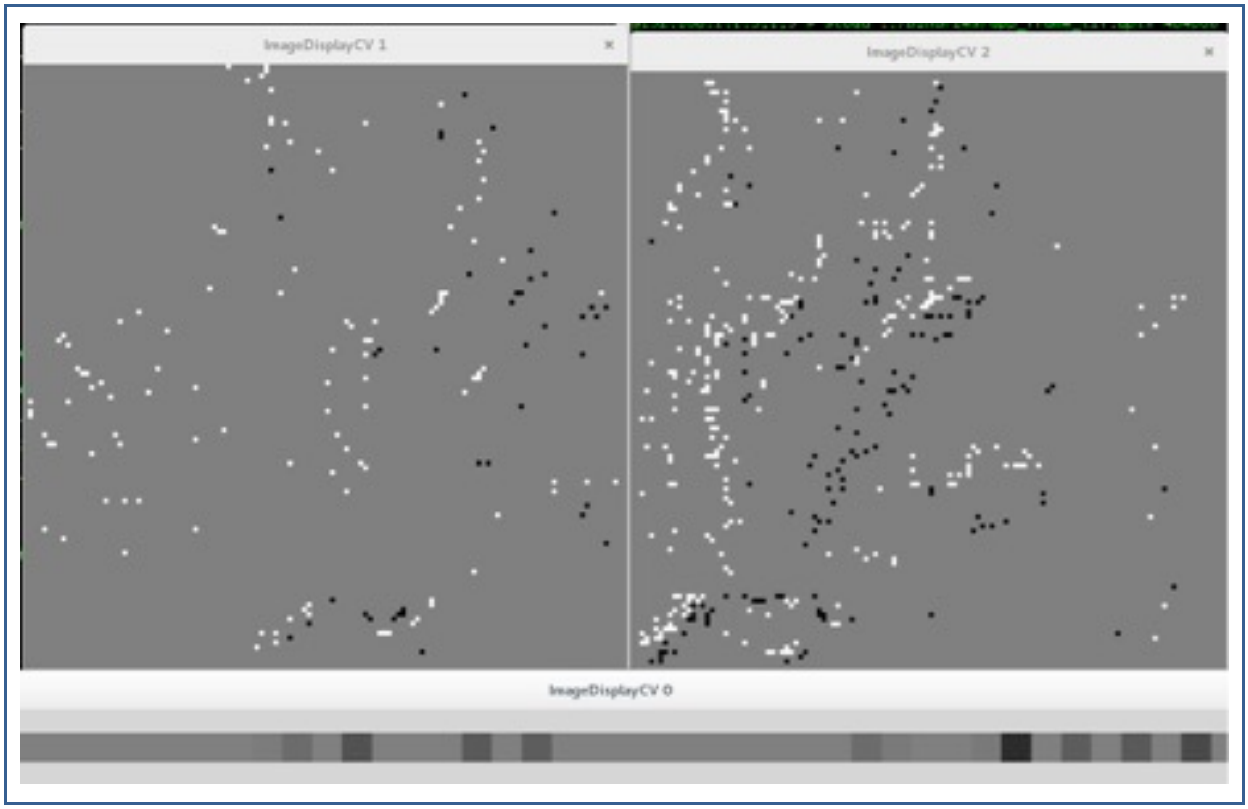

Figure 56: Live demo of disparity computation from the input from two ATIS.

When the stimulus is close to the cameras the disparity, colour coded in the horizontal bar, is higher and encoded in darker colours.

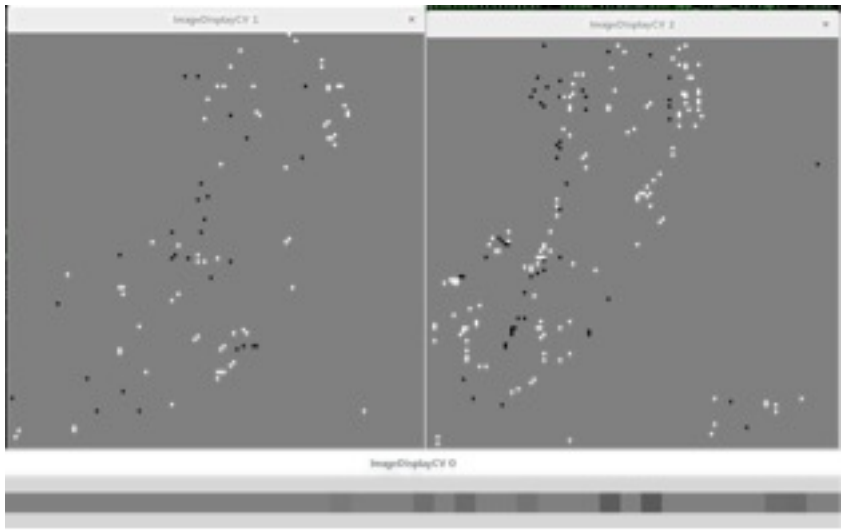

Figure 57: Live demo of disparity computation from the input from two ATIS.

