Three Theorems: Brain-Like Networks Logically Reason and Optimally Generalize

Juyang Weng *Fellow, IEEE*

***Abstract*— Finite Automata (FA) is a base net for many sophisticated probability-based systems of artificial intelligence. However, an FA processes symbols, instead of images that the brain senses and produces (e.g., sensory images and motor images). Of course, many recurrent artificial neural networks process images. However, their non-calibrated internal states prevent generalization, let alone the feasibility of immediate and error-free learning. I wish to report a general-purpose Developmental Program (DP) for a new type of, brain-anatomy inspired, networks — Developmental Networks (DNs). The new theoretical results here are summarized by three theorems. (1) From any complex FA that demonstrates human knowledge through its sequence of the symbolic inputs-outputs, the DP incrementally develops a corresponding DN through the image codes of the symbolic inputs-outputs of the FA. The DN learning from the FA is incremental, immediate and error-free. (2) After learning the FA, if the DN freezes its learning but runs, it generalizes optimally for infinitely many image inputs and actions based on the embedded inner-product distance, state equivalence, and the principle of maximum likelihood. (3) After learning the FA, if the DN continues to learn and run, it "thinks" optimally in the sense of maximum likelihood based on its past experience.**

I. Introduction

Models for intelligent agents fall into two large categories, symbolic and emergent.

A. Symbolic networks

Given a task, a human designer in Artificial Intelligence (AI) [11], [6] or Cognitive Science [1], [24] handcrafts a Symbolic Network (SN), using handpicked task-specific concepts as symbols, as illustrated in Fig. 1(a). The "common denominator" network underlying many such SNs is the Finite Automaton (FA) whose probabilistic extensions include the Hidden Markov Model (HMM), the Partially Observable Markov Decision Processes (POMDP) and the Bayesian Nets (also called belief nets, semantic nets, and graphical models). Fig. 2 gives an FA that simulates a simple cognitive and behavioral animal.

Such an FA is powerful by recursively directing many different sensory sequences (e.g., "kitten" and "young cat") into the same equivalent state (e.g., z_3) and its future processing is always based on such an equivalence. For example, state z_4 means that the last meaning of all input subsequences that end at z_4 is "kitten looks" or equivalent. However, the resulting machine does not truly understand the symbolic concepts and is unable to learn new concepts beyond possible re-combinations of handpicked symbols.

Juyang Weng is with the Department of Computer Science and Engineering, Cognitive Science Program, and Neuroscience Program, Michigan State University, East Lansing, MI, 48824 USA (email: weng@cse.msu.edu).

The author would like to thank Z. Ji, M. Luciw, K. Miyan and other members of the Embodied Intelligence Laboratory at Michigan State University; Q. Zhang and other members of the Embodied Intelligence Laboratory at Fudan University whose work have provided experimental supports for the theory presented here.

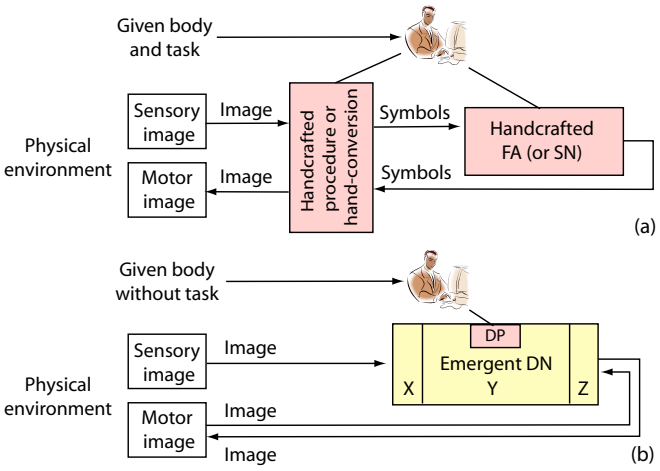

Fig. 1. Comparison between a symbolic FA (or SN) and an emergent DN. (a) Given a task, an FA (or SN), symbolic, handcrafted by the human programmer using a static symbol set. (b) A DN, which incrementally learns the FA but takes sensory images directly and produces motor images directly. Without given any task, a human designs the general-purpose Developmental Program (DP) which resides in the DN as a functional equivalent of the "genome" that regulates the development — fully autonomous inside the DN.

B. Emergent networks

The term "connectionist" has been misleading, diverting attention to only network styles of computation that do not address how the internal representations emerge without human programmer's knowledge about tasks. Furthermore, the term "connectionist" has *not* been very effective to distinguish (emergent) brain-like networks from SNs. For example, Jordan & Bishop [9] used neural networks to name SNs, and Tenenbaum et al. [25] used SNs to model the mind.

Definition 1 (Emergent representation): An emergent representation emerges autonomously from system's interactions with the *external* world (outside the brain or network) and the *internal world* via its sensors and its effectors without using the handcrafted (or gene-specified) content or the handcrafted boundaries for concepts about the extra-body environments.

Feed-forward [22], [21] and recurrent [7], [33] networks, use images (numeric patterns) as representations. Recurrent networks can run continuously to take into account temporal

978-1-4244-9636-5/11/$26.00 ©2011 IEEE

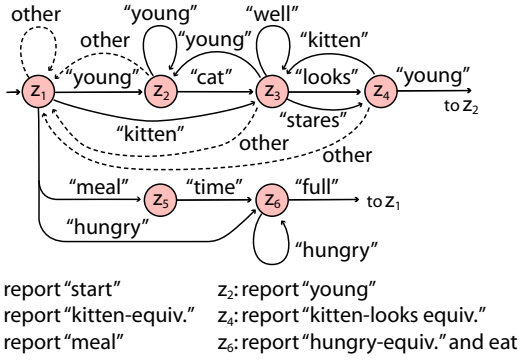

Fig. 2. An FA simulates an animal. Each circle indicates a context state. The system starts from state z_1. Supposing the system is at state q and receives a symbol σ and the next state should be q', the diagram has an arrow denoted as $q \xrightarrow{\sigma} q'$. A label "other" means any symbol other than those marked from the out-going state. Each state corresponds to a set of actions, indicated below the FA. The "other" transitions from the lower part are omitted for brevity.

