Establish the Three Theorems: Brain-Like DNs Logically Reason and Optimally Generalize

Juyang Weng

***Abstract*—In artificial intelligence (AI) there are two major schools, symbolic and connectionist. Weng 2011 [6] proposed three major properties of the Developmental Network (DN) which bridged the two schools: (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. This paper presents the three major theorems and their proofs.**

I. Introduction

The major differences between a symbolic network (SN) and a Developmental Network (DN) are illustrated in Fig. 1.

Marvin Minsky 1991 [4] 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.

Computationally, feed-forward connections serve to feed sensory features [5] to motor area for generating behaviors. It has been reported that feed-backward connections can serve as class supervision [2], attention [1], and storage of time information.

In the following, we analyze how the DN theory bridges the symbolic school and the connectionist school.

II. DP Algorithm

The small DP algorithm self-programs logic into a huge DN directly from physics. A DN 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. 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.

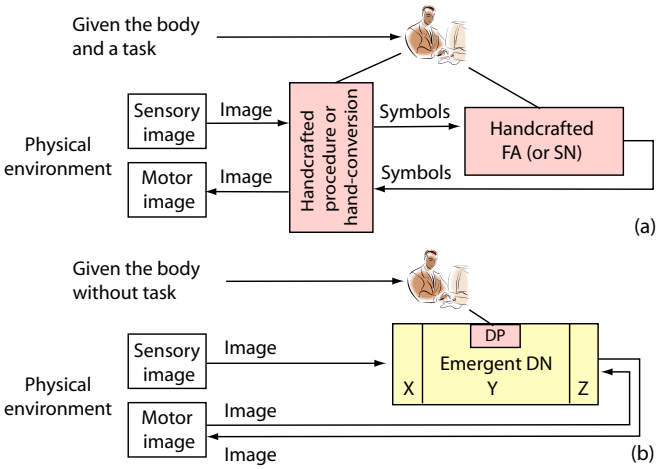

Fig. 1. Comparison between a symbolic FA (or SN) and an emergent DN. (a) Given a task, an FA (or SN), symbolic, handcrafted by the human programmer using a static symbol set. (b) A DN, which incrementally learns the FA but takes sensory images directly and produces motor images directly. Without given 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.

Algorithm 1 (DP): 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 $\mathbf{r} \leftarrow \mathbf{r}'$.

Juyang Weng is a professor at 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) and a visiting professor at the School of Computer Science and Engineering at Fudan University, Shanghai, China.

The author would like to thank Hao Ye at Fudan University who carefully proof-read the proofs presented here and raised two gaps that I have filled since then. The author would also like to thank Z. Ji, M. Luciw, K. Miyan and other members of the Embodied Intelligence Laboratory at Michigan State University; Q. Zhang, Yuekai Wang, Xiaofeng Wu and other members of the Embodied Intelligence Laboratory at Fudan University whose work have provided experimental supports for the theory presented here.

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.

The area function f which is based on the theory of Lobe Component Analysis (LCA) [7], 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 $\dot{\mathbf{p}} = (\dot{\mathbf{b}}, \dot{\mathbf{t}}) = (\mathbf{b}/\|\mathbf{b}\|, \mathbf{t}/\|\mathbf{t}\|)$.

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 $r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t})$.

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.

III. Formulations

As we need a slight deviation from the standard definition of FA, let us look at the standard definition first.

Definition 1 (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 states, consists of symbols. 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.

This classical definition is for a language acceptor, which accepts all strings x from the alphabet Σ that belongs to a language L. It has been proved [3] that given any *regular language* L from alphabet Σ, there is an FA that accepts L, meaning that it accepts exactly all $\mathbf{x} \in L$ but no other string not in L. Conversely, given any FA taking alphabet Σ, the language L that the FA accepts is a regular language. However, a language FA, just like any other automata, only deals syntax not semantics. The semantics is primary for understanding a language and the syntax is secondary.

We need to extend the definition of FA for agents that run at discrete times, as follows:

Definition 2 (Agent FA): A finite automaton (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 inputs to an FA are symbolic. The input space is denoted as $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_l\}$, which can be a discretized version of a continuous space of input. In sentence recognition, the FA reads one word at a time. The number l is equal to the number of all possible words — the size of the vocabulary. For a computer game agent, l is equal to the total number of different percepts.

The outputs (actions) from a language acceptor FA are also symbolic, $A = \{a_1, a_2, ..., a_n\}$ which can also be a discretized version of a continuous space of output. For a sentence detector represented by an FA, when the FA reaches the last state, its action reports that the sentence has been detected.

