A Generic Software Platform for Brain-Inspired Cognitive Computing

Koichi Takahashi[*123], Kotone Itaya[*12], Masayoshi Nakamura[*3], Moriyoshi Koizumi[*4], Naoya Arakawa[*3], Masaru Tomita[*2] and Hiroshi Yamakawa[*3]
[1] RIKEN QBiC, [2] Keio University Graduate School of Media and Governance,
[3] Dwango AI Laboratory, [4] Open Collector, Inc.
ktakahashi@riken.jp

Abstract
We have been developing BriCA (Brain-inspired Computing Architecture), the generic software platform that can combine an arbitrary number of machine learning modules to construct higher structures such as cognitive architectures inspired by the brain. We discuss requirements analysis and design principles of this cognitive computing platform, report its implementation, and describe plans for further development.

Keywords: software platform, cognitive architecture, machine learning, modularity, the whole brain architecture

1 Introduction

Inspired by the recent advance in neuroscience and machine learning (ML), we propose a hypothesis on reproducing the whole brain level functionality out of ML modules, to be tested with the software platform introduced below.

1.1 Background

There are trends in ML research such as the following: 1) ML research has been inspired by recent development of the cognitive neuroscience and often referring to neuro-scientific features of the brain. 2) Learning algorithms of heterogeneous paradigms have been combined to realize performance or functionality that could not be attained with a single paradigm (Vinyals et al. [1], Karpathy and Fei-fei [2], Minh et al. [3], and Gao et al. [4]). 3) The breakthrough in acquiring distributed representations with deep learning (Le et al. [5]) may have resolved the basic issues in AI such as the frame problem, the symbol grounding problem and knowledge acquisition bottleneck.

1.2 The Hypothesis

The following is the hypothesis on which the current article is based:

> The brain combines modules, each of which can be modeled with a machine learning algorithm, to attain its functionalities, so that combining machine learning modules in the way the brain does enables us to construct a generally intelligent machine with human-level or super-human cognitive capabilities.

The hypothesis involves the three following sub-hypotheses:

1) Modularity of the Brain

We shall assume the hypothesis that the brain is constituted of computationally independent modules (the modularity hypothesis). The brain apparently is constituted of anatomically independent modules in various levels, such as organs like neo-cortices, hippocampi, and basal ganglia, cortical areas like the primary visual cortex and the primary motor cortex, and sub-organ loci like CA1 and CA3 in the hippocampus.

2) Brain Organs and Machine Learning

For the hypothesis above to hold, the brain modules must be modeled in a functionally encapsulated manner. In other words, they must be modeled in the functional level (or Marr's computational level). Here, we assume that the learning functionality of brain modules can be modeled with the functionalities of (known or to-be-invented) ML algorithms (the machine learning hypothesis).

3) Emergent Cognitive Functions

It is not evident that combining ML modules yields the whole brain cognitive functionality (the emergence hypothesis). The system should be carefully designed to attain desired functional emergence and put to the test. Meanwhile, we could follow suit with the effort in the ML discipline to combining algorithms as mentioned above.

2 Platform Requirement

Our aim here is to test the hypothesis above and to realize highly capable cognitive systems, hopefully at the human level. As the attainment of knowledge and techniques in neuroscience, machine learning, and cognitive architecture would require decades to come, the effort should be better done on a long-lasting platform. While a research organization could also be an important platform, the current paper focuses on the software platform for testing the hypothesis. The platform is called BriCA (Brain-inspired Computing Architecture) and is intended to support the build-and-test approach in an efficient way.

2.1 The Necessity of a Software Platform

To test the hypothesis above, models must be implemented (for the build-and-test approach). This requires an implementation mechanism that simultaneously runs heterogeneous ML modules and arbitrates them in an efficient manner.

The construction of the entire cognitive architecture also requires various processes such as brain architecture modeling, cognitive architecture design, research and development of ML algorithms, cognitive architecture fostering (by making them learn in a certain environment), and application to products. These processes would involve collaboration of numerous participants in the community and certainly be helped with a software platform that integrates them.

2.2 Desirable Features

BriCA as the platform is required to fulfill the following conditions:

1. Modules can be used as 'libraries' so that new or ready-made implementations of ML algorithms can be used as plug-ins.
2. Module-submodule hierarchy should be supported to mimic the hierarchy of organs in the brain.
3. Arbitrary modules in the platform can be connected via a unified method of communication.