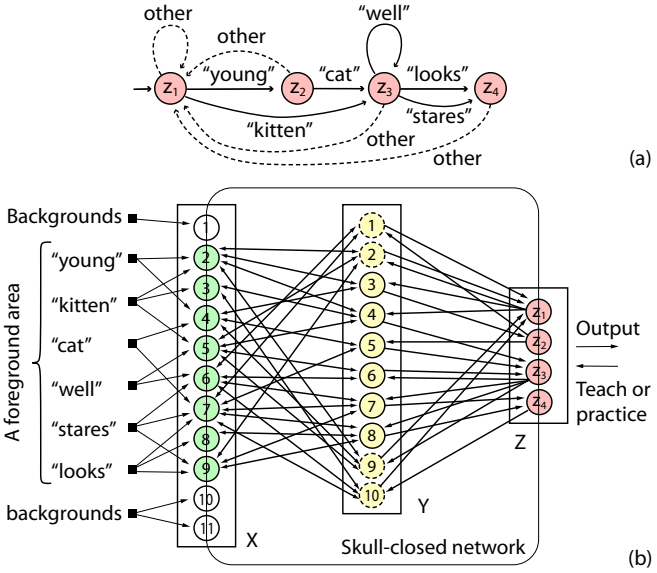

Fig. 3. Conceptual correspondence between an Finite Automaton (FA) with the corresponding DN. (a) An FA, handcrafted and static. (b) A corresponding DN that simulates the FA. It was taught to produce the same input-out relations as the FA in (a). A symbol (e.g., z_2) in (a) corresponds to an image (e.g., $(z_1, z_2, ..., z_4) = (0, 1, 0, 0)$) in (b).

information. The network representations are emergent in the sense that the internal representations, such as network connection patterns, multiple synaptic weights, and neuronal responses, emerge automatically through the interactions between the learner system and its environment. However, it is unclear how a recurrent network can model a brain.

Vincent Müller [19] stated: "How does physics give rise to meaning? We do not even know how to start on the hard problem." This question is indeed challenging to answer since the internal representations inside the brain skull do not permit handcrafting. They emerge from a single cell (zygote) through experience, regulated by the Developmental Program (DP) — the genome program in the nucleus of every cell. As illustrated in Fig. 1(b), an artificial DP is handcrafted by a human, to short cut extremely expensive evolution.

C. Symbolic networks vs. emergent networks

Marvin Minsky 1991 [17] and others argued that symbolic models are logical and clean, while connectionist (he meant emergent) models are analogical and scruffy. The logic capabilities of emergent networks are still unclear, categorically.

Neuroanatomical studies, surveyed by so far probably the most extensive and detailed review by Felleman & Van Essen [4] reported that in the brain the motor areas feed its signals back to the earlier sensory areas and, furthermore, in general, almost every area in the brain feeds its signals to multiple earlier areas.

Computationally, feed-forward connections serve to feed sensory features [20], [23] to motor area for generating behaviors. It has been reported that feed-backward connections can serve as class supervision [7], attention [2], [3], and storage of time information [33]. Foreseeably, there are many other functions to which we can attribute feed-backward connections to. Gallistel reviewed [5]: "This problem-specific structure, they argue, is what makes learning possible." "Noam Chomsky ... , Rochel Gelman, Elizabeth Spelke, Susan Carey, and Renee Baillargeon have extended this argument."

However, the theory introduced hear seems to show that the brain does not have to work in such a piece-meal way or a problem specific way if we understand the brain connections using the automata theory developed for modeling computer-like reasoning. The Developmental Network (DN) here provides an example — a problem-specific (or task-specific) structure is unnecessary for DN learning.

The remainder of this paper is organized as follows. Section II briefly introduces the Finite Automata and an extension to agent FA. The DN framework is outlined in Section III. The major meanings of the three theorems were explained in Section IV. The published experimental results that support the framework here were briefly summarized in Section VI. Section VII provides concluding remarks.

II. Symbolic Networks

The brain's spatial network seems to deal with general temporal context without any explicit component dedicated to time as argued by [16], [10], but its mechanisms are still largely elusive.

A. Intuitive introduction to FA

FA is amenable to understanding the brain's way of temporal processing. An FA example is shown in Fig. 3(a). At each time instance, the FA is at a state. At the beginning, our example is at state z_1. Each time, it receives a label as input (e.g., "young"). Depending on its current state and the next input, it transits to another state. For example, if it is at z_1 and receives label "young", it transits to "z_2",

meaning "I got 'young'." All other inputs from z_1 leads back to z_1 meaning "start over". The states have the following meanings: z_1: start; z_2: "young"; z_3: "kitten" or equivalent; z_4: "kitten looks" or equivalent. An FA can reason. For example, our FA example treats "young cat" and "kitten" the same in its state output.

B. Formal definition of FA and agent FA

Definition 2 (Language acceptor FA): A finite automaton (FA) M is a 5-tuple $M = (Q, \Sigma, q_0, \delta, A)$, where Q is a finite set of symbols called states; Σ is a finite alphabet of input symbols; $q_0 \in Q$ is the initial state; $A \subset Q$ is the set of accepting states; $\delta : Q \times \Sigma \mapsto Q$ is the state transition function.