An agent FA is an extension from the corresponding language FA, in the sense that it outputs the state, not only the acceptance property of the state. The meanings of each state, which are handcrafted by the human programmer but are not part of the formal FA definition, are only in the mind of the human programmer. Such meanings can indicate that a state is an accepting state or not, as a special case of many other meanings associated with the state. However, such concepts are only in the mind of the human system designer, not something that the FA is "aware" of. This is a fundamental limitation of all symbolic models. The Developmental Network (DN) described below do not use any symbols, but instead (image) vectors from the real-world

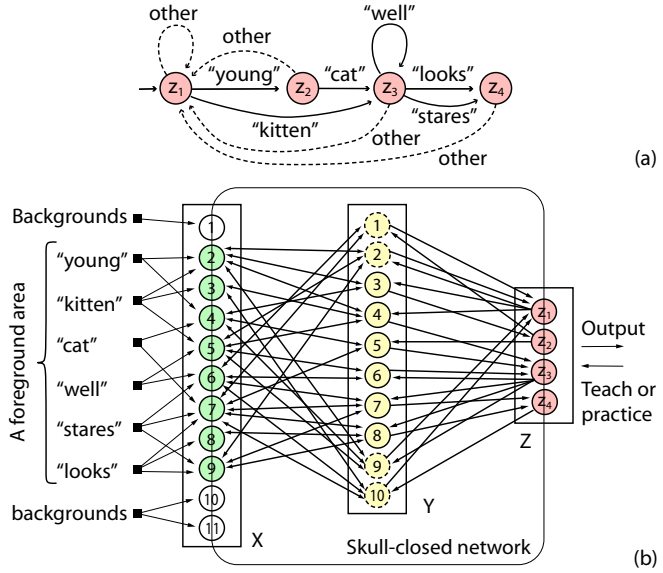

Fig. 2. Conceptual correspondence between an Finite Automaton (FA) with the corresponding DN. (a) An FA, handcrafted and static. (b) A corresponding DN that simulates the FA. It was taught to produce the same input-out relations as the FA in (a). A symbol (e.g., z_2) in (a) corresponds to an image (e.g., $(z_1, z_2, ..., z_4) = (0, 1, 0, 0)$) in (b).

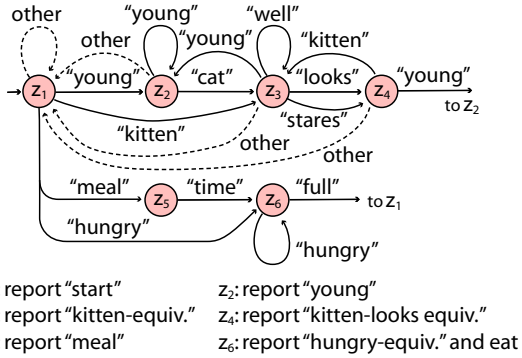

Fig. 3. An FA simulates an animal. Each circle indicates a context state. The system starts from state z_1. Supposing the system is at state q and receives a symbol σ and the next state should be q', the diagram has an arrow denoted as $q \xrightarrow{\sigma} q'$. A label "other" means any symbol other 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.

sensors and real-world effectors. As illustrated in Fig. 2, a DN is grounded in the physical environment but an FA is not.

Fig. 3 gives an example of the agent FA. Each state is associated with a number of cognitive states and actions, shown as text in the lower part of Fig. 3, reporting action for cognition plus a motor action. The example in Fig. 3 shows that an agent FA can be very general, simulating an animal in a micro, symbolic world. The meanings of each state in the lower part of Fig. 3 are handcrafted by, and only in the mind of, the human designer. These meanings are not a part of the FA definition and are not accessible by the machine that simulates the FA.

Without loss of generality, we can consider that an agent FA simply outputs its current state at any time, since the state is uniquely linked to a pair of the cognition set and the action set, at least in the mind of human designer.

A. Completeness of FA

It has been proved [3] 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 indistinguishable by the FA. Since these strings are indistinguishable, 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. 3. The FA partitions all possible strings into 6 equivalent classes. $[\Lambda]$ = ["calculus"] as the agent does not know about "calculus" although it is in Σ. 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. Given a vocabulary Σ representing the elements of a symbolic world, a natural language L is defined in terms of Σ where the meanings of all sentences in L are defined by the set of equivalent classes, denoted by Q. 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.

B. DN Simulates FA

Next, let us consider how a DN learns to simulate any FA. First we consider the mapping from symbolic sets Σ and Q, to vector spaces X and Z.

Definition 3 (Symbol-to-vector mapping): A symbol-to-vector mapping m is a one-to-one 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 $v(t) = \sum_{t-T<t_i\leq t} s(t_i)/T$, 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 4 (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 m is one to one: That is, if $\mathbf{z}_i \equiv m(q_i)$ then $q_i \neq q_j$ implies $\mathbf{z}_i \neq \mathbf{z}_j$.

The larger the p the more symbols the space of Z can represent. However, through a binary-p mapping, each symbol q_i always has a unique vector $\mathbf{z} \in Z$. Note that different q's are mapped to not only different $\mathbf{z}$'s but also different directions of $\mathbf{z}$'s as the input $\mathbf{p}$ of DN is a unit $\dot{\mathbf{p}}$.

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 5 (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-response of z_i is larger than the machine zero — a small positive bound estimating computer round-off noise.

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.