4. The cognitive architecture can interact with the environment via a sensor-actuator interface, and work with other platforms such as the Robot OS or game engines.
5. The platform has a scheduler that arbitrates asynchronous calls among its modules. When the environment is real, the scheduler must work in real-time.
6. The platform is scalable with regard to the number of modules and processing speed.
7. The platform has a mechanism for supporting learning curricula in order to deal with the problem of combined learning (discussed later).
8. The platform is capable of distributed development.
9. The platform provides with user interfaces for managing experimental setting (and internal states of modules, if possible).
10. The platform should not be proprietary.

Some of the requirements are met with robotics middleware such as ROS (ros.org) and MIRA[*], data analysis platforms such as Weka[†], Garuda[‡], Jubatus (jubat.us), and Neural Network Toolbox™[§], and modular simulation environments such as Simulink[**] and E-Cell (Takahashi et al. [6]). Neural simulators such as Nengo (Eliasmith [7]) and *emergent* (Aisa et al. [8]) are also relevant. However, robotics middleware such as ROS and MIRA does not support hierarchical design, neural simulators are committed to the algorithmic levels, which we do not want, Neural Network Toolbox™ is proprietary, and other platforms do not support the multi-ML module communication requirement. Thus, we could not find a platform that satisfies all of our requirements above and set out to design a platform of our own.

3 Design and Implementation

3.1 The BriCA Core

The core of the platform implements a CognitiveArchitecture, consisting of a Scheduler and Module instances and Connection instances. A Module is an abstracted ML module with input and output ports, the update interval, and internal states. An output port can be connected to an input port via a Connection instance. Time elapses as the Scheduler calls Module instances according to the update interval. A port value is a numerical vector of an arbitrary length (a signed 16 bit integer in the current implementation).

Each module has the following method:

```
out, states <- fire(in, states)
```

Namely, a Module instance updates the values of output ports and the internal states solely dependent on the values of input ports and the current internal states by calling the user-defined `fire()` method. The operation is carried out in three phases:

1. `input`: updates input ports according to the values of the connected output ports
2. `fire: result, states <- fire(in, states)`
3. `output: out <-result`

The `fire` phase keeps the result in the result buffer so that the I/O buffers are modified only in the two critical I/O phases. This is to make the system scalable by assuring parallel firing of modules.

[*] http://www.mira-project.org/joomla-mira/

[†] http://www.cs.waikato.ac.nz/ml/weka/

[‡] http://www.garuda-alliance.org/

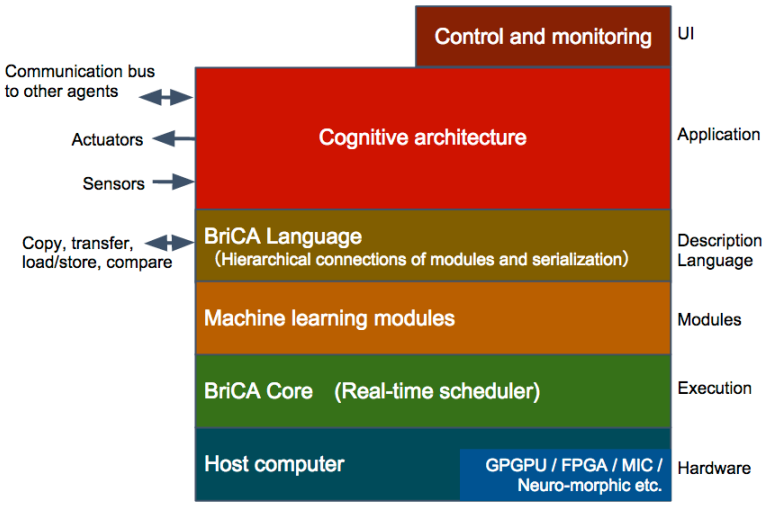
[§] http://www.mathworks.com/products/neural-network/

[**] http://jp.mathworks.com/products/simulink/

The firing of Module instances is discrete and the values of output ports are defined as piecewise constant (Zeigler et al. [8]). The firing scheduling is done by the discrete event scheduler that supports real-time and virtual time scheduling (Takahashi et al. [6]). Inter-module communication employs message passing (as opposed to shared memory) for the sake of scalability.