We extend the definition of the FA to agent FA:

Definition 3 (Agent FA): An agent FA M for a finite symbolic world is a 4-tuple $M = (Q, \Sigma, q_0, \delta)$, where Σ and q_0 are the same as above and Q is a finite set of states, where each state $q \in Q$ is a symbol, corresponding to a set of concepts. The agent runs through discrete times $t = 1, 2, ...$, starting from state $q(t) = q_0$ at $t = 0$. At each time $t - 1$, it reads input $\sigma(t-1) \in \Sigma$ and transits from state $q(t-1)$ to $q(t) = \delta(q(t-1), \sigma(t-1))$, and outputs $q(t)$ at time t, illustrated as $q(t-1) \stackrel{\sigma(t-1)}{\longrightarrow} q(t)$.

The input space is denoted as $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_l\}$. The concept set of each state has two subsets, the cognition set and the action set; but the cognition set can also be considered as actions — report the cognition. Regardless of meanings, the set of states can be denoted as $Q = \{q_1, q_2, ..., q_n\}$. The example of agent FA in Fig. 2 hints that an FA can be very general.

C. Completeness of FA

Let Σ^* denote the set of all possible strings of any finite $n \geq 0$ number of symbols from Σ. All possible input sequences that lead to the same state q are equivalent as far as the FA is concerned. It has been proved that an FA with n states partitions all the strings in Σ^* into n sets. Each set is called equivalence class, consisting of strings that are equivalent. Since these strings are equivalent, any string x in the same set can be used to denote the equivalent class, denoted as $[x]$. Let Λ denote an empty string. Consider Fig. 2. The FA partitions all possible strings into 6 equivalent classes. All the strings in the equivalent class $[\Lambda]$ end in z_1. All strings in the equivalent class ["kitten" "looks"] end in z_4, etc.

The completeness of agent FA can be described as follows. When the number of states is sufficiently large, a properly designed FA can sufficiently characterize the cognition and behaviors of an agent living in the symbolic world of vocabulary Σ.

D. Other types of automata

Furthermore, there are four types of well known automata, FA, Pushdown Automata, Linear Bounded Automata (LBA) and Turing machines.

Automata have been used to model the syntax of a language, which does not give much information about semantics. As argued by linguisticists [26], [8], semantics is primary in language acquisition, understanding and production, while syntax is secondary.

The DN theory below enables the semantics to emerge implicitly in its connection weights in the network. In particular, it treats syntax as part of the emergent semantics. It does not separately treat syntax as the above three types of automata. Therefore, FA is sufficient for our purpose.

E. Symbolic networks: Probabilistic variants

FA has many probabilistic variants (PVs), e.g., HMM, POMDP, and Bayesian Nets. Like FA, each node (or module) of a PV is defined by the handcrafted meaning which determines what data humans feed it during training. A PV can take vector inputs (e.g., images) based on handcrafted features (e.g., Gabor filters). The PV determines a typically better boundary between two ambiguous symbolic nodes (or modules) using probability estimates, e.g., better than the straight nearest neighbor rule. However, this better boundary does not change the symbolic nature of each node (or module). Therefore, FA and all its PVs are all called Symbolic Networks (SNs) here.

F. Power of SN

The major power of SN lies in the fact that it partitions infinitely many input sequences into a finite number of states. Each state lumps infinitely many possible state trajectories (e.g., "kitten" and "young cat") into the same single state (z_3). For example, state z_4 means that the last meaning of all input subsequences that end at z_4 is "kitten looks" or equivalent. Regardless what the previous trajectories were before reaching the current state, as long as they end at the same state now they are treated exactly the same in the future. This enables the SN to generalize (act correctly) for infinitely many state trajectories that it has not been observed. For example, in Fig. 2(a), as long as "kitten" has been taught to reach z_3, "kitten looks", "kitten stares", "kitten well looks" so on all lead to z_4, although these strings have never been observed.

G. Limitations of SN

An SN has the following major limitations:

(1) An SN is static. It does not have emergent representations like those in the brain. Therefore, it cannot think like the brain. For example, it cannot be creative, going beyond a finite number of combinations of these handcrafted static concepts.

(2) An SN is intractable for dealing with many concepts. Suppose that a task involves c concepts (e.g., location, type, scale) and each concept has v values. The number of states is potentially v^c, exponential in c. For $v = 4$ and $c = 22$, $v^c = 4^{20} = 16^{11} > 10^{11} = 100,000,000,000$, larger than the number of neurons in the human brain. Here are 23 examples of concept: object type, horizontal direction, vertical direction, object pose, apparent scale on the retina,

viewing distance, viewing angle, surface texture, surface color, surface reflectance, lighting direction, lighting color, lighting uniformity, material, weight, temperature, deformability, purpose, usage, owner, price, horizontal relationship between two attended objects, vertical relationship between two attended objects. It is intractable for a human to examine that many symbolic states and decide which ones are equivalent and should be merged as a single meta symbolic state. Therefore, a human designs conditions for every meta state without exhaustively checking its validity. This is a new complexity reason why symbolic agents are brittle.

III. Developmental Networks

Weng 2010 [28] stated that a DN can simulate any FA.

A. DN architecture

A basic DN has three areas, the sensory area X, the internal (brain) area Y and the motor area Z. An example of DN is shown in Fig. 3(b). The internal neurons in Y have bi-directional connection with both X and Z.

The DP for DNs is not task-specific as suggested for the brain in [31] (e.g., not concept-specific or problem specific). In contrast to a static FA, the motor area Z of a DN can be directly observed by the environment (e.g., by the teacher) and thus can be calibrated through interactive teaching from the environment. The environmental concepts are learned incrementally through interactions with the environments. For example, in Fig. 3(b), the "young" object makes the pixels 2 and 4 bright and all other green pixels dark. However, such an image from the "young" object is not known during the programming time for the DP.