C. DP for Generative DN (GDN)

Algorithm 2 (DP for GDN): A GDN is a DN that gives the following specific way of initialization. It starts from pre-specified dimensions for the X and Z areas, respectively. X represents receptors and is totally determined by the current input. But it incrementally generates neurons in Y from an empty Y. Each neuron in Z is initialized by a synaptic vector $\mathbf{v}$ of dimension 0, age 0. Suppose $V = \{\mathbf{v}_i = (\mathbf{x}_i, \mathbf{z}_i) \mid \mathbf{x} \in X, \mathbf{z} \in Z, i = 1, 2, ..., c\}$ is the current synaptic vectors in Y. Whenever the network takes an input $\mathbf{p} = (\mathbf{x}, \mathbf{z})$, compute the pre-resppnses in Y. If the top-1 winner in Y has a pre-response lower than 2 (i.e., $\mathbf{p} \notin V$), simulate mitosis-equivalent by doing the following:

1) Increment the number of neurons, $c \leftarrow c + 1$.
2) Add a new Y neuron. Set the weight vector $\mathbf{v} = \dot{\mathbf{p}}$, its age to be 0, and its pre-response to be 2 since it is the perfect match based on Eq. (1). There is no need to recompute the pre-responses.

The response value of each Z neuron is determined by the starting state (e.g., background class). As soon as the first Y neuron is generated, every Z neuron will add the first dimension in its synaptic vector in the following DN update. This way, the dimension of its weight vector continuously increases together with the number c of Y neurons.

Lemma 1 (Properties of a GDN): Suppose a GDN simulates any given FA using top-1 competition for Y, binary-p mapping, and binary conditioning for Z, and update at least twice in each unit time. Each input $\mathbf{x}(t-1)$ is retained during all DN updates in $(t-1, t]$. Such a GDN has the following properties for $t = 1, 2, ...$:

1) The winner Y neuron matches perfectly with input $\mathbf{p}(t-1) \equiv (q(t-1), \sigma(t-1))$ with $\mathbf{v} = \dot{\mathbf{p}}$ and fires, illustrated in Fig. 4(a) as a single transition edge (red).
2) All the synaptic vectors in Y are unit and they never change once initialized, for all times up to t. They only advance their firing ages. The number of Y neurons c is exactly the number of learned state transitions up to time t.
3) Suppose that the weight vector $\mathbf{v}$ of each Z neuron is $\mathbf{v} = (p_1, p_2, ..., p_{c(Y)})$, and Z area uses the learning rate straight recursive average $w_2(n_j) = 1/n_j$. Then the weight p_j from the j-th Y neuron to each Z neuron is
$$\begin{aligned} p_j &= \text{Prob}(j\text{-th } Y \text{ neuron fires} \mid \text{the } Z \text{ neuron fires}) \\ &= f_j/n, \end{aligned} \tag{6}$$
$j = 1, 2, ..., c(Y)$, where f_j is the number of times the j-th Y neuron has fired conditioned on that the Z neuron fires, and n is the total number of times the Z neuron has fired.
4) Suppose that the FA makes transition $q(t-1) \stackrel{\sigma(t-1)}{\longrightarrow} q(t)$, as illustrated in Fig. 4(a). After the 2nd DN update, Z outputs $\mathbf{z}(t) \equiv q(t)$, as long as Z of DN is supervised for the 2nd DN update when the transition is received by Z the first time. Z then retains the values automatically till the end of the first DN update after t.

Proof: The proof below is a constructive proof, instead of an existence one. To facilitate understanding, the main ideas are illustrated in Fig. 4. Let the X of the DN take the equivalent inputs from Σ using a symbol-to-vector mapping. Let Z be supervised as the equivalent states in Q, using a binary-p mapping. The number of firing neurons Z depends on the binary-p mapping. The DN lives in the simulated sensorimotor world $X \times Z$ determined by the sensory symbol-to-vector mapping: $m_x : \Sigma \mapsto X$ and the binary-p symbol-to-vector mapping $m_z : Q \mapsto Z$.

We prove it using induction on integer t.

Basis: When $t = 0$, set the output $\mathbf{z}(0) \equiv q(0) = q_0$ for the DN. Y has no neuron. Z neurons have no synaptic weights. All the neuronal ages are zeros. The properties 1, 2, 3 and 4 are trivially true for $t = 0$.

Hypothesis: We hypothesize that the above four properties are true up to integer time t. In the following, we prove that the above properties are true for $t + 1$.

Induction step: During t to $t + 1$, suppose that the FA makes transition $q(t) \stackrel{\sigma(t)}{\longrightarrow} q(t + 1)$. The DN must do the equivalent, as shown below.

At the next DN update, there are two cases for Y: Case 1: the transition is observed by the DN as the first time. Case 2: the DN has observed the transition.

