Accelerator for Bayesian Cognition (ABC): A Stochastic Computing Platform for Cognition Computing

ABSTRACT

Building computers that learn and reason like people has been the dream for generations of human beings. Due to its massive representational breadth and depth as well as its ability to handle uncertainty, Bayesian cognition model is fast rising as a promising approach to account for a diverse range of human cognition behaviors. However, Bayesian computing can be expensive on modern computers because they are ill-equipped for probabilistic computations and memory operations. In this work, we propose Accelerator for Bayesian Cognition (ABC), which is a sampling based stochastic computing platform supporting efficient Bayesian probabilistic computations. ABC integrates three different types of resources into a single architecture: a stochastic computing engine for random sampling, a hardware based parallel Markov Chain Monte Carlo (MCMC) for estimating posterior distributions, and an exemplar memory model for emulating human memory behaviors. ABC is highly efficient in Bayesian computations. Evaluation shows that its FPGA implementation outperforms a four-core CPU in MCMC by a factor of 32 and beats a TITAN X GPU in Bayesian network learning by a factor of up to 10. Experiments demonstrate that ABC is able to effectively perform a range of cognition behavior at various levels. Especially, we show how to use ABC to perform visual concept learning, which is a basic cognition capability of human beings.

1. Introduction

Human brain, as an incredible cognition device capable of perception, induction, inference, and decision making, has long been arousing the curiosity of human beings. One example is that a typical child can learn 10^4 categories of objects by the age of six [1]. Infants learn how to make inference by implicitly following the Newton laws [2]. Moreover, the responsiveness and power efficiency of human brains are also amazing: people can perform inference in around ~20ms with a peak power budget of ~25W. As a result, understanding human brains and developing machines that learn and reason like people have been the ultimate goal of the artificial intelligence, neuroscience and cognition communities [3].

Marr outlined the three levels, namely computational level, algorithmic level, and implementation level, toward a systematic understanding of human brains [4]. The computational level considers the cognitive problem as well as the input and output. The algorithmic level focuses on how to solve the problem listed in computational level, while the implementation level involves the underlying physical realization. By decoupling specification and implementation, Marr's three levels provide a taxonomy for different approaches to building cognition systems and more importantly imply opportunities to integrate techniques at different levels.

With the fast growing computing power and available data, recent years witnessed a strong momentum in developing algorithmic and implementation level cognition solutions. Figure 1 illustrates main approaches toward understanding human brains and building people-alike intelligent machines organized in Marr's three levels.

Neuroscientists have made substantial progress in understanding the computation model of brains for various cognition tasks [5, 6]. At the algorithmic level, there is no consensus about the multiple potential solutions. The biologically inspired approach adopts the spiking neuron network, which is the computing chassis of brains, to perform cognition computing. The resultant implementation can be built with digital or analog circuits [7-9]. However, the major problem of this approach is that the respective neuroscience is not fully understood yet and we often cannot verify if the behaviors of spiking neuron networks really lead to high-level cognition. Another computation model is the fast rising deep neural networks (DNNs). Although DNNs perform exceptionally well on object recognition tasks, their dependency on data-intensive supervised learning makes them less amenable for complex high-level cognition tasks.

On the other hand, using Bayesian statistics based learning and inference to interpret cognition process and develop corresponding models has received significant successes in the past a few decades [10]. It is suggested that human perception processes following Bayes' rule and the brain is a Bayesian computing platform [11-13]. Researchers have built Bayesian models to account for a wide range of cognition behaviors including object perception [14], word learning [15], working memory [16], and sensorimotor integration [17]. Bayesian models also deliver strong performance in learning and reasoning. For instance, Lake et al. [18] realized a Bayesian model that could learn from sparse data and successfully recognize, write and create texts that are indistinguishable from those written by human. In other words, the model actually passes the Turing Test on this specific problem.

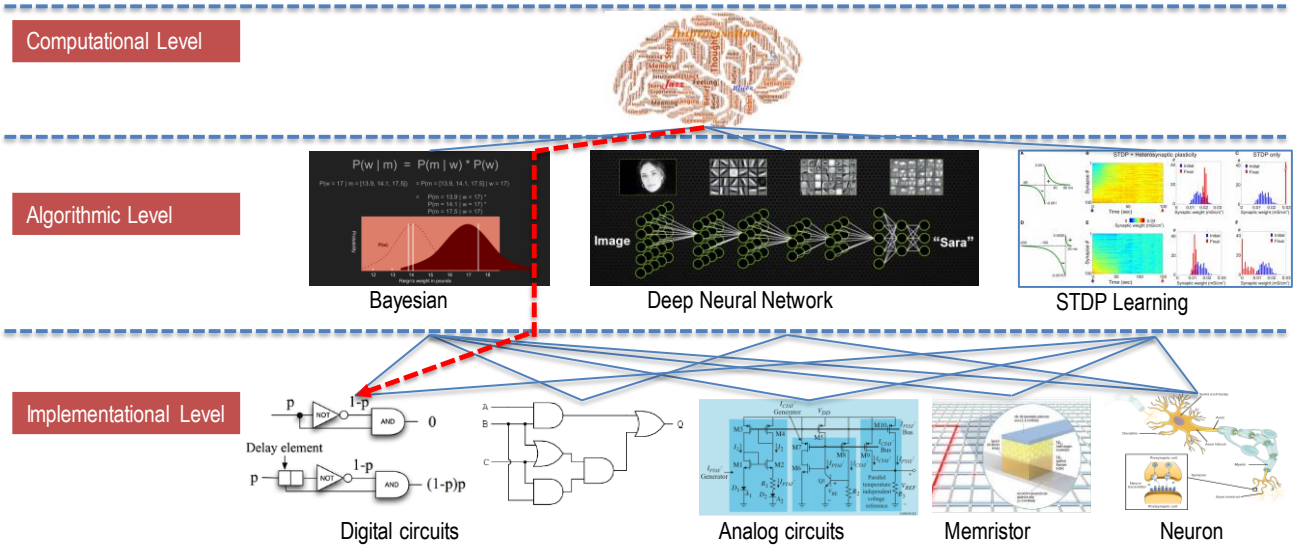


Figure 1. Approaches for cognition computing. The solid line indicates the above level can be realized by the respective lower level approach. The dotted line indicates the approaches taken by this work.

Although the Bayesian based cognition offers a highly appealing direction, a series of challenges have be resolved before we can build machines with human-level cognition behaviors implemented as Bayesian computations. First, the building blocks of Bayesian cognition are various probabilistic computations such as random number generation and sampling probabilistic distributions. Such computations are not natively support by current computers. Second, the core of Bayesian cognition is the computation of posterior distribution, which may not have a closed-form expression, under observed data. The posterior. The invention of the Markov chain Monte Carlo (MCMC) enables us to approximate arbitrary posterior distributions, but it is likely to be expensive in terms of time for convergence [19]. Third, Bayesian cognition depends on special memory models (e.g. exemplar model [20]), which do not have a direct equivalent in current computers.

As the first step towards a systematic effort to develop machines with high-level cognition capabilities, we propose a stochastic computing platform supporting efficient Bayesian probabilistic computations. ABC integrates three different types of resources into a single architecture: a stochastic computing engine for random sampling, a hardware based parallel Markov Chain Monte Carlo for estimating posterior distributions, and an exemplar memory model for emulating human memory behaviors. ABC is highly efficient in Bayesian computations and improves the performance of Bayesian network learning by a factor of 10. Experiments shows that ABC is able to effectively perform a range of cognition behavior at various levels. To demonstrate the capability of ABC for high-level cognition tasks, we show how to use ABC to perform visual category learning, which is a basic cognition capability of human beings [21]. By designing a Bayesian cognition algorithm and mapping to ABC, we are able to attain an accuracy of 62.4%.