3.2 Module Reusability and the BriCA Language

As we expect collaborators to work together for improving module implementations while keeping other parts of the architecture intact, we have designed a domain specific language for collaborative architecture design and for circulating and archiving designed architectures. The language is to define the specification of modules, ports and connections and thus implementation independent. The language supports hierarchical (nesting) definition of modules. It uses object description languages such as JSON or Turtle for its syntactic representation.



4 Results

We have implemented Version 0 of the platform for proof-of-concept in Java and used it to implement a cognitive architecture called CITTA (Yamakawa et al. [10]). The current Version 1 is being implemented in Python. The core libraries for BriCA Version 1 are available for free use under the terms of the Apache License. The source code is accessible from the GitHub repository[††], and API documentation along with a basic tutorial is hosted at a GitHub Pages site[‡‡]. Some examples scripts are provided inside the python/examples directory within the V1 repository. Samples include code for utilizing classifiers from scikit-learn, a wrapper for a simple autoencoder implemented with Chainer, and an implementation of the NeuralTalk architecture with pre-trained weights. An interface for ROS integration is prepared as the brica1.ros module, which provides an adapter class that acts as a ROS node and can be connected to a BriCA agent via BriCA port interface. The BriCA language is also being implemented. Version 2 will provide multi-language support (combining modules written in heterogeneous computer languages), interfaces to other middleware that provide simulated and real environment, and more advanced real-time scheduling and scalability to run on multiple computer nodes.

[††] https://github.com/wbap/V1/
[‡‡] https://wbap.github.io/V1/

5 Discussions

5.1 Distributed Representation

On BriCA, modules exchange information in the numerical vector format (distributed representation), as do most of computational neural network and ML models. On the other hand, cognitive architectures often use symbolic graph representation such as trees and stacks (as was the case with CITTA implemented for Version 0). Thus, general 'design patterns,' such as decomposition by graph traversal, would be required to handle graphs with distributed representation. The computational neuroscientific modeling of cognitive processes that have been considered to require graph structure would also certainly help in this regard (Eliasmith [7]).

5.2 The Problem of Combined Learning and Curricula

In a system consisting of multiple ML modules, the varying I/O function of a module may wield influence on the function of another module (Sculley et al. [11]). For example, motion learning in the motor cortex would wield influence on the motion learning in the cerebellum. As learning is fixed from the lower layer in the development of visual cortices, curricula seem important for learning. Thus, it is desirable for the platform to support mechanisms for users to test, generalize, implement and share various curricula.

5.3 Fostering and Application

As the platform will be applied to various fields and environments such as robotics, games, and data analysis, it would be better combined with other platforms. As a practical cognitive architecture must learn in an environment, it is important to design the 'fostering' environment. In robotics, middleware such as ROS, as well as virtual robotic simulators such as Gazebo, is widely used. Game engines and data analysis platforms could work together with BriCA in the future.

5.4 Populating the Library

The utility of BriCA largely depends on the richness of its ML modules, which could profit from existing implementation. Open source generic ML libraries such as Scikit-learn (scikit-learn.org), Theano[§§], PyBrain (pybrain.org), PyLearn2[***], and Weka are available to be embedded in modules. Implementation for more specific functions includes SLAM implementation such as RatSLAM (Milford et al. [12]), neocortex models such as BESOM (Ichisugi and Takahashi [13]), DeSTIN (Arel et al. [14]) and HTM (George and Hawkins [15]), and cerebellar models such as MOSAIC (Haruno et al. [16]). Besides making these implementations available for BriCA, it would be also important to encourage the users to implement, evaluate, and share new ML methods.

5.5 Performance Issue

The rate of inter-neuron information processing would be slower than the maximum firing rate of about 1kHz. Thus, communication delays between BriCA modules would be required to be negligibly small with regard to 1ms regardless of the number of modules. If the negligible overhead is 1%, it

[§§] http://deeplearning.net/software/theano/
[***] http://deeplearning.net/software/pylearn2/

amounts to 10 μs (10^{-5}s). In Version 0, the intra-process data exchange is done synchronically and simplex delay is about 100 ns.