In principle, the X area can model any sensory modality (e.g., vision, audition, and touch). The motor area Z serves both input and output. When the environment supervises Z, Z is the input to the network. Otherwise, Z gives an output vector to drive effectors (muscles) which act on the real world. The order of areas from low to high is: X, Y, Z. For example, X provides bottom-up input to Y, but Z gives top-down input to Y.

B. DN algorithm

DN is modeled as an area of the brain. It has its area Y as a "bridge" for its two banks, X and Z. If Y is meant for modeling the entire brain, X consists of all receptors and Z consists of all muscles neurons. Y potentially can also model any Brodmann area in the brain. According to many studies in detailed review by Felleman & Van Essen [4], each area Y connects in both ways with many other areas as its two extensive banks.

The most basic function of an area Y seems to be prediction — predict the signals in its two vast banks X and Y through space and time. The prediction applies when part of a bank is not supervised. The prediction also makes its bank less noisy if the bank can generate its own signals (e.g., X).

A secondary function of Y is to develop bias (like or dislike) to the signals in the two banks, through what is known in neuroscience as neuromodulatory systems.

This work focuses on the first, most basic function — prediction for the two banks.

Algorithm 1 (DN): Input areas: X and Z. Output areas: X and Z. The dimension and representation of X and Y areas are hand designed based on the sensors and effectors of the species (or from evolution in biology). Y is the skull-closed (inside the brain), not directly accessible by the outside.

1) At time $t = 0$, for each area A in $\{X, Y, Z\}$, initialize its adaptive part $N = (V, G)$ and the response vector $\mathbf{r}$, where V contains all the synaptic weight vectors and G stores all the neuronal ages. For example, use the generative DN method discussed below.
2) At time $t = 1, 2, ...$, for each A in $\{X, Y, Z\}$ repeat:
   a) Every area A performs mitosis-equivalent if it is needed, using its bottom-up and top-down inputs $\mathbf{b}$ and $\mathbf{t}$, respectively.
   b) Every area A computes its area function f, described below,
   $$(\mathbf{r}', N') = f(\mathbf{b}, \mathbf{t}, N)$$
   where $\mathbf{r}'$ is its response vector.
   c) For every area A in $\{X, Y, Z\}$, A replaces: $N \leftarrow N'$ and $\mathbf{r} \leftarrow \mathbf{r}'$.

In the remaining discussion, we assume that Y models the entire brain. If X is a sensory area, $\mathbf{x} \in X$ is always supervised. The $\mathbf{z} \in Z$ is supervised only when the teacher chooses to. Otherwise, $\mathbf{z}$ gives (predicts) motor output.

Put intuitively, like the brain, the DN repeatedly predicts the output Z for the next moment. When the predicted Z is mismatched, learning proceeds to learn the new information from Z. But, there is no need to check mismatches: learning takes place anyway.

A generative DN (GDN) automatically generates neurons in the Y area. If $(\mathbf{b}, \mathbf{t})$ is observed for the first time ((the pre-action of the top-winner is not 1) by the area Y, Y adds (e.g., equivalent to mitosis and cell death, spine growth and death, and neuronal recruitment) a Y neuron whose synaptic weight vector is $(\mathbf{b}, \mathbf{t})$ with its neuronal age initialized to 1. The idea of adding neurons is similar to ART and Growing Neural Gas but they do not take action as input and are not state-based.

C. DN area function

The area function f which is based on the theory of Lobe Component Analysis (LCA) [30], a model for self-organization by a neural area. Each area A has a weight vector $\mathbf{v} = (\mathbf{v}_b, \mathbf{v}_t)$. Its pre-response vector is:

$$r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t}) = \frac{\mathbf{v}_b}{\|\mathbf{v}_b\|} \cdot \frac{\mathbf{b}}{\|\mathbf{b}\|} + \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|} \cdot \frac{\mathbf{t}}{\|\mathbf{t}\|} = \dot{\mathbf{v}} \cdot \dot{\mathbf{p}} \quad (1)$$

which measures the degree of match between the directions of $\dot{\mathbf{v}} = (\mathbf{v}_b/\|\mathbf{v}_b\|, \mathbf{v}_t/\|\mathbf{v}_t\|)$ and $\dot{\mathbf{p}} = (\dot{\mathbf{b}}, \dot{\mathbf{t}}) = (\mathbf{b}/\|\mathbf{b}\|, \mathbf{t}/\|\mathbf{t}\|)$.

This pre-response is inspired by how each neuron takes many lines of input from bottom-up and top-down sources. It generalizes across contrast (i.e., the length of vectors). It uses inner-product $\dot{\mathbf{v}} \cdot \dot{\mathbf{p}}$ to generalize across many different vectors that are otherwise simply different as with symbols in an FA. The normalization of the first and second terms above is for both the bottom-up source and top-down source to be taken into account, regardless the dimension and magnitude of each source.

To simulate lateral inhibitions (winner-take-all) within each area A, top k winners fire. Considering $k = 1$, the winner neuron j is identified by:

$$j = \arg \max_{1 \le i \le c} r(\mathbf{v}_{bi}, \mathbf{b}, \mathbf{v}_{ti}, \mathbf{t}). \quad (2)$$

The area dynamically scale top-k winners so that the top-k respond with values in $(0, 1]$. For $k = 1$, only the single winner fires with response value $y_j = 1$ and all other neurons in A do not fire. The response value y_j approximates the probability for $\dot{\mathbf{p}}$ to fall into the Voronoi region of its $\dot{\mathbf{v}}_j$ where the "nearness" is $r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t})$.