The remaining paper is organized as follows. Section 2 explains the background related to this work. In Section 3, we introduce the overall architecture of proposed Bayesian cognition accelerator. Section 4 covers the design details and Section 5 reports the performance evaluation results. In Section 6, we elaborate a cognition application, visual category learning, which is implemented with ABC. Sections 7 concludes the paper.

2. Background

In this section, we first review existing approaches for cognition computing and then discuss the common computer patterns of Bayesian computing. A brief survey of Bayesian computing hardware is given in Section 2.3.

2.1 Approaches to Cognition Computing

The main approaches to cognition computing are shown in Figure 1 as organized according to Marr's three levels. Human cognition has its implementation level realized as a network of billions of interconnected neurons, which are the basic unit of cognition computing. The neurons communicate with each other through spikes with the underlying encoding not fully understood yet [22]. The spiking neural network of brain can be trained with many mechanisms among which the spike-timing-dependent plasticity (STDP) algorithm is probably the best understood one [23]. To mimic the behaviors of brain network, researchers proposed various approaches to implement artificial neurons and interconnect with digital circuit (e.g.

[7]) and analog circuit (e.g. [8][24]) as well as novel devices like mersisters (e.g. [9][25]. Such "silicon neurons" are trained with STDP variants. Although these neuromorphic computing solutions perform well on certain classification problems [7], the major problem of them is the hardness in verifying whether the neural level dynamics does lead to high-level cognitions.

The increasingly popular deep neural networks (DNNs) provide another algorithmic level solution for cognition. On lower level cognition tasks such as image classification and detection, DNNs enable unprecedented performance [26]. Empirical results proved that DNNs outperformed both the human beings [27] and STDP on a spiking neural network implemented in digital hardware [28]. On the other hand, DNNs seem to be more amenable to cognition behaviors at lower levels (e.g. image recognition). To the best of the authors' knowledge, there is no successful applications of DNNs in higher level cognition tasks like concept learning, induction, and inferring physical laws, which are fundamental skills of every child. Another problem of DNNs is that they require a huge amount of labeled data for training purpose. By contrast, people can learn a significant amount of information from sparse data [29]. For instance, kids are able to learn categories from a few positive examples.

The Bayesian cognition theory constitutes another class of algorithmic level models [30]. Besides being able to explaining a wide range of high-level cognition behaviors, Bayesian cognition offers a few essential advantages including naturally handling uncertainties [10], learning from sparse data [18], and being capable of creative behavior [31]. The Bayesian probabilistic computation is intuitively similar to brain's cognition processes. Given a set of observations and the prior knowledge of the world, a manually constructed or learned Bayesian model is used to derive the posterior probabilistic distribution of the hypotheses generating the observation data. One well-known example is the problem of color constancy, i.e. determining the color of objects, for observations under different environmental results. The spectrum of light reflected from an object's surface into the observer's eye is a product of 1) the surface's color spectrum, and 2) the spectrum of the light illuminating the scene. The brain will infer the 1) by considering the received spectrum and prior relating to 2). Although the sensed spectrum varies almost every time, the human vision demonstrates approximate color constancy for a given object [32].

2.2 Bayesian Computations

The Bayesian cognition models are built on various computing patterns, which all center on Bayes' rule. Known as a method to calculate the posterior distribution of given random variables, Bayes' rule indicates that statistical inference can be made as follows:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \quad (1)$$

where H is the hypothesis space that containing all the hypotheses potentially leading to the observation and D represents the observed data from the environment. Given a priori probability $P(H)$ and likelihood $P(D|H)$, $P(H|D)$ is the posterior distribution that we what to know when receiving D. When the hypothesis space is discrete, we can infer the probability of one hypothesis according to Bayes' rule:

$$P(h_i|D) = \frac{P(D|h_i)P(h_i)}{P(D)} = \frac{P(D|h_i)P(h_i)}{\sum_{h_j \in H} P(d|h_j)P(h_j)} \quad (2)$$

The above procedure, often dubbed as Bayesian inference, is able to predict both human reasoning [33] and neural processes [34]. We can see from Equation (2) that the key point of Bayesian inference is calculating $P(D|h_i)P(h_i)$, which can be split into two major steps.

The first step is to establish a Bayesian probabilistic model explaining the problem with proper likelihood and priori distributions. We then need to figure out all the variables that have an impact on inferring and organize them as a joint distribution in a given form. For example, in order to learn and write words as humans, Lake et al. [18] built a joint distribution capturing various factors such as stroke features, stroke spatial relations, and token control points to depict a handwritten character model. Note that such probabilistic models can be readily expressed as a probabilistic graphic model exemplified by Bayesian networks. The structure of this model can be either manually defined or learned from data. Nevertheless, learning Bayesian network is an NP-hard problem. The time complexity can be intractable even on relatively small problems with a few tens of random variables [35].

The second step is to calculate the posterior distribution of interest. For empirical cases, it is often impossible to derive a close-form analytical solution. Under such a circumstance, the Markov chain Monte Carlo (MCMC) method can be used to approximate the target distribution. MCMC can be performed with many algorithms like Metropolis-Hastings algorithm, Gibbs algorithm, and importance sampling [36]. However, the original formulation of MCMC is a sequential process due to the characteristic of Markov chain. Brockwell [37] attempted to create parallelization from MH algorithm by pre-fetching. Wang et al. [38] partitioned the data into subsets for parallelization extraction. Both methods can only achieve limited performance improvement. Calderhead [39] provides an efficient way to convert the sequential process into a parallel one by introducing an additional level of finite-state Markov chain. Despite the above progresses in algorithms, MCMC computations are not well supported by current processors for the extensive use of random number and frequent data synchronization of different threads.