Case 1: new Y input. First consider Y. As the input $\mathbf{p}(t) = (\mathbf{x}(t)), \mathbf{z}(t))$ to Y is the first time, $\dot{\mathbf{p}} \notin V$. Y initializes a new neuron whose weight vector is initialized as $\mathbf{v}_j = \dot{\mathbf{p}}(t)$ and age $n_j = 0$. The number of Y neurons c is incremented by 1 as this is a newly observed state

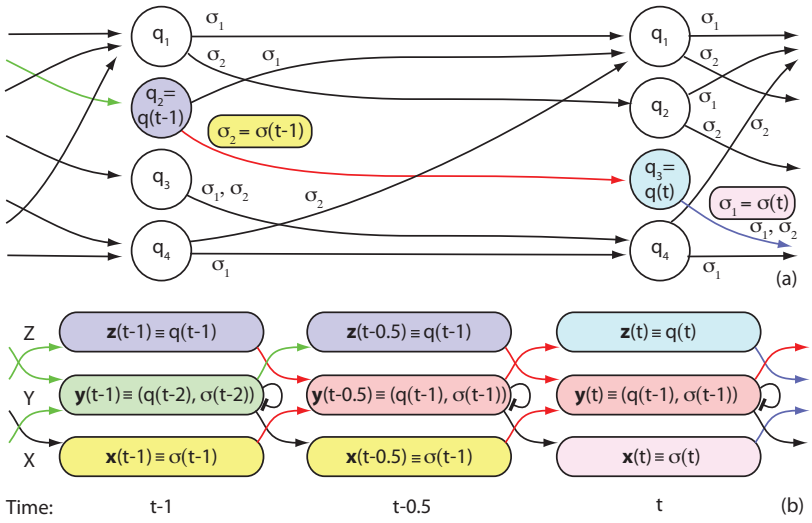

Fig. 4. Model the brain mapping, DN, and SN. In general, the brain performs external mapping $b(t) : X(t-1) \times Z(t-1) \mapsto X(t) \times Z(t)$ on the fly. (a) An NS samples the vector space Z using symbolic set Q and X using Σ, to compute symbolic mapping $Q(t-1) \times \Sigma(t-1) \mapsto Q(t)$. 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.

transition. From the hypothesis, all previous Y neurons in V are still their originally initialized unit vectors. Thus, the newly initialized $\mathbf{v}_j$ is the only Y neuron that matches $\dot{\mathbf{p}}(t)$ exactly. With $k = 1$, this new Y neuron is the unique winner and it fires with $y_j = 1$. Its Hebbian learning gives age advance $n_j \leftarrow n_j + 1 = 0 + 1 = 1$ and Eq.(3) leads to

$$\begin{aligned} \mathbf{v}_j &\leftarrow w_1(n_j)\dot{\mathbf{p}} + w_2(n_j) \cdot 1 \cdot \dot{\mathbf{p}} \\ &= (w_1(n_j) + w_2(n_j))\dot{\mathbf{p}} = 1 \cdot \dot{\mathbf{p}} = \dot{\mathbf{p}}. \end{aligned} \quad (7)$$

As DN updates at least twice in the unit time, Y area is updated again for the second DN update. But X and Z retain their values within each unit time, per simulation rule. Thus, the Y winner is still the same new neuron and its vector still does not change as the above expression is still true. Thus, properties 1 and 2 are true for the first two DN updates within $(t, t+1]$.

Next consider Z. Z retains its values in the first DN update, per hypothesis. For the 2nd DN update, the response of Z is regarded the DN's Z output for this unit time, which uses the above Y response as illustrated in Fig. 4. In Case 1, Z must be supervised for this second DN update within the unit time. According to the binary-p mapping from the supervised $q(t+1)$, Eq. (3) is performed for up to p Z neurons:

$$\mathbf{v}_j \leftarrow w_1(n_j)\dot{\mathbf{v}}_j + w_2(n_j) \cdot 1 \cdot \dot{\mathbf{p}}. \quad (8)$$

Note that Z has only bottom input $\mathbf{p} = \mathbf{y}$ and the normalized vector $\dot{\mathbf{p}}$ is binary. That is, only one component (the new one) in $\dot{\mathbf{p}}$ is 1 and all other components are zeros. All Z neurons do not link with this new Y neuron before the 2nd DN update. Consider two subcases, subcase (1.a) the Z neuron should fire at the end of this unit time, and subcase (1.b) the Z neuron should not fire.

Subcase (1.a): the Z neuron should fire. All Z neurons that should fire, up to p of them, are supervised to fire for the 2nd DN update by the Z area function. Suppose that a supervised-to-fire Z neuron has a synapse vector $\mathbf{v} = (p_1, p_2, ..., p_c)$ with the new p_c just initialized to be 0 since the new Y neuron $j = c$ now fires. From the hypothesis, $p_i = f_i/n$, $i = 1, 2, ..., c-1$. But, according to the Z initialization in GDN, $p_c = 0$ for the new dimension initialization. Then from $0 = p_c = f_c/n$, we have $f_c = 0$ which is correct for f_c. From Eq. (3), the c-th component of $\mathbf{v}$ is

$$v_c \leftarrow \frac{n}{n+1} \cdot \frac{f_c}{n} + \frac{1}{n+1} \cdot 1 \cdot 1 = \frac{f_c + 1}{n+1} = \frac{1}{n+1}. \quad (9)$$

which is the correct count for the new v_c, and the other components of $\mathbf{v}$ are

$$v_i \leftarrow \frac{n}{n+1} \cdot \frac{f_i}{n} + \frac{1}{n+1} \cdot 1 \cdot 0 = \frac{f_i + 0}{n+1} = \frac{f_i}{n+1}, \quad (10)$$

for all $i = 1, 2, ..., c-1$, which is also the correct count for other components of the $\mathbf{v}$ synaptic vector. Every firing Z neuron advances its age by 1 and correctly counts the firing of the new c-th Y neuron. As Y response does not change for more DN updates within $(t, t+1]$ and the firing Y neuron meets a positive $1/n_j$ weight to the firing Z neuron with

age n_j, the Z area does not need to be supervised after the second DN update within $(t, t+1]$.