D. DN learning: Hebbian

All the connections in a DN are learned incrementally based on Hebbian learning — cofiring of the pre-synaptic activity $\dot{\mathbf{p}}$ and the post-synaptic activity y of the firing neuron. If the pre-synaptic end and the post-synaptic end fire together, the synaptic vector of the neuron has a synapse gain $y\dot{p}$. Other non-firing neurons do not modify their memory. When a neuron j fires, its firing age is incremented $n_j \leftarrow n_j + 1$ and then its synapse vector is updated by a Hebbian-like mechanism:

$$\mathbf{v}_j \leftarrow w_1(n_j)\mathbf{v}_j + w_2(n_j) y_j \dot{\mathbf{p}} \quad (3)$$

where $w_2(n_j)$ is the learning rate depending on the firing age (counts) n_j of the neuron j and $w_1(n_j)$ is the retention rate with $w_1(n_j) + w_2(n_j) \equiv 1$. The simplest version of $w_2(n_j)$ is $w_2(n_j) = 1/n_j$ which corresponds to:

$$\mathbf{v}_j^{(i)} = \frac{i-1}{i}\mathbf{v}_j^{(i-1)} + \frac{1}{i} 1 \dot{\mathbf{p}}(t_i), i = 1, 2, ..., n_j, \quad (4)$$

where t_i is the firing time of the post-synaptic neuron j. The above is the recursive way of computing the batch average:

$$\mathbf{v}_j^{(n_j)} = \frac{1}{n_j} \sum_{i=1}^{n_j} \dot{\mathbf{p}}(t_i) \quad (5)$$

The initial condition is as follows. The smallest n_j in Eq. (3) is 1 since $n_j = 0$ after initialization. When $n_j = 1$, $\mathbf{v}_j$ on the right side is used for pre-response competition but does not affect $\mathbf{v}_j$ on the left side since $w_1(1) = 1 - 1 = 0$.

A component in the gain vector $y_j\dot{\mathbf{p}}$ is zero if the corresponding component in $\dot{\mathbf{p}}$ is zero. Each component in $\mathbf{v}_j$ so incrementally computed is the estimated probability for the pre-synaptic neuron to fire under the condition that the post-synaptic neuron fires..

E. GDN area functions

Algorithm 2 (Y area function): This version has $k = 1$ for top-k competition.

1) Every neuron computes pre-response using Eq. (1).
2) Find the winner neuron j using Eq. (2).
3) If the winner pre-response is not 2, generate a Y neuron using the input $\dot{\mathbf{p}}$ as the weight with age 0. The new Y neuron as it is the winner for sure.
4) The winner neuron j increment its age: $n_j \leftarrow n_j + 1$, fire with $y_j = 1$, and updates its synaptic vector, using Eq. (3).
5) All other neurons do not fire, $y_i = 0$, for all $i \neq j$, and do not advance their ages.

Algorithm 3 (Z Area function): This version has $k = 1$ for top-k competition within each concept zone.

1) If the dimension of Y has not been incremented, do:
   a) Every neuron computes pre-response using Eq. (1).
   b) Find the winner neuron j using Eq. (2).

   Otherwise, do the following:
   a) Supervise the pre-response of every neuron to be 1 or 0 as desired.
   b) Add a dimension for the weight vector of every neuron, initialized to be 0, which may be immediately updated below.
2) Each winner or supervised-to-fire neuron j increment its age: $n_j \leftarrow n_j + 1$, fire with $z_j = 1$, and updates its synaptic vector, using Eq. (3).
3) All other neurons do not fire, $z_i = 0$, for all $i \neq j$, and do not advance their ages.

The Y area function and the Z functions are basically the same. Z can be supervised but Y cannot since it is inside the closed "skull". During the simple mode of learning discussed here, neurons responding for backgrounds are suppressed (not attending), so that no neurons learn the background.

IV. Introduction to the Three Theorems

For those who do not want to read the exact formal exposition of the three theorems, the following introduction is sufficient to understand the meaning and importance. For those who do, the following introduction gives the motivation. Due to the space limit, the formal presentation of the three theorems and the full proofs are not presented here but are available at Weng 2011 [29].

A. About Theorem 1

The text version of Theorem 1 is as follows.

The general-purpose DP can incrementally grow a GDN to simulate any given FA on the fly, so that the performance of the DP is immediate and error-free, provided that the Z area of the DN is supervised when the DN observes each new state transition from the FA. The learning for each state transition completes within two network updates. There is no need for a second supervision for the same state transition to reach error-free future performance. The number of Y neurons in the DN is the number of state transitions in the FA. However,

the DN generalizes with 0% action error for infinitely many equivalent input sequences that it has not observed from the FA but are in the brain-mind of the human FA designer.

As a sketch of the proof, Fig. 4 illustrates how the DN simulates each new state transition of FA by creating a new Y neuron that immediately initializes with the image code of the state $q(t-1)$ and the image code of the input $\sigma(t-1)$ through the first network update (see the Y area at time $t-0.5$). During the next network update, the Z area is supervised as the image code of the desired state $q(t)$ and the links from the uniquely firing new Y neuron to the firing Z neurons are created through a Hebbian mechanism. Since the match of the new Y neuron is exact and only one Y neuron fires at any time, the Z output is always error-free if all image codes for Z are known to be binary (spikes).

Let us discuss the meaning of this theorem. Suppose that the FA is collectively acquired by a human society, as a static ontology (common sense knowledge and specialty knowledge). Each input image $\mathbf{x}(t) \in X$ is a view of attended object (e.g., a cat). Then this FA serves as a society intelligence demonstrator representing many human teachers whom an agent meets incrementally from childhood to adulthood. A different FA represents a different career path. Then, a DN can learn such symbolic knowledge of the FA immediately, incrementally, and error-free. This is not what any prior neural network can do. They require many iterative approximations that may lead to local minima.