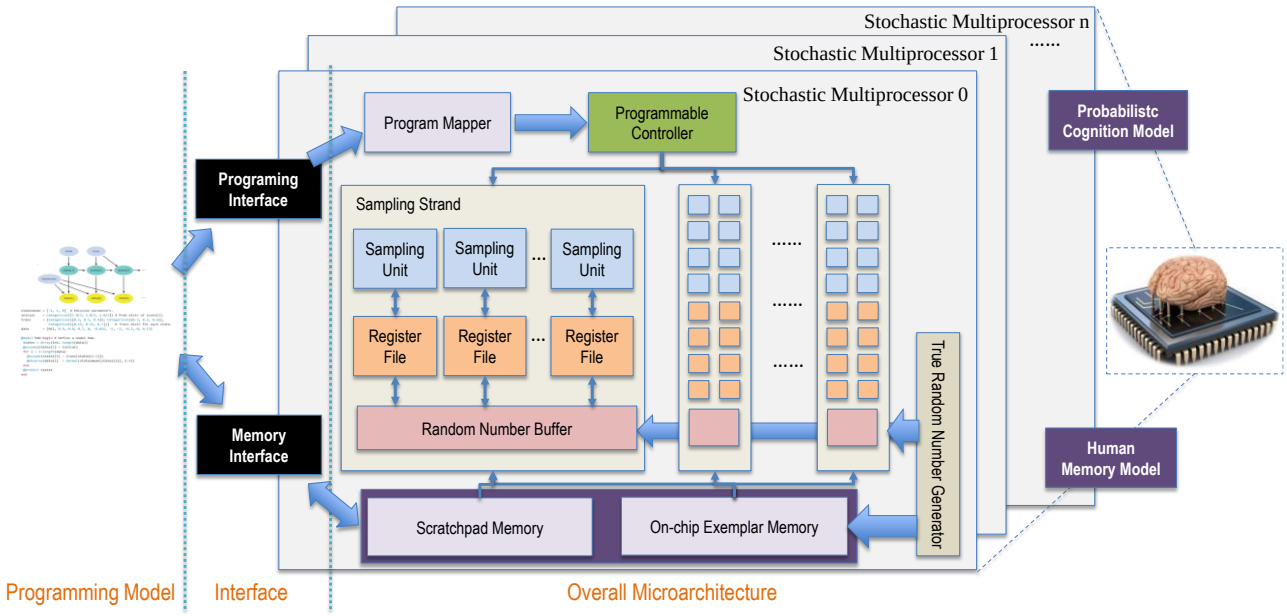


Figure 2. The overall architecture of ABC

Besides the computation models, a memory model is also required by most, if not all, cognition tasks. Cognition memory also attracts a large body of research and a comprehensive review is out of the scope of this paper. Interesting readers please refer to Squire and Wixted's review [40] as a starting point. In this work, we use the exemplar model, which is a cognitive memory model explaining how people store concepts with a few observed instances [41]. When making judgment about a concept, we actually judge the similarity between the stored instances and the current observations [42]. Exemplar models have found successful applications in image recognition [42-43]. Shi et al. [44] identified the underlying relation between exemplar model and Bayesian inference [20]. It is shown that the exemplar model can be formulated as sampling based Bayesian inference and the respective models can closely match the learning performance of human subjects.

2.3 Hardware Accelerator for Bayesian computing

Bayesian computing heavily uses random sampling and probabilistic inference. Such computing patterns are not natively supported by current general-purpose computing platforms like CPU and GPU. To address the problems, researchers proposed dedicated Bayesian computing hardware solutions. Asadi et al. [45] developed hardware and software techniques to support MCMC based Bayesian network learning. The performance is limited because the MCMC algorithm they adopted lacks parallelism. Lin et al. [46] implemented a Bayesian network learning algorithm on FPGA, but it can only work as an accelerator for learning Bayesian networks. Mansinghka et al. [47] proposed the concept of stochastic logic circuits and its respective implementation on digital circuits. The idea is to use a random number generator as the source of entropy, which is then transformed into desired distributions with proper organization of digital logics.

The above results are far from sufficient to perform cognition tasks involving both learning and inferring processes. First of all, the lack of support for parallel MCMC hinders the efficient usage of silicon estate. In fact, it is futile to have a large degree of integration as long as we can only perform sequential MCMC. Second, memory models have not been considered although they are essential for almost any cognition tasks. Third, the brain by its nature is a multi-task processing engine in which multiple explicit and implicit cognition tasks are running concurrently. We will have to perform an overhaul on the computer architecture to sustain multi-tasking of probabilistic cognition tasks.

3. ABC Architecture

In this section we introduce the overall architecture including both programming model and microarchitecture. The overall microarchitecture is illustrated in Figure 2.

3.1 Programming Model and Interface

ABC mimics people to learn and think by manipulating probabilistic computation and memory models. The probabilistic inference processes are captured as Bayesian programs, which is one form of probabilistic programs [48]. In this work, we use a simplified version of the Stan probabilistic programming language [49].

While a full-fledged compilation framework is being developed, we currently use a simple translator to convert an input probabilistic program into a function based internal

representation, which can be processed by an on-chip scheduler. The probabilistic computation is conformed into an instruction set similar to NVIDIA's PTX with a toolchain developed for GPU microarchitecture simulation [50].

3.2 Overall Microarchitecture

The ABC microarchitecture is organized as multiple stochastic multiprocessors to support simultaneous running of multiple cognition tasks. The stochastic multiprocessors can be allocated to execute a single cognition task with one or more sampling strands inside. A sampling strand is equipped with a set of 8 sampling units. Note that these sampling units can be invoked in a SIMD manner. As will be elaborated in Sub-Sections 4.1 and 4.2, a sampling unit is composed of a random number buffer and stochastic logic circuit with the purpose of drawing random samples of a given set common distributions (e.g. uniform, binomial, Poisson, et al.). Random numbers are supplied by a commercial IP core for true random number generator [51]. A programmable controller is designed to coordinate the parallel sampling according to the given MCMC program.

A stochastic multiprocessor is also equipped with an on-chip exemplar memory, which is actually a hybrid computing/memory module, which has its computing resource as well as random number access to manipulate exemplar operations (more details can be found in Sub-Section 4.4). Each stochastic multiprocessor also has a scratchpad memory for fast access of general-purpose data.

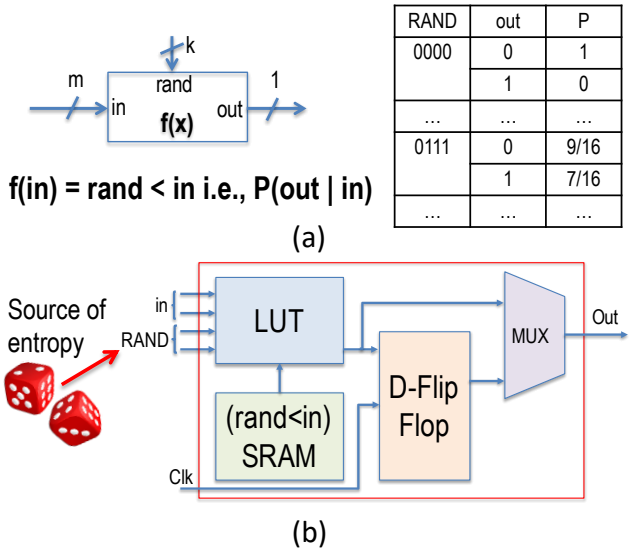
4. DESIGN AND IMPLEMENTATION DETAILS OF ABC

4.1 True Random Number Generator

The Bayesian cognition tasks pose a high demand for random numbers. In fact, at least two random numbers are used for drawing a sample of a random variable. Given complex distributions and a high degree of parallelization of the MCMC process, it is critical to maintain a substantial throughput of random number generation. The exemplar memory is also responsible for some computation tasks requiring random numbers. The true random number generator IP core used in this work consists of several slice generators that each is capable to produce 1Gbps of random bits. The generating rate of one single slice is enough to meet the demand of a stochastic multiprocessors or an exemplar memory. So we implement one slice generator for each stochastic multiprocessors and additional one for the exemplar memory. As random number generator works at a different clock rate from that of other functional blocks on FPGA, we implement all slice generators within a separate clock domain.

4.2 Stochastic Multiprocessors

The stochastic multiprocessor equipped with sampling strands and exemplar memory is the basic unit for cognition computation. As depicted in Figure 2, a sampling strand contains 8 sampling units with each being responsible for a single thread of sampling operation. The sampling units have their own register file and one shared random number buffer. As many as sixteen 32-bit random numbers can be buffered so as to satisfy demand of eight sampling units.



RAND	out	P
0000	0	1
	1	0
…	…	…
0111	0	9/16
	1	7/16
…	…	…

Figure 3.FPGA based random sampling circuits

The workload of a single sampling unit includes both general-purpose basic arithmetic computing and variable sampling. We propose a design shown in Figure 3 for the sampling computing by adopting the idea of combinational stochastic logic [47]. Figure 3(a) illustrate a combinational stochastic logic gate, which receives random streams to generate output streams following a target. We re-design the combinational stochastic logic gate to make it amenable for FPGA implementation. The sampling circuit is shown in Figure 3(b). The LUT of FPGA is configured as a multi-bit random gate. The acceptance rate is determined by comparing the input streams with random numbers generated by the true random number generator to output sampling results.