The computational power of the human brain is said to be about 1 PFLOPS. If the state-of-the-art processors have the computation speed of about 10 GFLOPS (with a moderate estimation), it means that about a hundred thousand cores are required. Then, as it is rather difficult for off-the-shelf hardware to guarantee the delay to be less than 10μs for arbitrary inter-thread communication with payload, we should also watch advances in hardware such as neurochips, as well as in software technology.

6 Conclusion

In this article, we reported the attempt of developing a software platform (BriCA) while posing a hypothesis for realizing brain-inspired cognitive machines. To prove the hypothesis, we have to construct exemplar realizations of emergent cognitive functionality by combining more than one ML modules. This task is currently being done with Version 1 and put to the test on hackathons. We are also intending to foster a community of researchers and developers of cognitive architectures in line with the hypothesis mentioned above.

Acknowledgment

We would like to show our gratitude to Yuuji Ichisugi, Ryutaro Ichise, Takashi Omori, Hideki Kashioka, Satoshi Kurihara, Takeshi Sakurada, Takeshi Sato, Makoto Taiji, Nishimoto Shinji, Hidemoto Nakada and Yutaka Matsuo for their kind advices. We also appreciate discussions with the youth group for the whole brain architecture. Finally, we note that the BriCA project is part of the efforts at the Whole Brain Architecture Initiative (NPO, wbai.jp).

References

[1] Vinyals O, Toshev A, Bengio S, Erhan, D. Show and Tell: A Neural Image Caption Generator. *arXiv*, 1411.4555; 2014

[2] Karpathy A, Fei-fei L. Deep Visual-Semantic Alignments for Generating Image Descriptions. *arXiv*, 1412.2306; 2014

[3] Mnih V, Kavukcuoglu K, Sliver D, Rusu AA, Veness J, Bellmare MG, Graves A, Riedmiller M, FidjeLand AK, Ostrovski G, Petersen S, Beattle C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. *Nature*; 2015

[4] Gao H, Mao J, Zhou J, Huang Z, Wang L, Xu W. Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question Answering. *arXiv*:1505.05612; 2015

[5] Le Q, Ranzato M, Monga R, Devin M, Chen K, Corrado G, Dean J, Ng A. Building high-level features using large scale unsupervised learning. *International Conference in Machine Learning*; 2012

[6] Takahashi K, Kaizu K, Hu B, Tomita M. A multi-algorithm, multi-timescale method for cell simulation. *Bioinformatics*; 2004

[7] Eliasmith C. *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford: Oxford University Press; 2013

[8] Aisa B, Mingus B, O'Reilly R. The emergent neural modeling system. *Neural Networks*; 2008:21:1146-52

[9] Zeigler BP, Prähofer H, Kim TG. *Theory of Modeling and Simulation*, Second Edition; Academic Press; 2000
[10] Yamakawa H, Okada H, Watanabe N, Matsuo K. Distributed Intelligent Architecture for Real World Autonomous Learning. In: *Real World Computing Symposium* 1998:253-8
[11] Sculley D, Holt G, Golovin D, Davydov E, Phillips T, Ebner D, Chaudhary V, Young M. Machine Learning: The High Interest Credit Card of Technical Debt. In: *Software Engineering for Machine Learning* (NIPS 2014 Workshop); 2014
[12] Milford MJ, Wyeth GF, Prasser D. RatSLAM: a hippocampal model for simultaneous localization and mapping. In: *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*; 2004, p.403-408
[13] Ichisugi Y, Takahashi N. An Efficient Recognition Algorithm for Restricted Bayesian Networks. In: *International Joint Conference on Neural Networks 2015*, (to appear).
[14] Arel I, Rose D, Coop R. DeSTIN: A Scalable Deep Learning Architecture with Application to High-Dimensional Robust Pattern Recognition. In: *Proc. of the AAAI 2009 Fall Symposium on Biologically Inspired Cognitive Architectures (BICA)*; 2009
[15] George D, Hawkins J. Towards a Mathematical Theory of Cortical Micro-circuits. *PLoS computational biology*; 2009:5:10:e1000532
[16] Haruno M, Wolpert DM, and Kawato M. Mosaic model for sensorimotor learning and control, *Neural Computation*; 2001:13:2201-20