Furthermore, the DN does not just do rote learning. Each teacher only teaches *piece-meal* knowledge, (e.g., report the same cognition for "young cat" and "kitten"), but the teacher did not indicate how such a piece of knowledge should be transferred to many other equivalent settings (e.g., infinitely many possible sensory sequences which contains "young cat" or "kitten"). The DN transfers such a piece-meal knowledge to future all possible (infinitely many) equivalent input sequences although it has only saw one of such sequences, as we discussed above about the power of FA. Any DN can do such transfers automatically because of the brain-inspired architecture of the DN. Prior neural networks and any conventional databases cannot do that, regardless how much memory they have.

B. About Theorem 2

Suppose that the $\mathbf{x}$ and $\mathbf{z}$ codes for the FA are similar to those from the real physical world. This is important for the skills learned from FA to be useful for the real physical world, as illustrated in Fig. 1(b). The number of symbols in Σ is finite, but the number of images $\mathbf{x} \in X$ (e.g., images on the retina) from the real physical world is unbounded, although finite at any finite age if the video stream is sampled at a fixed sampling rate (e.g., 30Hz).

The following is text version of Theorem 2.

Suppose that the GDN learning is frozen after learning the FA but still run (generating responses) by taking sensory inputs beyond those of the FA, the DN generalizes optimally. It generates the Maximum Likelihood (ML) internal responses and actions based on its experience of learning the FA.

The DGN "lives" in the real world and generalizes optimally, going beyond the FA, as explained in Fig. 1.

C. About Theorem 3

The following is text version of Theorem 3.

Suppose that the GDN has run out of its new Y neurons as soon as it has finished simulating the FA. If it still learns by updating its adaptive part, the DN generalizes ("thinks") optimally by generating the ML internal responses and actions based on the limited network resource, the limited skills from FA, and real-world learning up to the last network update.

Such a unified, general-purpose, task nonspecific, incremental, immediate learning DP can potentially develop a DN to learn a subset of human society's knowledge as an FA, but each DN it develops only learns one such FA in its lifetime. Many DNs learn and live through their own career trajectories to become many different experts who also share the common sense knowledge of the human society. The human programmer of a DP does not need to know the meanings of the states of each possible FA, which are only in the minds of the future human teachers and the learned DNs.

The following gives detail about DN learning FA.

V. DN Simulates FA

First consider the mapping from symbolic sets Σ and Q, to vector spaces X and Z, respectively.

Definition 4 (Symbol-to-vector mapping): A symbol-to-vector mapping m is a mapping $m : \Sigma \mapsto X$. We say that $\sigma \in \Sigma$ and $\mathbf{x} \in X$ are equivalent, denoted as $\sigma \equiv \mathbf{x}$, if $\mathbf{x} = m(\sigma)$.

A binary vector of dimension d is such that all its components are either 0 or 1. It simulates that each neuron, among d neurons, either fires with a spike ($s(t) = 1$) or without ($s(t) = 0$) at each sampled discrete time $t = t_i$. From discrete spikes $s(t) \in \{0, 1\}$, the real valued firing rate at time t can be estimated by $v(t) = \sum_{t-T<t_i\le t} s(t_i)/T$, where T is the temporal size for averaging. A biological neuron can fire at a maximum rate around $v = 120$ spikes per second, producible only under a laboratory environment. If the brain is sampled at frequency $f = 1000$Hz, we consider the unit time length to be $1/f = 1/1000$ second. The timing of each spike is precise up to $1/f$ second at the sampling rate f, not just an estimated firing rate v, which depends on the temporal size T (e.g., $T = 0.5$s). Therefore, a firing-rate neuronal model is less temporally precise than a spiking neuronal model. The latter, which DN adopts, is more precise for fast sensorimotor changes.

Let B_p^d denote the d-dimensional vector space which contains all the binary vectors each of which has *at most* p components to be 1. Let $E_p^d \subset B_p^d$ contains all the binary vectors each of which has *exactly* p components to be 1.

Definition 5 (Binary-p mapping): Let $Q = \{q_i \mid i = 1, 2, ..., n\}$. A symbol-to-vector mapping $m : Q \mapsto B_p^d$ is a binary-p mapping if $p > 0$ and m is one to one: That is, if $\mathbf{z}_i \equiv m(q_i)$, $i = 1, 2, ..., n$, then $q_i \neq q_j$ implies $\mathbf{z}_i \neq \mathbf{z}_j$.