4.3 MCMC for Posterior Distribution

To better understand how sampling block works, we elaborate how to implement MCMC with a parallel Metropolis-Hastings (MH) algorithm. The MH algorithm ensures that a Markov chain converges to the target distribution by maintaining a proper acceptance rate on the sampling results. Although MH was long considered as a sequential algorithm, Calderhead [39] proposes a novel parallel Metropolis-Hastings algorithm with the introduction of an extra conditional variable. Figure 4 illustrates the sampling flows of both serial and parallel MH algorithms. The parallel algorithm can be explained as 3 steps as follows.

1. First, sample N values of Markov chain based on the present sample in parallel.

2. Second, introduce a new variable *I*, which a uniform distributed integer in the range of 1 to N and calculate its stationary distribution based on the results in step 1.

3. Third, sample *N* values of *I* according to the distribution calculated in step 2. And use the outcome as indexes to assign values in step 1 as the final sampling results.

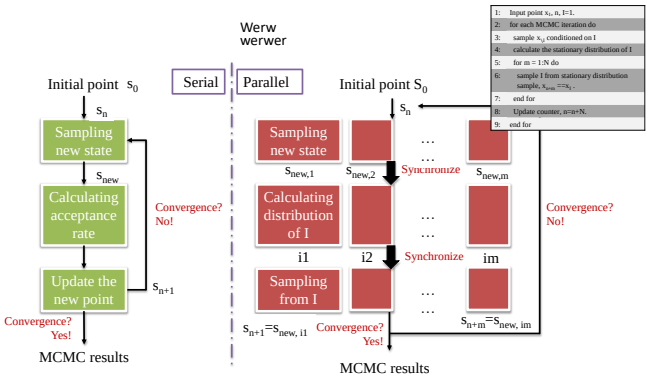


Figure 4. Sampling flow of serial and parallel MCMC

Each step above can be treated as fully independent processes assigned to *N* parallel threads with proper synchronization. To better utilize the hardware resource, we design a sampling unit that is capable to accomplish one step of processing. The sampling unit works in a cycling flow as depicted in Figure 5 and synchronize with other sampling units at the end of each cycle.

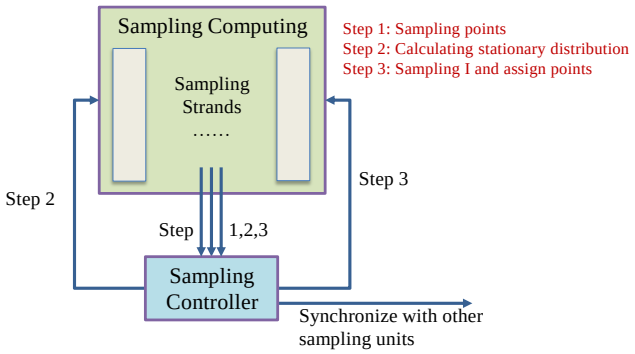


Figure 5. Processing flow of a sampling unit

The computational complexity in the above step 2 is determined by both specific probabilistic model and the paralleled thread number *N*. A fixed setting of *N* is not flexible enough for different tasks and it is impossible for hardware to know the better choice of *N* in advance. As a result, *N* is defined by program and is implement by the programmable controller in Figure 2. To simplify the design of programmable controller, 8 sampling units are grouped into a sampling strand and the controller can choose to use 1, 2, 4 or 8 strands according to the workload.

4.4 Exemplar Memory

In the exemplar model, a concept is memorized as several representative instances. Induction and inference about a concept are in fact made by referring to the instances of that concept. Instances of a concept keep updating as more data received showing the fact that number and selection of instances is stochastic to some extent. In Bayesian computing, an exemplar model is expressed by parameters representing instances and a distance function for each category [43] measuring the similarity between instances.

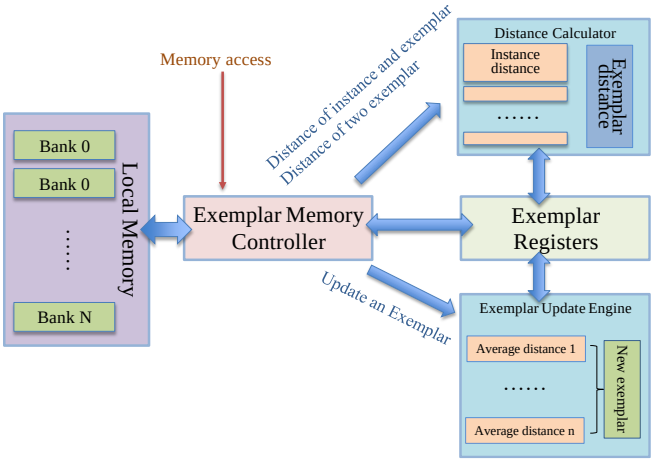


Figure 6. Local exemplar memory

As illustrated in Figure 6, the on-chip exemplar memory consists of 5 major parts as follows.

- The exemplar memory controller is in charge of accessing local memory and scheduling computation resources such as the exemplar update engine and the distance engine. It must be emphasized that the exemplar memory controller is also the memory interface towards sampling unit and so it accepts only memory accesses. Without instructions, the exemplar memory controller determines the operation to be executed by distinguishing the accessed data. For example, upon loading a new exemplar, the exemplar memory controller invokes an access process and calls the distance calculator to return the distance between the stored instance and the new exemplar. In fact, such a feature is quite similar to human cognition behaviors which are not based on commands but are dominated by reflexes to different input information.

- The exemplar register stores all intermediate results including exemplar indexes, calculated distances, memory address, and other I/O data.

- The local memory is partitioned into banks and each bank stores a single exemplar. Each instance in the exemplar is represented as a vector of parameters. A matrix of distance functions and distance results between every pair of instances are also stored. At present, we avoid memory overflow by setting algorithmic constrains. In other words, local memory has an exact upper limit for the number of exemplars,

number of instances for one exemplar, and the dimension of parameters for the instance representation.

- The distance engine calculates the distance of a new input instance with each instances of a targeting exemplar in parallel. Each thread performs matrix multiplication of two instance vector and one distance matrix.
- The exemplar update engine modifies an exemplar when a new instance is classified into a concept. Distance of the new instance from each existing instance is first computed by a set of calculator hardware. Suppose that there are *N* instances in this exemplar including the new one. There are *N* combination to choose *N-1* instances and 1 way to choose *N* instances. The exemplar update engine calculates the average distance of these *N+1* situations and picks one instance with the preference on those with a longer distance to update the exemplar.

With these five parts working together, the local exemplar memory is actually a computational memory. It is capable of storing concept exemplars, returning the distance of a new instance with an existing exemplar, and updating exemplars when new data comes.

5. Evaluation

In this section, we evaluate the implementation and applications of the ABC microarchitecture. An FPGA implementation of ABC with one stochastic multiprocessor is first presented in Section 5.1. Then we evaluate the performance of the implementation for Metropolis-Hastings MCMC. We also shown the performance advantage of using ABC to conduct MCMC based Bayesian network learning. In Section 5.3, we explain how to use ABC to perform visual category learning, which is a fundamental high-level cognition skill for human beings.

5.1 FPGA Implementation of ABC

A prototyping ABC microarchitecture is implemented on a Xilinx VC709 development board. Due to the limit of resources, we only implement two stochastic multiprocessors and an exemplar memory. Table 1 shows the microarchitecture configuration.