Subcase (1.b): the Z neuron should not fire. All Z neurons that should not fire must be supervised to be zero (not firing). All such Z neurons cannot be linked with the new Y neuron before, as it was not present. The new added weight for this new Y neuron is initialized to 0 in the Z area function. All these non-firing neurons keep their counts and ages unchanged. As Y response does not change for more DN updates within $(t, t+1]$, the Z area does not need to be supervised after the second DN update within $(t, t+1]$, since the only firing Y neuron meets a 0 weight to the Z neuron.

The binary conditioning for Z makes sure that all the Z neurons that have a positive pre-response to fire fully. That is, the properties 3 and 4 are true from the first two ND updates within $(t, t+1]$.

Case 2: old Y input. First consider Y. To Y, $\mathbf{p}(t) = (\mathbf{x}(t), \mathbf{z}(t))$ has been an input before. From the hypothesis, the winner Y neuron j exactly matches $\dot{\mathbf{p}}(t)$, with $\mathbf{v}_j = \dot{\mathbf{p}}(t)$. Eq. (7) still holds using the inductive hypothesis, as the winner Y neuron fires only for a single $\dot{\mathbf{p}}$ vector. Thus, properties 1) and 2) are true from the first ND update within $(t, t+1]$.

Next consider Z. Z retains its previous vector values in the first DN update, per hypothesis. In the 2nd DN update, the transition is not new, we show that Z does not need to be supervised during the unit time $(t, t+1]$ to fire perfectly. From Eq. (1), the Z pre-response is computed by

$$r(\mathbf{v}_b, \mathbf{b}) = \frac{\mathbf{v}_b}{\|\mathbf{v}_b\|} \cdot \frac{\mathbf{b}}{\|\mathbf{b}\|} = \frac{\mathbf{v}_b}{\|\mathbf{v}_b\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|} \tag{11}$$

where $\dot{\mathbf{y}}$ is binary with only a single positive component and $\mathbf{t}$ is absent as Z does not have a top-down input. Suppose that Y neuron j fired in the first DN update. From the hypothesis, every Z neuron has a synaptic vector $\mathbf{v} = (p_1, p_2, ..., p_c)$, where $p_j = f_j/n$ counting up to time t, where f_i is the observed frequency (occurrences) of Y neuron j firing, $i = 1, 2, ..., c$, and n is the total number of times the Z neuron has fired. Consider two sub-cases: (2.a) the Z neuron should fire according to the transition, and (2.b) the Z neuron should not.

For sub-case (2.a) where the Z neuron should fire, we have

$$\begin{aligned} r(\mathbf{v}_b, \mathbf{b}) &= r(\mathbf{v}, \mathbf{y}) = \dot{\mathbf{v}} \cdot \dot{\mathbf{y}} = \frac{p_j}{\|\mathbf{v}\|} \cdot 1 = \frac{p_j}{\|\mathbf{v}\|} \\ &= \frac{f_j/n}{\|\mathbf{v}\|} = \frac{f_j}{n\|\mathbf{v}\|} > 0 \end{aligned}$$

because the Z neuron has been supervised at least the first time for this transition and thus $f_j \geq 1$. We conclude that the Z neuron guarantees to fire at 1 after its binary conditioning. From Eq. (3), the j-th component of $\mathbf{v}$ is:

$$v_j \leftarrow \frac{n}{n+1} \cdot \frac{f_j}{n} + \frac{1}{n+1} \cdot 1 \cdot 1 = \frac{f_j + 1}{n+1}. \tag{12}$$

which is the correct count for the j-th component, and the other components of $\mathbf{v}$ are:

$$v_i \leftarrow \frac{n}{n+1} \cdot \frac{f_i}{n} + \frac{1}{n+1} \cdot 1 \cdot 0 = \frac{f_i + 0}{n+1} = \frac{f_i}{n+1}, \tag{13}$$

for all $i \neq j$, which is also the correct count for all other components in $\mathbf{v}$. The Z neuron does not need to be supervised after the second DN update within $(t, t+1]$ but still keeps firing. This is what we want to prove for property 3 for every firing Z neuron.