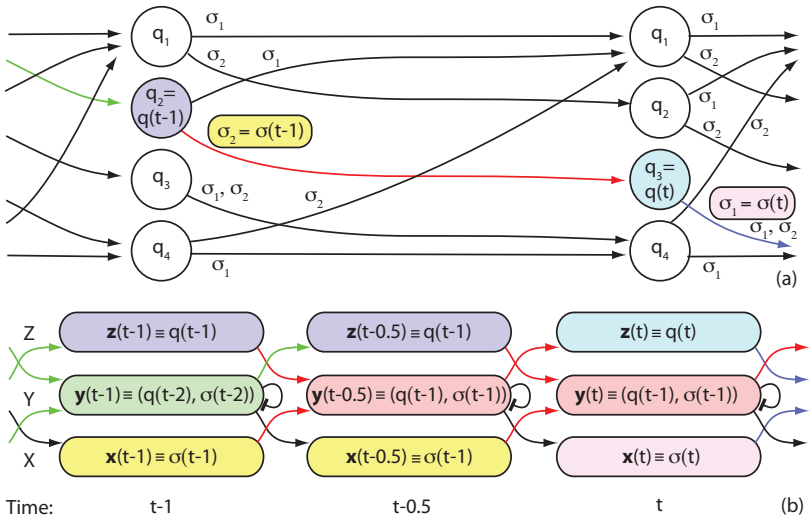

Fig. 4. Model the brain mapping, DN, and SN. In general, the brain performs external mapping $b(t) : X(t-1) \times Z(t-1) \mapsto X(t) \times Z(t)$ on the fly. (a) An NS samples the vector space Z using symbolic set Q and X using Σ, to compute symbolic mapping $Q(t-1) \times \Sigma(t-1) \mapsto Q(t)$. This example has four states $Q = \{q_1, q_2, q_3, q_4\}$, with two input symbols $\Sigma = \{\sigma_1, \sigma_2\}$. Two conditions (q, σ) (e.g., $q = q_2$ and $\sigma = \sigma_2$) identify the active outgoing arrow (e.g., red). $q_3 = \delta(q_2, \sigma_2)$ is the target state pointed to by the (red) arrow. (b) The grounded DN generates the internal brain area Y as a bridge, its bi-directional connections with its two banks X and Z, the inner-product distance, and adaptation, to realize the external brain mapping. It performs at least two network updates during each unit time. To show how the DN learns a SN, the colors between (a) and (b) match. The sign $\equiv$ means "image code for". In (b), the two red paths from $q(t-1)$ and $\sigma(t-1)$ show the condition $(\mathbf{z}(t-1), \mathbf{x}(t-1)) \equiv (q(t-1), \sigma(t-1))$. At $t-0.5$, they link to $\mathbf{y}(t-0.5)$ as internal representation, corresponding to the identification of the outgoing arrow (red) in (a) but a DN does not have any internal representation. At time t, $\mathbf{z}(t) \equiv q(t) = \delta(q(t-1), \sigma(t-1))$ predicts the action. But the DN uses internal $\mathbf{y}(t-0.5)$ to predict both state $\mathbf{z}(t)$ and input $\mathbf{x}(t)$. The same color between two neighboring horizontal boxes in (b) shows the retention of (q, σ) image in (a) within each unit time, but the retention should be replaced by temporal sampling in general. The black arrows in (b) are for predicting X. Each arrow link in (b) represents many connections. When it is shown by a non-black color, the color indicates the corresponding transition in (a). Each arrow link represents excitatory connections. Each bar link is inhibitory, representing top-k competition among Y neurons.

The larger the p the more symbols the space of Z can represent.

Suppose that a DN is taught by supervising binary-p codes at its exposed areas, X and Z. When the motor area Z is free, the DN performs, but the output from Z is not always exact due to (a) the DN outputs in real numbers instead of discrete symbols and (b) there are errors in any computer or biological system. The following binary conditioning can prevent error accumulation, which the brain seems to use through spikes.

Definition 6 (Binary conditioning): For any vector from $\mathbf{z} = (z_1, z_2, ..., z_d)$, the binary conditioning of $\mathbf{z}$ forces every real-valued component z_i to be 1 if the pre-action potential of z_i is larger than the machine zero.

The output layer Z that uses binary-p mapping must use the binary conditioning, instead of top-k competition with a fixed k, as the number of firing neurons ranges from 1 to p.

VI. Experiments with DN

Our DN had several versions of experimental embodiments, from networks for general object recognition from 360° views [12], to Where-What Networks that detect (in free viewing), recognize, find (given type or location), multiple objects from natural complex backgrounds [13], to Multilayer In-place Learning Networks (MILN) that learn and process text of natural language [32] (e.g., the part-of-speech tagging problem and the chunking problem using natural languages from the Wall Street Journal), to Where-What Networks that incrementally acquire early language from interactions with environments and also generalize [18]. Preliminary versions of the DN thinking process has been observed by [15], [14] for vision as the DN predicts while learning, and by [18] for language acquisition as the DN predicts across categories and superset and subset while learning. However, the impressive results from such DNs are difficult to understanding without a clear theoretical framework here that links DNs with the well-known automata theory and the mathematical properties presented as the three theorems.

VII. Conclusions

This work focuses on the theory of brain-mind. When the complex nature like the brain-mind has been explained in terms of precise mathematics, the complex nature can be better understood by more analytically trained researchers, regardless their home disciplines.

The new brain-mind theory uses mapping $X(t-1) \times Z(t-1) \mapsto X(t) \times Z(t)$ to model real-time external brain functions. All SNs are special cases of DN in the following sense: An SN allows humans to handcraft its base net, but a DN does not. In other words, an SN is a human handcrafted model outside the brain, while DN is emergent like the brain inside its closed skull.

On one hand, using an SN, the human written symbolic text for each node is for consensual communications among humans only. The machine that runs the SN does not truly

understand such symbolic text. Mathematically, an SN uses handcrafted symbols in Q to sample the vector space Z and uses handcrafted feature detectors to get a symbolic feature set Σ as samples in X. Probabilistic variants of SN do not change the handcraft nature of the base net from Q and Σ. SNs are brittle in real physical world due to the static natures of symbols and the exponential number of elements in the sensory space X and the motor space Z.

On the other hand, existing emergent networks, feed-forward and recurrent, were motivated by brain-like internal computation through emergent internal area Y. However, their learning is slow, not exact, and scruffy.