Table 1. ABC configuration

FPGA Model	Xlinx VC709
Clock Rate	128MHz
Sampling Strands	32
Random Number Generator Frequency	512MHz
Random Number Buffer	64B
Exemplar Memory Banks	32
Bank Size	500B
Scratchpad Memory	32KB

Working at 128MHZ, ABC is able to distribute sampling tasks to two stochastic multiprocessors with 16 strands involved in each one. The random number buffer has a size of 64B allowing two 32-bit random numbers buffered for each thread. The exemplar memory has a total of 32 banks, which means that up to 32 concepts can be stored on-chip for one stochastic multiprocessor. Note that the on-chip exemplar memory corresponds to the working memory (i.e. short-term memory) in human brains [51]. The capacity of each bank is 500B, which is sufficient for ten 10-parameter instances. Besides, a stochastic multiprocessor has a 32KB scratchpad memory used for conventional memory accesses. The hardware resource utilization for a stochastic multiprocessor is listed in Table 2. When compared with an FPGA based implementation of NVIDIA Pascal GPU [50], a stochastic multiprocessor uses about one half the hardware resources of a stream multiprocessor. We also synthesized ABC with one stochastic multiprocessor targeting a 16nm ASIC process and proves that ABC can be running at 1.6 GHz.

Table 2 Resource utilization of a stochastic multiprocessor

Type	Random number generator	Sampling strand	Exemplar Memory	ABC	Usage
LUT	5436	3768	1163	181536	40.0%
Register	8239	2049	843	197561	22.3%
Mux	513	311	79	14836	6.9%

5.2 MCMC Sampling

We first evaluate the performance of performing Metropolis-Hastings MCMC on the FPGA implementation of ABC. In this evaluation, we choose a Beta distribution, $X \sim Beta(\alpha, \beta)$, as the target posterior distribution to perform MCMC.

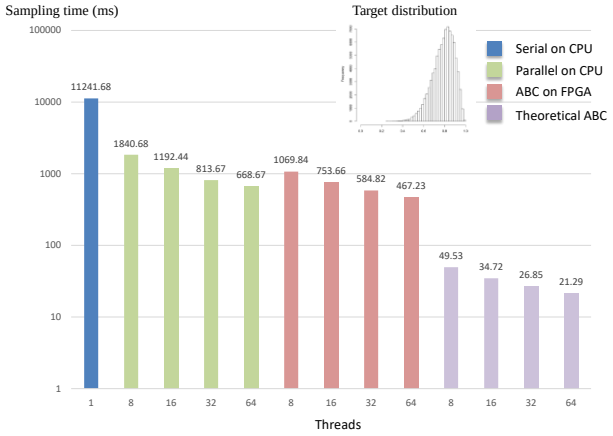


Figure 7. Performance of MCMC sampling

ABC is programed to use 1, 2, 4 or 8 strands to sample as many as 160 thousands points. We test a serial MH algorithm and a parallel MH algorithm [39] implemented with an increasing number of threads on a Core i7 4-core CPU running at 2.7 GHz for comparison. Figure 7 shows the performance comparison of serial algorithm on CPU, parallel algorithm on CPU, ABC implementation on FPGA

and the performance of theoretical ABC. Although the parallel MH algorithm is highly scalable, the performance of the multithreaded implementation improves slower when the number of threads is beyond 16. On the other hand, the ABC based solution scales much better. The 16nm ASIC version of outperforms the serial version by a factor of 528 and the 64-thread version by a factor of 32.3, respectively.

5.3 Bayesian Network Learning

Bayesian network is a probabilistic graphic model to characterize a set of random variables and their conditional dependencies [10]. It has been widely used as the model for various cognition processed [53]. The leaning a Bayesian network, including learning both the network topology and parameters (conditional probabilities), is known to be NP-hard. MCMC based structure learning is a major approach to build Bayesian networks [21].

We compared three implementations of Bayesian network learning, a CPU based sequential version[54], a GPU based parallel version, and two ABC based parallel versions (on FPGA and ASIC, respectively). Both the GPU and ABC implementations use the parallel MH algorithm by Calderhead [39]. The CPU version is tested on a Core i7 CPU, while the GPU version is implemented on a Titan X acceleration card (Pascal GPU). The ABC version is tested on FPGA with 2 strands (a single stochastic processor). Figure 8 shows the performance measures with regard to the size of Bayesian networks. In spite of running at 128MHz with much fewer computing resource, ABC already performs better than a full-fledged GPU. The 16nm ASIC version of ABC outperforms GPU by over 10 times.

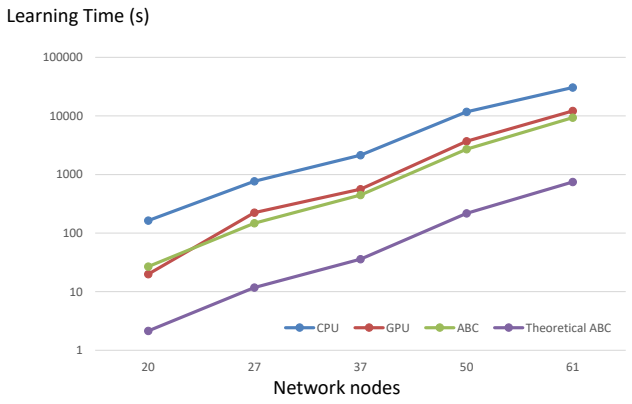


Figure 8. Speed up of Bayesian network learning on ABC

6. Visual Learning on ABC

As the first step to systematically testify the feasibility of using ABC for high-level cognition tasks, we developed a Bayesian based visual category algorithm and its implementation on ABC.

The visual category learning problem [55] can be formulated as follows. Given images from multiple categories as input data, learn the underlying concepts and their structured relation from data. The whole process is actually a non-stopping one. During learning, each image received is classified into a learned concept, while new concepts can be created and inserted into a proper position of the overall category structure.

Note that the visual category learning of human beings is fundamentally different from the image classification problems for deep neural networks. 1) The image classification problem on which DNNs have attained dramatic success [56] is defined on a fixed set of categories specified *a priori*, but human category learning is defined on an open space of categories. 2) DNN requires a significant numbers of labeled images for each category, but human beings are able to learn from sparse data. In fact, kids only need a few examples to learn a concept and many times a single positive example suffices. 3) Human beings infers new category and memorize it when seeing something cannot fit into other categories, but DNNs are unable to classify an object that does not exist in training data. 4) In visual category learning, concepts are organized into a multi-layer tree structure and can be inferred at different level. For example, a Husky dog can be referred to as Husky, dog, and animal under different contexts. Such multi-level structure is essential for people to make generalizations [57].

In recent years, a few researchers explored how to extend the DNN based techniques for visual category learning (e.g., [1][9][58]). However, these results only received limited success due to the inherent difficulty of DNNs to understand the inherent hierarchical structure.

In this work, we propose a visual category learning algorithm based on the exemplar model and Bayesian computing as well as the respective implementation on ABC. With this application as a case study, we show that ABC is capable of performing high-level cognition tasks consisting of a wide range of fundamental cognition behaviors as follows.

- Learning a concept of category from several instances from a category
- Structuring the concepts into a category tree
- Making inference about an instance at multiple levels in a category tree
- Inferring new concept without data-intensive training and adding it to category tree at an exact position
- Learning a Bayesian network from multi-labeled data to perform classification

In the remainder of this section, we first explain how the exemplar model and Bayesian framework work for the visual category learning problem. Then we report the experimental on ABC.

6.1 Exemplar Model of Category Concept

During visual category learning, each category actually corresponds to a concept in a tree structure. Such a hierarchical concept structure is found to be psychologically plausible [57]. Concepts in this framework are represented and memorized with the exemplar model. The basic idea of exemplar model is to characterize a concept with several instances belonging to a category. Thus the key issue of the exemplar model is to find the most representative instances (so far). Given a similarity function *S(a,b)*, measuring the similarity of two instances, we are able to select instances that maximally cover the range of the respective concept, namely $argmin(\sum_{a,b\in C} S(a,b))$.