Next consider sub-case (2.b) where the Z neuron should not fire. Similarly we have $r(\mathbf{v}_b, \mathbf{b}) = r(\mathbf{v}, \dot{\mathbf{y}}) = f_j/(n\|\mathbf{v}\|) = 0$, from the hypothesis that this Z neuron fires correctly up to time t and thus we must have $f_j = 0$. Thus, they do not fire, change their weights, or advance their ages. The Z neuron does not need to be supervised after the second DN update within $(t, t+1]$ but keeps not firing. This is exactly what we want to prove for property 3 for every non-firing Z neuron.

Combining the sub-cases (2.a) and (2.b), all the Z neurons act perfectly and the properties 3 and 4 are true for the first two DN updates. We have proved for Case 2, old Y input.

Therefore, the properties 1, 2, 3, 4 are true for first two DN updates. If DN has time to continue to update before time $t+1$, we see that we have always Case 2 for Y and Z within the unit time and Y and Z retain their responses since the input $\mathbf{x}$ retains its vector value. Thus, the properties 1, 2, 3, 4 are true for all DN updates within $(t, t+1]$.

According to the principle of induction, we have proved that the properties 1, 2, 3 and 4 are all true for all t. ■

D. Theorem 1: DN simulates FA incrementally, immediately, and error-free

Using the above lemma, we are ready to prove:

Theorem 1 (Simulate any FA as scaffolding): The general-purpose DP incrementally grows a GDN to simulate any given FA $M = (Q, \Sigma, q_0, \delta, A)$, error-free and on the fly, if 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.

Proof: Run the given FA and the GDN at discrete time t, $t = 1, 2, ...$ Using the lemma above, each state transition $q \xrightarrow{\sigma} q'$ is observed by the DN via the mappings m_x and m_z. Update the DN at least twice in each unit time. In DN, if $\mathbf{p} = (\mathbf{z}, \mathbf{x})$ is a new vector to Y, Y adds a new neuron. Further, from the proof of the above lemma, we can see that as soon as each transition in FA has been taught, the DN has only Case 2 for the same transition in the future, which means that no need for second supervision for any transition. Also from the proof of the lemma, the number of Y neurons corresponds to the number of state transitions in the FA. ■

If the training data set is finite and consistent (the same (q, σ) must go to the unique next state q'), re-substitution test (using the training set) corresponds to simulating an FA using pattern codes. Theorem 1 states that for the DGN any resubstitution test for consistent training data is always immediate and error-free. Conventionally, this will mean that the system over-fits data as its generalization will be poor.

However, the DGN does not over-fit data as the following Theorem 2 states, since the nature of its parameters is optimal and the size of the parameter set is dynamic. In other words, it is optimal for disjoint tests.

Definition 6 (Grounded DN): Suppose that the symbol-to-vector mapping for the DN is consistent with the real sensor of the a real-world agent (robot or animal), namely, each symbol σ for FA is mapped to an sub-image $\mathbf{x}$ from the real sensor, excluding the parts of the irrelevant background in the scene. Then the DN that has been trained for the FA is called grounded.

For a grounded DN, the SN is a human knowledge abstraction of the real world. After training, a grounded DN can run in the real physical world, at least in principle. However, as we discussed above, the complexity of symbolic representation for Σ and Q is exponential in the number of concepts. Therefore, it is intractable for any SN to sufficiently sample the real world since the number of symbols required is too many for a realistic problem. The fact that there are enough symbols to model the real world causes the symbolic system to be brittle. All the probability variants of FA can only adjust the boundaries between any two nearby symbols, but the added probability cannot resolve the fundamental problem of the lack of sufficient number of symbols.

E. Theorem 2: DN generalizes optimally while frozen

The next theorem states how the frozen GDN generalizes for infinitely many sensory inputs.

Theorem 2 (DN generalization while frozen): Suppose that after having experienced all the transitions of the FA, from time $t = t_0$ the GDN turns into a DN that

1) freezes: It does not generate new Y neurons and does not update its adaptive part.
2) generalizes: It continues to generate responses by taking sensory inputs not restricted to the finite ones for the FA.

Then the DN generates the Maximum Likelihood (ML) action $\mathbf{z}_n(t)$, recursively, for all integer $t > t_0$:

$$n(t) = \arg\max_{\mathbf{z}_i \in Z} \ \mathrm{h}\ (\dot{\mathbf{p}}(t-1)|\mathbf{z}_i(t), \mathbf{z}(t-1)). \tag{14}$$

where the probability density $\mathrm{h}(\dot{\mathbf{p}}(t-1)|\mathbf{z}_i(t), \mathbf{z}(t-1))$ is the probability density of the new last observation $\dot{\mathbf{p}}(t-1)$, with the parameter vector $\mathbf{z}_i$, conditioned on the last executed action $\mathbf{z}(t-1)$, based on its experience gained from learning the FA.

Proof: Reuse the proof of the lemma. Case 1 does not apply since the DN does not generate new neurons. Only Case 2 applies.

First consider Y. Define c Voronoi regions in $X \times Z$ based on now frozen $V = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_c)$, where each R_j consisting of $\dot{\mathbf{p}}$ vectors that are closer to $\dot{\mathbf{v}}_j$ than to other $\dot{\mathbf{v}}_i$:

$$R_j = \{\dot{\mathbf{p}} \mid j = \arg\max_{1 \le i \le c} \dot{\mathbf{v}}_i \cdot \dot{\mathbf{p}}\}, j = 1, 2, ..., c.$$

Given observation $\mathbf{p}(t-1)$, V has two sets of parameters, the X synaptic vectors and the Z synaptic vectors. They are frozen.