When the stimulus is far away the disparity, color coded in the horizontal bar, is lower and encoded in lighter colors.

This same experiment is repeated by taking inputs from the two cameras and feeding them live on SpiNNaker. The following figures show different cases, obtaining waving an hand in front of the cameras. We show the events coming from each silicon retina and the result of the disparity computation. The horizontal position represents the location of the stimulation, while the colour encodes the disparity value: bigger levels of disparity corresponds to darker shades of grey or, in other words, to the stimulus being closer to the

camera (Figure 56). Likewise, lighter shades of grey corresponds to a lower level of disparity, representing an object further away (Figure 57).

4.7 T11.3.6 (UOS): Implementing a Spiking Classifier Network on HiCANN

Table 9: Data sets, neuron models, classifiers and implementation status in T11.3.6

Data set	Neuron Model	Classifier	Implementation status (NM-PM1, NM-MC1)	Parameters evaluated	Comments
Enose timeseries recordings of 20 odours	Spiking classifier	GeNN	GPU – Titan Black	Num. odors (classes), neuron count, virtual receptor (VR) count	Initial work was funded by EPSRC eFutures
MNIST	Spiking classifier	GeNN	GPU – Titan Black	Num. digits (classes), neuron count, virtual receptor (VR) count	This work adapted the classifier design to handle the static, high dimensional MNIST dataset
MNIST	Spiking classifier	PyNN/ SpyNNAker	SpiNNaker SPiNN3 (NM-MC1)	Num. digits (classes), neuron count, virtual receptor (VR) count	An implementation of the design was created for the SpiNNaker platform (4-chip SpiNN3 board)
MNIST	Spiking classifier	PyNN/ SpyNNAker	SpiNNaker SPiNN5 (NM-MC1)	Num. digits (classes), neuron count, virtual receptor (VR) count	An implementation of the design was created for the SpiNNaker platform (49-chip SpiNN5-board)

(1): E.g.: Performance, speed, power, etc.

4.7.1 Description of data sets, neuron models (or counts) and classifier in experimental set-up

Initial work on the neuromorphic classifier targeting a dataset of Enose time series recordings of 20 odors received EPSRC eFutures funding and was completed under HBP. The work then switched focus to obtaining a reasonable classification performance from the full MNIST hand-written digit recognition dataset. This is a high dimensional dataset comprising tens of thousands of observations. The classifier design has been modified and adapted for implementation on different neuromorphic platforms. We have analysed how each implementation performs differently with varying model sizes (up to 30,000 neurons, 18 million synapses), with different number of classes (2-10 digits), and when incorporating increasing numbers of "virtual receptors" (VRs) in input space (20 – 500+). The second part of the work focused on using a large model on higher capacity "large-scale" hardware. We used the SPiNN5 SpiNNaker board for this as availability of the HICANN "wafer-scale" platform was delayed.

The implementation on the various platforms is detailed and compared in a paper in Frontiers In Neuroscience (see Outreach). The findings and conclusions of the subsequent "scaling-up" exercise are detailed in our milestone report (see Outreach).

4.7.2 Implementation of classifier architecture on Neuromorphic Computing Platform and evaluated parameters

Comparative implementation of the same classifier design using three contrasting neuromorphic computing platforms under a set of evaluated parameters is fully detailed in our publication (Diamond et al, Front. Neurosci 2016; see Outreach). The implementation on the large scale SpiNN5 board is detailed in our milestone report (see Outreach). The implementations on the SPiNN3 and SPiNN5 have been released for general use as open source software (see Outreach).

4.7.3 Description of major use case(s) and target users of your application

The primary use case is to investigate a generic pattern recognition model applicable to run on neuromorphic hardware. Target users are users in research and development who want to employ neuromorphic classifiers in their work, and researchers who want to explore derived classifier concepts in their own research.