Suppose there are *n* instances $\{x_1, \ldots, x_n\}$ and *m* concepts $\{c_1, \ldots, c_m\}$. The probability of a new object *x* belonging to a concept c_i is given as follows:

$$P(c_i|x) = \frac{\sum_{x_j \in c_i} S(x,x_j)}{\sum_{j=1}^{n} S(x,x_j)} \tag{3}$$

As the denominator is invariable, we only need to consider the numerator. Similarly, we can measure how much two concept are related by calculating $\sum_{x_i \in c_1, x_j \in c_2} S(x_i, x_j)$.

To implement the exemplar model on ABC, instances are represented by a vector of parameters *V* and the similarity is computed as:

$$S(a,b) = {}^{1}/_{D(a,b)} \tag{4}$$

where *D(a,b)* is the distance of parameter vector of instance *a* and *b*.

6.2 Bayesian Model for Category Inference

Concepts in the proposed Bayesian framework are from a concept space *C*, while each concept c_i is represented by an exemplar e_i from exemplar space *E*. An exemplar e_i contains several instances in_{ij} from an instance space *IN*. Learning a concept is carried out by sampling the following posterior probability:

$$P(E|C, IN) \tag{5}$$

This sampling process attempts to figure out the new exemplar from given concept and instances. The learning task is to consider instances of an exemplar one by one. Here we detail one learning step at the point where a current exemplar e_i of a concept c_i has been confirmed and we need to learn the new $e_{i,new}$ given a new instance in_{new} belonging to c_i. Suppose that there are *n* instances in e_i, an alternative exemplar space *A* is given containing *n+1* kind of *n*-element combination from in_{ij} and in_{new} together with the full set, namely *n+2* choices in *A*. The exemplar result is given by sampling the distribution of *A* as follows:

$$P_l(e_{i,new}|c_i, in_{new}) = P(a_{ij}|c_i, in_{new}) = \frac{\sum_{instance\ pair\ (p,q)\ from\ a_{ij}} D(p,q)/n_{ij}^{\varepsilon}}{\sum_{j=1}^{n+2} \sum_{instance\ pair\ (p,q)\ from\ a_{ij}} D(p,q)/n_{ij}^{\varepsilon}} \tag{6}$$

where n_{ij} represents the total number of instance pairs in a_{ij} and ε defines the weight of chosen number of instances. The greater ε is, the more likely we obtain a fewer number of instances.

We perform the sampling process above in not only concept learning but also every case we confirm a new instance for a concept is inferred. Namely the exemplar e_i of concept c_i is timely updated.

After successfully learning the concept, we make inference about an instance by sampling the distribution $P(C|IN, E)$, Here we sample the corresponding concept for a new instance based on the exemplar model learnt before. The detailed distribution is:

$$P_i(c_i|in_{new}, e_i) = \frac{\frac{De_i^{\delta} * n_i}{\sum_{j=1}^{n_i} D(in_{new}, in_{ij})}}{\sum_{c_i \in C} \frac{De_i^{\delta} * n_i}{\sum_{j=1}^{n_i} D(in_{new}, in_{ij})}} \tag{7}$$

where De_i represents the average distance of exemplar e_i for all instance pairs and δ is the weight for inferring among multi levels of concepts. As a higher level of concept corresponds to a greater average distance, selecting a lower δ make it more likely to infer a instance to lower level concept.

A Bayesian network is built with concepts and their attributes. The concept node is represented by its exemplar. By expanding the equation (7) function with this Bayesian network, it is possible to process a classification application.

The multi-level concept tree is leaned by adding new concept to a known structure. In this framework, we sample the parent node of a given new concept from existing structure described as:

$$P_c(c_i|e_i, c_{new}) = \frac{\frac{De_i^{\theta} * n_i * n_{new}}{\sum_{in_{ij} \in IN_i} \sum_{in_{new,k} \in IN_{new}} D(in_{ij}, in_{new,k})}}{\sum_{c_i \in C} \frac{De_i^{\theta} * n_i * n_{new}}{\sum_{in_{ij} \in IN_i} \sum_{in_{new,k} \in IN_{new}} D(in_{ij}, in_{new,k})}} \tag{8}$$

where θ is a parameter controlling the level discrimination of the concept structure. It is more likely to sample fewer level when setting a high θ.

6.3 Visual Concept Learning Implemented on ABC

In the next sub-section, we will introduce the implementation of the exemplar memory and Bayesian framework discussed in the previous two sub-sections. We use the images from ImageNet [56] as the training data. The images are conceptually organized on the basis of based on WordNet with 21841 concepts. In this work, we are only interested at the 971 concepts of natural object, living things