According to the dependence of parameters in DN, first consider consider c events for area Y: $\dot{\mathbf{p}}(t-1)$ falls into R_i, $i = 1, 2, ..., c$ partitioned by the c Y vectors in V. The conditional probability density $g(\dot{\mathbf{p}}(t-1)|\dot{\mathbf{v}}_i, \mathbf{z}(t-1))$ is zero if $\dot{\mathbf{p}}(t-1)$ falls out of the Voronoi region of $\dot{\mathbf{v}}_i$:

$$g(\dot{\mathbf{p}}(t-1)|\mathbf{v}_i, \mathbf{z}(t-1)) = \begin{cases} g_i(\dot{\mathbf{p}}(t-1)|\mathbf{v}_i, \mathbf{z}(t-1)) & \text{if } \dot{\mathbf{p}}(t-1) \in R_i \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $g_i(\dot{\mathbf{p}}(t-1)|\mathbf{v}_i, \mathbf{z}(t-1))$ is the probability density within R_i. Note that the distribution of $g_i(\dot{\mathbf{p}}(t-1)|\mathbf{v}_i, \mathbf{z}(t-1))$ within R_i is irrelevant as long as it integrates to 1.

Note that $\mathbf{p}(t-1) = (\mathbf{x}(t-1), \mathbf{z}(t-1))$. Given $\dot{\mathbf{p}}(t-1)$, the ML estimator for the binary vector $\mathbf{y}_j \in E_1^c$ needs to maximize $g(\dot{\mathbf{p}}(t-1)|\mathbf{v}_i, \mathbf{z}(t-1))$, which is equivalent to finding

$$j = \arg\max_{1 \le i \le c} g(\dot{\mathbf{p}}(t-1)|\mathbf{v}_i, \mathbf{z}(t-1)) = \arg\max_{1 \le i \le c} \dot{\mathbf{v}}_i \cdot \dot{\mathbf{p}}(t-1), \tag{16}$$

since finding the ML estimator j for Eq. (15) is equivalent to finding the Voronoi region to which $\dot{\mathbf{p}}(t-1)$ belongs to. This is exactly what the Y area does, supposing $k = 1$ for top-k competition.

Next, consider Z. The set of all possible binary-1 Y vectors and the set of producible binary-p Z vectors have a one-to-one correspondence: $\mathbf{y}_j$ corresponds to $\mathbf{z}_n$ if and only if the single firing neuron in $\mathbf{y}_j$ has non-zero connections to all the firing neurons in the binary-p $\mathbf{z}_n$ but not to the non-firing neurons in $\mathbf{z}_n$. Namely, given the winner Y neuron j, the corresponding $\mathbf{z} \in Z$ vector is deterministic. Furthermore, for each Y neuron, there is only unique $\mathbf{z}$ because of the definition of FA. Based on the definition of probability density, we have:

$$g(\dot{\mathbf{p}}(t-1)|\mathbf{v}_j, \mathbf{z}(t-1)) = h(\dot{\mathbf{p}}(t-1)|\mathbf{z}_n(t), \mathbf{z}(t-1))$$

for every $\mathbf{v}_j$ corresponding to $\mathbf{z}_n(t)$. Thus, when the DN generates $\mathbf{y}(t-0.5)$ in (16) for ML estimate, its Z area generates ML estimate $\mathbf{z}_n(t)$ that maximizes (14). ■

F. Theorem 3: DN thinks optimally

There seems no more proper terms to describe the nature of the DN operation other than "think." The thinking process by the current basic version of DN seems similar to, but not exactly the same as, that of the brain. At least, the richness of the mechanisms in DN that has demonstrated experimentally to be close to that of the brain.

Theorem 3 (DN generalization while updating): Suppose that after having experienced all the transitions of the FA, from time $t = t_0$ the GDN turns into a DN that

1) fixes its size: It does not generate new Y neurons
2) adapts: It updates its adaptive part $N = (V, A)$.
3) generalizes: It continues to generate responses by taking sensory inputs not restricted to the finite ones for the FA.