4.7.4 Documentation of experimental set-up

Documented implementations of the experimental setup on the SPiNN3 and SPiNN5 have been made publically available as a github repository. See project "spinnaker-neuromorphic-classifier" (https://github.com/alandiamond/spinnaker-neuromorphic-classifier/). A related DIC was filled in the Dataset_Information_Catalog/ Collaboratory.

4.7.5 Outreach

Journal Papers, peer reviewed:

- Diamond, A., Nowotny, T., and Schmuker, M. *Comparing Neuromorphic Solutions in Action: Implementing a Bio-Inspired Solution to a Benchmark Classification , on Three Parallel-Computing Platforms*. Frontiers in Neuroscience, 2016. **9:**491. doi:10.3389/fnins.2015.00491. This paper has received over 400 views in its first 2 weeks of publication.
- Diamond, A., Schmuker, M., Berna, A., Trowell, S., and Nowotny, T. *Classifying continuous, real-time e-nose sensor data using a bio-inspired spiking network modeled on the insect olfactory system*. Bioinspiration and Biomimetics (in print). This work was partly funded by the HBP and examines the use of this classifier model on neuromorphic GeNN/GPU platform to classify real chemical sensor responses. The paper has been accepted for publication (as of January 2016).

Conference Presentation and Poster:

- Diamond, A., Schmuker, M., Yavuz, E., Turner, J., and Nowotny, T. (2015). *Implementing neuromorphic Computing with Large, High-Dimensional Data Sets Using GeNN - a Meta-Compiler for Neuronal Modelling on General Purpose GPU-Accelerators*. BIH2015 –International Conference on Brain Informatics and Health, London, UK, 30 August – 2 September 2015. p21.

Annex A: Overview Component-Task linkage SP11_RUP to SGA1

RUP Task	Component	Description	Component Owner	Component type	*DIC/ PLA	Related SGA1 Use Case(s)
T11.1.1	HBP visual crowding	Python source code for a NEST simulation of visual cortex that investigates visual crowding. Includes an analysis script that is related to the draft document describing the model and its relation to psychophysical data.	Greg Francis	Software	☑	
T11.1.2	Source code for integrated brain-body control		Florian Röhrbein	Software	☑	SP10 - Manipulation experiments with humanoid robots and human avatars SP10 - Mouse rehabilitation experiment in the Neurorobotics platform
T11.2.1						
T11.3.1	Software model and network architecture	Description in upcoming publication: Online Trainable Spiking CMAC Network by Deiseroth B., Brefeld U., Debes Ch., and Gottfried F.	SAP	Publication	☑	This task ended in RUP and has no link to SGA1.
T11.3.2	NEAL Code	There is java and python code to run simulations on SpiNNaker, HiCANN, in Nest and in standalone java.	Chris Huyck	Software	☑	
T11.3.3	Source code, data-set, and configuration	Source files of implementations along with the dataset and instructions how to configure and run the experiments.	Bernabe Linares	Software	☑	This task ended in RUP and has no link to SGA1.

T11.3.4	Spiking associative network simulation and evaluation software and data	Platform-neutral implementation of associative memory benchmarks with supporting software for running simulations on different target platforms, primarily the HBP neuromorphic hardware platforms. Also includes software tools to support design space exploration of system and neuron parameters and evaluate the results.	Michael Thies	Software	☑	SP10 - Manipulation experiments with humanoid robots and human avatars SP10 - Mouse rehabilitation experiment in the Neurorobotics platform
T11.3.5	How to run a retina model, stereo vision and optic flow models on a 4-chip SpiNNaker board	Software, PyNN scripts and interface examples on how to run the retina model, stereo vision and optic flow models on the 4-chip SpiNNaker board. The files contained in this directory are used to initialize, configure, start and stop the interface between the ATIS camera and the 4-chip SpiNNaker board.	Ryad Benosman	Software	☑	This task ended in RUP and has no link to SGA1.
T11.3.6	Spinnaker-neuromorphic classifier	As an output, the SpiNNaker implementation for large-scale version of neuromorphic classifier network, including the C-based library developed for live spike injection and collection is available alongside the earlier spike-source based classifier model for SpiNNaker. The repository is available at the GITHUB project "spinnaker-neuromorphic-classifier".	Alan Diamond	Software	☑	

**DIC: Dataset Information Card – HBP internal management tool*

**PLA: Project Lifecycle App - HBP internal management tool*

Annex B: References

[1] Martínez-Cañada, P., Morillas, C., Romero, S., and Pelayo, F. *Modeling retina adaptation with multiobjective parameter fitting*. In: Advances in Computational Intelligence. Springer International Publishing, 2015. pp. 175-184.