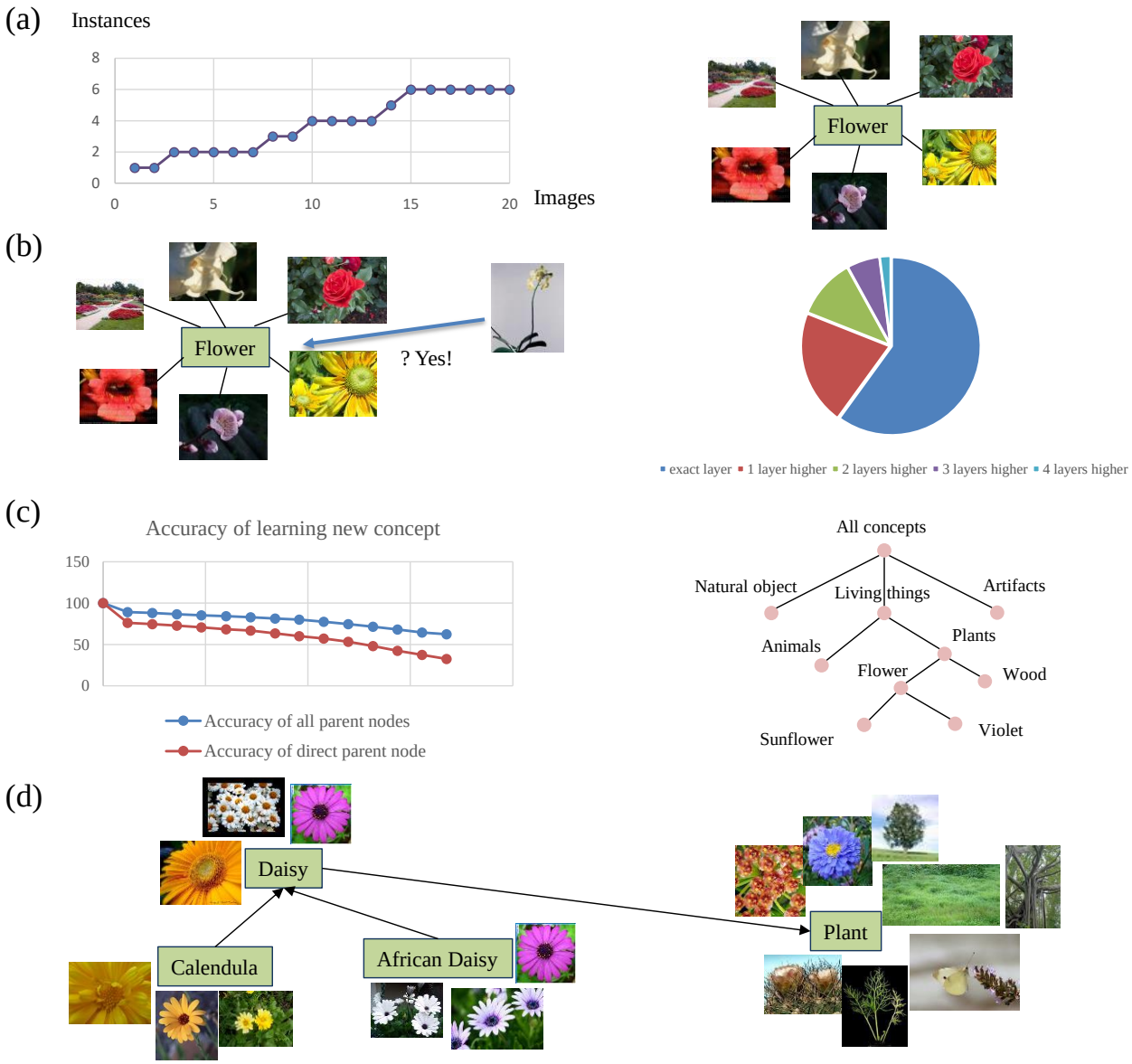


Figure 9. Partial results of visual category learning

(a) The result of learning a single exemplar. Instances number grows from 6 to 20 input images. The right image are the instances of the final exemplar of the concept “flower”. (b) Learning 300 concepts and perform categorization. The fan chart shows the proportion of classifying images to an exact concept and a parent concept in different levels. (c) Learning 671 new concepts on the basis of the 300 learned concepts in (b). The line chart shows the accuracy of putting new concept into proper position in the concept tree. The tree diagram on the right is an example of a learned 5-layer concept tree. (d) Partial learning results for three concepts, daisy, African daisy, and calendula, during the learning process. Among the results, “African daisy” is correctly categorized to be a child of “daisy”, “Calendula” was wrongly put under “daisy”, and “African daisy” is directly placed to be under “plant”. We can see that exemplar of daisy is updated with an instance that also belongs to the category “African daisy”. The exemplar of an upper layer node, plant, contain more instances.

and artifact but leave the learning of more abstracted concepts or categories as future work. ImageNet offers 395 categories of images, which are labeled with four different kinds of attributes, color, pattern, shape and texture. These labeled images enable us to train a simple 5-node Bayesian network for the "infant-level" learning (explained later in this sub-section). We also use the labeled data of ImageNet as ground truth to evaluate our learning results.

The visual category learning on ABC is performed as two sages, corresponding to infant-level learning and kid-level learning, respectively. It is similar to the cognitive development process of children. At the infant stage, children learn category concepts mainly from their parents, who would name the category upon seeing an object (e.g. "this is a dog"). When children grow older, they are able to learn concepts and infer the underlying structure by themselves. The learning models for both stages are coded as probabilistic programs in a simplified version of STAN.

During learning, all images are represented by parameters extracted from several local features [59]. The parameter of ε, δ and θ are set as one in order to lower the computational complexity. All sampling processes utilize two strands of ABC for parallel execution. The maximum number of instances for a single exemplar is set to 10 and it is ensured that the exemplar memory will not overflows.

For the infant-level learning, we prepared 300 categories and their 5-layer structures from ImageNet as the training data. We randomly choose 20 Imagenet images from each category for the purpose of learning and the remaining for test. Such a small number of labeled images suggests a sparse learning process. Figure 9(a) shows an example of learning the exemplar for the concept "flower". The number of instances increases when more images come and finally, ABC sampling these 6 images as the exemplar for the concept flower. Figure 9(b) shows the results of inferring new images of bottom-layer concepts with the total accuracy rate of concept inferring is 70.2%. The accurate rate is evaluated as

$$Accuracy_{infer} = \frac{Count_{direct} + Count_{indirect}}{Count_{all}} \qquad (9)$$

including results that point the image to the corresponding concept and its parent concepts. The proportion of inferring to higher level concept is shown on the right.

For the kid-level learning, ABC samples the remaining 671 concepts out of the 971 categories of ImageNet to derive the structure of categories. We order these 671 concepts in the algorithm to ensure that the direct parent concept node exists in the category structure when sampling a new concept. We randomly pick 3 instances for every concept. When getting a sampling result of direct parent concept of input, all the concepts exemplars along the structure need to be updated. Figure 9(c) shows the accuracy rate of sampling new concept to correct direct parent concept and to correct parent concept, respectively as 32.6% and 62.4%. Besides, it can be inferred from the broken line that the accuracy rate decreases along with the continuing learning procedure as errors accumulated. The final output is a 7-layer category structure. Checking the details of a part of the inference results in Figure 9(d), we can see that the structure resembles to human knowledge.

We also learn a Bayesian network for 100 non-structured concepts to perform classification. We use ABC to re-sample the concept sampling sequence and attain an accuracy rate of 78.5%.

With ABC, it is feasible to learn the 971 concepts and build the structure within 30 minutes. The computations would otherwise be rater expensive (e.g. over 1 day on a Core i7 CPU).

7. Conclusion

In this work, we focus on developing a computing architecture supporting human-alike learning and thinking. Instead of pursuing a biologically plausible architecture, we adopt a Bayesian based approach to seek a computing chassis enabling high-level cognition behaviors. The proposed ABC architecture offers native support for random sampling of probabilistic distributions and parallel MCMC computations. Experiments demonstrates the significant performance advantages of ABC over multi-core CPU and GPU platforms. Moreover, we show how to use ABC to implement a fundamental and essential cognition task, visual category learning algorithm.

References

[1] Li Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol: 28, No. 4, 594 – 611, 2006.

[2] Peter W. Battaglia1, Jessica B. Hamrick, and Joshua B. Tenenbaum, "Simulation as an engine of physical scene understanding," Proceedings of the National Academy of Sciences, vol. 110 no. 45 18327–18332, 2013.

[3] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman, "Building Machines That Learn and Think Like People," Behavioral and Brain Sciences (2016): 1-101.

[4] David Marr and A. Vision, "A computational investigation into the human representation and processing of visual information," WH San Francisco: Freeman and Company 1.2 (1982).

[5] André M.M. Sousa, Kyle A. Meyer, Gabriel Santpere, Forrest O. Gulden, Nenad Sestan, "Evolution of the Human Nervous System Function, Structure, and Development," Cell, Volume 170, Issue 2 July 13, 226–247 , 2017.

[6] Evan F. Risko and Sam J. Gilbert, "Cognitive Offloading," Trends in Cognitive Sciences Volume 20, Issue 9, p641-714, September 2016.

[7] Paul A. Merolla, et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface," Science, Vol. 345, Issue 6197, pp. 668-673, 2014.

[8] Jennifer Hasler and Bo Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," Frontiers in Neuroscience, September 10, 2013.

[9] Bipin Rajendran and Fabien Alibart, "Neuromorphic Computing Based on Emerging Memory Technologies," IEEE Journal on

Emerging and Selected Topics in Circuits and Systems Year, Vol. 6, No. 2, 198 - 211,

[10] Robert A. Jacobs and John K. Kruschke, "Bayesian learning theory applied to human cognition," WIREs Cognitive Science, Vol. 2, No.1, 8-21, 2011.

[11] Matt Jones, and Bradley C. Love, "Bayesian fundamentalism or enlightenment? On the explanatory status and theoretical contributions of Bayesian models of cognition," Behavioral and Brain Sciences 34, no. 4 (2011): 169-188.

[12] David C. Knill, and Alexandre Pouget, "The Bayesian brain: the role of uncertainty in neural coding and computation," TRENDS in Neurosciences27, no. 12 (2004): 712-719.

[13] Matteo Colombo, and Peggy Seriès, "Bayes in the brain—on Bayesian modelling in neuroscience," The British Journal for the Philosophy of Science63, no. 3 (2012): 697-723.