Then the DN "thinks" (i.e., learns and generalizes) recursively and optimally: For all integer $t > t_0$, the DN recursively generates the Maximum Likelihood (ML) response $\mathbf{y}_j(t-0.5) \in E_1^c$: with

$$j = \arg\max_{1 \le i \le c} g(\dot{\mathbf{p}}(t-1) | \dot{\mathbf{v}}_i(t-1), \mathbf{z}(t-1)) \quad (17)$$

where $g(\dot{\mathbf{p}}(t-1)|\dot{\mathbf{v}}_i(t-1), \mathbf{z}(t-1))$ is the probability density, conditioned on $\dot{\mathbf{v}}_i(t-1), \mathbf{z}(t-1)$. And the Z has the pre-response vector $\mathbf{z}(t) = (r_1, r_2, ..., r_{c(Z)})$, where r_n, $n = 1, 2, ..., c(Z)$, is the conditional probability for the n-th Z neuron to fire:

$$\begin{aligned} r_n &= p_{nj}(t) = \text{Prob}(j\text{-th } Y \text{ neuron fires at time } t-0.5 \\ &\quad | \quad n\text{-th } Z \text{ neuron fires at time } t). \end{aligned} \quad (18)$$

The firing of each Z neuron has a freedom to choose a binary conditioning method to map the above the pre-response vector $\mathbf{z} \in R^{c(Z)}$ to the corresponding binary vector $\mathbf{z} \in B^{c(Z)}$.

Proof: Again, reuse the proof of the lemma with the synaptic vectors of Y to be $V(t-1) = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_c)$ now adapting.

First consider Y. Eq (16) is still true as this is what DN does but V is now adapting. The probability density in Eq. (15) is the currently estimated version based on past experience but V is now adapting. Then, when $k = 1$ for top-k Y area competition, the Y response vector $\mathbf{y}_j(t-0.5) \in E_1^c$ with j determined by Eq. (16) gives Eq.(17). In other words, the response vector from Y area is again the Maximum Likelihood (ML) estimate from the incrementally estimated probability density. The major difference between Eq.(16) and Eq.(17) is that in the latter, the adaptive part of the DN updates.

Next, consider Z. From the proof of the Lemma 1, the synaptic weight between the j-th Y neuron and the n-th Z neuron is

$$\begin{aligned} p_{nj} &= \text{Prob}(j\text{-th } Y \text{ neuron fires in the last DN update} \\ &\quad | \quad n\text{-th } Z \text{ neuron fires in the next DN update}). \end{aligned} \quad (19)$$

The total pre-response for the n-th neuron is

$$r_n = r(\mathbf{v}_n, \mathbf{y}) = \dot{\mathbf{v}}_n \cdot \dot{\mathbf{y}} = p_{nj} y_j = p_{nj} 1 = p_{nj},$$

since the j-th neuron is the only firing Y neuron at this time. The above two expressions give Eq. (18). ■

The last sentence in the theorem gives the freedom for Z to choose a binary conditioning method but a binary conditioning method is required in order to determine which Z neurons fire and all other Z neurons do not. In the brain, neural modulation (e.g., expected punishment, reward, or novelty) discourages or encourages the recalled components of $\mathbf{z}$ to fire.

The adaptive mode after learning the FA is autonomous inside the DN. A major novelty of this theory of thinking is that the structure inside the DN is fully emergent, regulated by the DP (i.e., nature) and indirectly shaped (i.e., nurture) by the external environment.

The neuronal resource of Y gradually re-distribute according to the new observations in $Y \times X$. It adds new context-sensory experience and gradually weights down prior experience. Over the entire life span, more often observed experience and less often observed experience are proportionally represented as the synaptic weights.

However, an adaptive DN does not simply repeat the function of the FA it has learned. Its new thinking experience includes those that are not applicable to the FA. The following cases are all allowed in principle:

(1) Thinking with a "closed eye": A closed eye sets $\mathbf{x} = \mathbf{u}$ where $\mathbf{u}$ has 0.5 for all its components (all gray image). The DN runs where Y responses mainly to $\mathbf{z}$ as $\mathbf{x}$ has little "preference" in matching.

(2) Thinking with an "open eye": In the sensory input $\mathbf{x}$ is different from any prior input.

(3) Inconsistent experience: From the same $(\mathbf{z}, \mathbf{x}) \equiv (q, \sigma)$, the next $\mathbf{z}' \equiv q'$ may be different at different times. FA does not allow any such inconsistency. However, the inconsistencies allow occasional mistakes, update of knowledge structures, and possible discovery of new knowledge.

The neuronal resources of Y gradually re-distribute according to the new context-motor experience in $Y \times Z$. The learning rate $w_2(n_j) = 1/n_j$ amounts to equally weighted average for past experience by each neuron. Weng & Luciw 2009 [7] investigated amnesic average to give more weight to recent experience.

In the developmental process of a DN, there is no need for a rigid switch between FA and the real-world learning.

The binary conditioning is suited only when Z is supervised according to the FA to be simulated. As the "thinking" of the DN is not necessarily correct, it is not desirable to use the binary conditioning for Z neurons.

The thinking process by the current basic version of DN seems similar to, but not exactly the same as, that of the brain. At least, the richness of the mechanisms in DN is not yet close to that of the brain. For example, the DN here does not use neuromodulators so it does not prefer any signals from receptors (e.g., sweet vs. bitter).

In conclusion, the above analysis and proofs have established the three theorems.

REFERENCES

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

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

[3] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston, MA, 2006.

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

[5] 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.

[6] J. Weng. Three theorems: Brain-like networks logically reason and optimally generalize. In *Proc. Int'l Joint Conference on Neural Networks*, pages 2983–2990, San Jose, CA, July 31 - August 5, 2011.

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