[2] Martínez-Cañada, P., Morillas, C., Nieves, J.L., Pino, B., and Pelayo, F. *First stage of a human visual system simulator: the retina*. In Computational Color Imaging. Springer International Publishing, 2015. pp. 118-127.

[3] Francis, G., Manassi, M., and Herzog, M.H. Neural dynamics of grouping and segmentation explain properties of visual crowding. In preparation.

[5] Suk, H.-I., Shen, D. Deep learning-based feature representation for AD/MCI classification. Medical Image Computing and Computer-Assisted Intervention–MICCAI, 2013. Pp.583-590.

[6] Kutalik, Z., Beckmann, J.S., and Bergmann, S. *A modular approach for integrative analysis of large-scale gene-expression and drug-response data*. Nature Biotechnology, 2008. **26:**531-539.

[7] Broderick, T., Boyd, N., Wibisono, A., Wilson, A.C., and Jordan, M. I. *Streaming variational Bayes*. NIPS 2013.

[8] Yousefzadeh, A., Plana, L., Temple, S., Serrano-Gotarredona, T., Furber, S., and Linares-Barranco, B. *Fast predictive handshaking in synchronous fpgas for fully asynchronous multi-symbol chip links. application to spinnaker 2-of-7 links*. IEEE Transactions on Circuits and Systems II: Express Briefs, 2016. **PP:**1–1.

[9] Temple, S. *AppNote 1 - SpiNN-3 Development Board*. University of Manchester, Manchester, Technical Report, 2011.

[10] Plana, L.A. *AppNote 8 - Interfacing AER devices to SpiNNaker using an FPGA*. University of Manchester, Manchester, Technical Report, 2013.

[11] Galluppi, F., Denk, C., Meiner,M.C., Stewart, T., Plana, L.A., Eliasmith, C., Furber, S., and Conradt, J. *Event-based neural computing on an autonomous mobile platform*. Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2014. pp. 1–6.

[12] X. Inc. Aurora 8B/10B Protocol Specification. Technical Report, 2011.

[13] Iakymchuk, T., Rosado, A., Serrano-Gotarredona, T.,Linares-Barranco, B., Jimenez-Fernandez, A., Linares-Barranco, A., and Jimenez-Moreno, G. *An aer handshake-less modular infrastructure pcb with x8 2.5 gbps lvds serial links*. In: Circuits and Systems (ISCAS), 2014 IEEE International Symposium on Circuits and Systems. IEEE, 2014. pp. 1556–1559.

[14] Berner, R., Delbruck, T., Civit-Balcells, A., and Linares-Barranco, A. *A 5 meps 100 usb2. 0 address-event monitor-sequencer interface*. In: Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on Circuits and Systems. IEEE, 2007. pp. 2451–2454.

[15] http://www.visionect.com/

[16] Giulioni, M., Lagorce, X., Galluppi, F., and Benosman, R.B. *Event-based computation of motion flow on a neuro-morphic analog neural platform*. Frontiers in Neuroscience, 2016. **10:** http://dx.doi.org/10.3389/fnins.2016.00035.

[17] Barlow, H.B, and Levick, W.R *The mechanism of directionally selective units in rabbit's retina*. The Journal of Physiology, 1965. **178:**477, 1965.

[18] Lagorce, X., Stromatias, E., Galluppi, F., Plana, L.A., Liu, S.-C., Furber, S.B., and Benosman, R.B. *Breaking the millisecond barrier on spinnaker: implementing asynchronous event-based plastic models with microsecond resolution*. Frontiers in Neuroscience, 2015. 9: http://dx.doi.org/10.3389/fnins.2015.00206.

[19] Lorach, H., Benosman, R., Marre, O., Ieng,S.-H., Sahel, J.A., and Picaud, S. *Artificial retina: the multichannel processing of the mammalian retina achieved with a neuromorphic asynchronous light acquisition device*. Journal of Neural Engineering, 2012. **9:**066004.

[20] Rogister, P. Benosman, R., Ieng, S.-H., Lichtsteiner, P., and Delbrück, T. *Asynchronous event-based binocular stereo matching*. IEEE Transactions on Neural Networks Learning Systems, 2012. **23:**347–353. http://dblp.uni-trier.de/db/journals/tnn/tnn23.html#RogisterBILD12