[14] Daniel Kersten, Pascal Mamassian, and Alan Yuille, "Object perception as Bayesian inference," Annu. Rev. Psychol. 55 (2004): 271-304.

[15] Fei Xu, and Joshua B. Tenenbaum, "Word learning as Bayesian inference," Psychological review 114, no. 2 (2007): 245.

[16] Wei Ji Ma, Masud Husain and Paul M Bays, "Changing concepts of working memory," Nature Neuroscience 17, 347–356 (2014).

[17] Konrad P. Körding and Daniel M. Wolpert, "Bayesian integration in sensorimotor learning," Nature, 244-247, 2004.

[18] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum, "Human-level concept learning through probabilistic program induction," Science 350, no. 6266 (2015): 1332-1338.

[19] Radu V. Craiu and Jeffrey S. Rosenthal, "Bayesian Computation Via Markov Chain Monte Carlo," Vol. 1:179-201, 2014.

[20] Douglas L. Medin, and Marguerite M. Schaffer, "Context theory of classification learning," Psychological review 85, no. 3 (1978): 207.

[21] David Madigan, Jeremy York, and Denis Allard, "Bayesian graphical models for discrete data," International Statistical Review/Revue Internationale de Statistique (1995): 215-232.

[22] Fred Rieke, "Exploring the Neural Code," MIT Press, 1999.

[23] Yang Dan, and Mu-ming Poo, "Spike timing-dependent plasticity of neural circuits," Neuron 44, no. 1 (2004): 23-30.

[24] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam et al, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems34, no. 10 (2015): 1537-1557.

[25] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B. Bhadviya, Pinaki Mazumder, and Wei Lu, "Nanoscale memristor device as synapse in neuromorphic systems," Nano letters 10, no. 4 (2010): 1297-1301.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," Nature521.7553 (2015): 436-444.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[28] Zidong Du, Daniel D. Ben-Dayan Rubin, Yunji Chen, Liqiang He, Tianshi Chen, Lei Zhang, Chengyong Wu, and Olivier Temam, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," In Proceedings of the 48th International Symposium on Microarchitecture, pp. 494-507. ACM, 2015.

[29] Amy Perfors, Joshua B. Tenenbaum, Thomas L. Griffiths, and Fei Xu. "A tutorial introduction to Bayesian models of cognitive development," Cognition120, no. 3 (2011): 302-321.

[30] Fei Xu, and Joshua B. Tenenbaum, "Word learning as Bayesian inference," Psychological review 114, no. 2 (2007): 245.

[31] Brenden Lake and Josh Tenenbaum, "Computational Creativity: Generating new objects with a hierarchical Bayesian model," Proceedings of the Annual Meeting of the Cognitive Science Society, 36.

[32] David H. Brainard and William T. Freeman, "Bayesian color constancy," Journal of the Optical Society of America A. 1997 Jul;14(7):1393-411.

[33] Joshua B. Tenenbaum, Charles Kemp, Thomas L. Griffiths, and Noah D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," science 331, no. 6022 (2011): 1279-1285.

[34] Matteo Colombo, and Peggy Seriès, "Bayes in the brain—on Bayesian modelling in neuroscience," The British Journal for the Philosophy of Science63, no. 3 (2012): 697-723.

[35] Tomi Silander, and Petri Myllymaki, "A simple approach for finding the globally optimal Bayesian network structure," arXiv preprint arXiv:1206.6875(2012).

[36] W. Keith Hastings, "Monte Carlo sampling methods using Markov chains and their applications," Biometrika 57, no. 1 (1970): 97-109.

[37] Anthony E. Brockwell, "Parallel Markov chain Monte Carlo simulation by pre-fetching," Journal of Computational and Graphical Statistics 15, no. 1 (2006): 246-261.

[38] Xiangyu Wang, Fangjian Guo, Katherine A. Heller, and David B. Dunson, "Parallelizing MCMC with random partition trees," In Advances in Neural Information Processing Systems, pp. 451-459. 2015.

[39] Ben Calderhead, "A general construction for parallelizing Metropolis−Hastings algorithms." Proceedings of the National Academy of Sciences 111, no. 49 (2014): 17408-17413.

[40] Larry R. Squire and John T. Wixted, "The Cognitive Neuroscience of Human Memory Since H.M.," Annual Review of Neuroscience Vol. 34:259-288, 2011.

[41] Michael L. Mack, Alison R. Preston, and Bradley C. Love, "Decoding the brain's algorithm for categorization from its neural implementation," Current Biology 23, no. 20 (2013): 2023-2027.

[42] Tomasz Malisiewicz, and Alexei A. Efros, "Recognition by association via learning per-exemplar distances," In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1-8. IEEE, 2008.

[43] Ondrej Chum, and Andrew Zisserman, "An exemplar model for learning object classes," In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pp. 1-8. IEEE, 2007.

[44] Lei Shi, Thomas L. Griffiths, Naomi H. Feldman, and Adam N. Sanborn, "Exemplar models as a mechanism for performing Bayesian inference," Psychonomic bulletin & review 17, no. 4 (2010): 443-464.

[45] Narges Bani Asadi, Teresa H. Meng, and Wing H. Wong, "Reconfigurable computing for learning Bayesian networks," In Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays, pp. 203-211. ACM, 2008.

[46] Mingjie Lin, Ilia Lebedev, and John Wawrzynek, "High-throughput bayesian computing machine with reconfigurable hardware," In Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays, pp. 73-82. ACM, 2010.

[47] Vikash K. Mansinghka, Eric M. Jonas, and Joshua B. Tenenbaum, "Stochastic digital circuits for probabilistic inference," Massachussets Institute of Technology, Technical Report MITCSAIL-TR 2069 (2008).

[48] Kamel Mekhnacha, "Bayesian Programming," Chapman and Hall/CRC 2013.

[49] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Michael A. Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell, "Stan: A probabilistic programming language," Journal of Statistical Software 20 (2016): 1-37.

[50] Kuan Fang, Yufei Ni, Jiayuan He, Zonghui Li, Shuai Mu, and Yangdong Deng, "FastLanes: An FPGA accelerated GPU microarchitecture simulator," In Computer Design (ICCD), 2013 IEEE 31st International Conference on, pp. 241-248. IEEE, 2013.

[51] Berk Sunar, William J. Martin, and Douglas R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," IEEE Transactions on computers 56, no. 1 (2007).

[52] Mark D'Esposito, and Bradley R. Postle, "The Cognitive Neuroscience of Working Memory," Annual Review of Psychology, Vol. 66:115-142, 2015.

[53] Thomas L. Griffiths, Charles Kemp, Joshua B. Tenenbaum, "Bayesian Models of Cognition," Ch.3, The Cambridge Handbook of Computational Psychology, 59-100, 2012.

[54] Hongzhi Ma, "Data Parallel Algorithms for Bayesian Network Structural Leaning," Master Thesis, Tsinghua University, 2017.

[55] Jennifer J. Richler, and Thomas J. Palmeri, "Visual category learning," Wiley Interdisciplinary Reviews: Cognitive Science 5, no. 1 (2014): 75-94.

[56] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al, "Imagenet large scale visual recognition challenge," International Journal of Computer Vision 115, no. 3 (2015): 211-252.

[57] Charles Kemp and Joshua B. Tenenbaum, "The discovery of structural form," Proceedings of the National Academy of Sciences of the United States of America, vol. 105 no. 31, 10687–10692, 2008.

[58] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu, "Learning fine-grained image similarity with deep ranking," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1386-1393. 2014.

[59] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio, "Large scale online learning of image similarity through ranking," Journal of Machine Learning Research 11, no. Mar (2010): 1109-1135.