A GDN is an emergent network, inspired by characteristics of internal brain area Y as discussed in [28]. It learns any complex FA immediately and error-free, through incremental observation of state transitions of the FA one at a time, using a finite memory. In particular, the GDN immediately generalizes, error-free, to many sensorimotor sequences that it has not observed before but are state-equivalent. There are no local minima problems typically associated with a traditional emergent recurrent network, regardless how complex the FA is. After learning the FA as scaffolding, the GDN can freeze its learning and optimally generalize, in the sense of maximum likelihood, for infinitely many input images arising from the real physical world. Alternatively, the GDN can continue to learn and optimally think, in the sense of maximum likelihood, by taking into account all past experience in a resource limited way. In particular, there seems no need for the human programmer to handcraft rigid internal structures, such as modules and hierarchies, for extra-body concepts. Such structures should be emergent and adaptive. For example, the input fields of every neuron should be emergent and adaptive, through mechanisms such as synaptic maintenance (see, e.g., Wang et al. 2011 [27]).

A GDN uses its inner product distance, the incrementally estimated probabilities, and state equivalence to interpolate for infinitely many vector pairs in $X(t-1) \times Z(t-1) \mapsto X(t) \times Z(t)$ from relatively few sample pairs represented in $Y(t-0.5)$. Much future work is needed along the line of GDN autonomous thinking, such as the creativity of GDN.

REFERENCES

[1] J. R. Anderson. *Rules of the Mind*. Lawrence Erlbaum, Mahwah, New Jersey, 1993.

[2] R. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18:193–222, 1995.

[3] A. Fazl, S. Grossberg, and E. Mingolla. View-invariant object category learning, recognition, and search: How spatial and object attention are coordinated using surface-based attentional shrouds. *Cognitive Psychology*, 58:1–48, 2009.

[4] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

[5] C. R. Gallistel. Themes of thought and thinking. *Science*, 285:842–843, 1999.

[6] D. George and J. Hawkins. Towards a mathematical theory of cortical micro-circuits. *PLoS Computational Biology*, 5(10):1–26, 2009.

[7] G. E. Hinton, S. Osindero, and Y-. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[8] J. M. Iverson. Developing language in a developing body: the relationship between motor development and language development. *Journal of child language*, 37(2):229–261, 2010.

[9] M. I. Jordan and C. Bishop. Neural networks. In A. B. Tucker, editor, *CRC Handbook of Computer Science*, pages 536–556. CRC Press, Boca Raton, FL, 1997.

[10] U. R. Karmarkar and D. V. Buonomano. Timing in the absence of clocks: encoding time in neural network states. *Neuron*, 53(3):427–438, 2007.

[11] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[12] M. Luciw and J. Weng. Top-down connections in self-organizing Hebbian networks: Topographic class grouping. *IEEE Trans. Autonomous Mental Development*, 2(3):248–261, 2010.

[13] M. Luciw and J. Weng. Where What Network 3: Developmental top-down attention with multiple meaningful foregrounds. In *Proc. IEEE Int'l Joint Conference on Neural Networks*, pages 4233–4240, Barcelona, Spain, July 18-23 2010.

[14] M. Luciw and J. Weng. Where What Network 4: The effect of multiple internal areas. In *Proc. IEEE 9th Int'l Conference on Development and Learning*, pages 311–316, Ann Arbor,, August 18-21 2010.

[15] M. Luciw, J. Weng, and S. Zeng. Motor initiated expectation through top-down connections as abstract context in a physical world. In *IEEE Int'l Conference on Development and Learning*, pages +1–6, Monterey, CA, Aug. 9-12 2008.

[16] M. D. Mauk and D. V. Buonomano. The neural basis of temporal processing. *Annual Review of Neuroscience*, 27:307–340, 2004.

[17] M. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):34–51, 1991.

[18] K. Miyan and J. Weng. WWN-Text: Cortex-like language acquisition with What and Where. In *Proc. IEEE 9th Int'l Conference on Development and Learning*, pages 280–285, Ann Arbor, August 18-21 2010.

[19] V. Müller. The hard and easy grounding problems. *AMD Newsletter*, 7(1):8–9, 2010.

[20] B. A. Olshaushen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 13 1996.

[21] T. T. Rogers and J. L. McClelland. Preecis of semantic cognition: A parallel distributed processing approach. *Behavioral and Brain Sciences*, 31:689–749, 2008.

[22] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

[23] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[24] J. B. Tenenbaum, T. L. Griffithsb, and C. Kemp. Theory-based bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10(7):309–318, 2006.

[25] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331:1279–1285, 2011.

[26] L. S. Vygotsky. *Thought and language*. MIT Press, Cambridge, Massachussetts, 1962. trans. E. Hanfmann & G. Vakar.

[27] Y. Wang, X. Wu, and J. Weng. Synapse maintenance in the where-what network. In *Proc. Int'l Joint Conference on Neural Networks*, pages +1–8, San Jose, CA, July 31 - August 5 2011.

[28] J. Weng. A 5-chunk developmental brain-mind network model for multiple events in complex backgrounds. In *Proc. Int'l Joint Conf. Neural Networks*, pages 1–8, Barcelona, Spain, July 18-23 2010.

[29] J. Weng. Three theorems about developmental networks and the proofs. Technical Report MSU-CSE-11-9, Department of Computer Science, Michigan State University, East Lansing, Michigan, May,12 2011.

[30] J. Weng and M. Luciw. Dually optimal neuronal layers: Lobe component analysis. *IEEE Trans. Autonomous Mental Development*, 1(1):68–85, 2009.

[31] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[32] J. Weng, Q. Zhang, M. Chi, and X. Xue. Complex text processing by the temporal context machines. In *Proc. IEEE 8th Int'l Conference on Development and Learning*, pages +1–8, Shanghai, China, June 4-7 2009.

[33] Y. Yamashita and J. Tani. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Computational Biology*, 4(11):e1000220, 2008.