Paper

Why Have We Passed “Neural Networks Do Not Abstract Well”?

Juyang Weng*

Michigan State University, USA
**corresponding author*: weng@cse.msu.edu

Abstract

It has been argued that prior artificial networks do not abstract well. A Finite Automaton (FA) is a base net for many sophisticated probability-based systems of artificial intelligence, for state-based abstraction. However, an FA processes symbols, instead of images that the brain senses and produces (e.g., sensory images and effector images). This paper informally introduces recent advances along the line of a new type of, brain-anatomy inspired, neural networks —Developmental Networks (DNs). The new theoretical results discussed here include: (1) From any complex FA that demonstrates human knowledge through its sequence of the symbolic inputs-outputs, the Developmental Program (DP) of DN 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 errorfree. (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. These three theoretical results have also been supported by experimental results using real images and text of natural languages. Together, they seem to argue that the neural networks as a class of methods has passed “neural networks do not abstract well”.

1. Introduction

Studies on artificial neural networks (ANN) in the 1970’s and 1980’s (e.g., Fukushima 1975 [7], Grossberg 1975 [11], Hopfield 1982 [14], Rumelhart, McClelland & others 1986 [35], [27]) have been supported by a series of documented advantages of neural networks, including (1) non-algorithmic in task space, (2) uniform processors suited for massively parallel hardware, (3) fault tolerance, (4) numerical in signal space, and (5) feedforward networks are universal approximators of a certain class of static, multivariate functions [8], [15], [2]. ANNs have been also been identified by their network style of computation, called connectionist approaches.

Marvin Minsky 1991 [28] and others argued that symbolic models are logic and neat, but connectionist models are analogical and scruffy. Such criticisms have different ways of characterization, but we can use a simple sentence “neural networks do not abstract well.” Clearly, a lot of new work has been done for neural network since then (e.g., see a recent review [46]). However, this image of ANN has not fundamentally changed in the larger research community of intelligence modeling, natural intelligence and artificial intelligence. For example, at the David Rumelhart Memorial talk August 3, 2011 during the International Joint Conference on Neural Networks, Michael I. Jordan started with a statement that neural networks do not abstract well and he will talk about symbolic methods today. Jordan did some work on neural networks in the 1980s [19].

The term “connectionist” is misleading in distinguishing symbolic models and ANNs, since a probability based symbolic model is also a network whose representation is also distributed. Weng 2011 [46] proposed two classes, symbolic models and emergent models. By definition [46], an *emergent representation* emerges autonomously from system’s interactions with the *external* (outside the skull) world and the *internal world* (inside the skull) via the brain’s sensors and effectors without using the handcrafted (or gene-specified) content or the handcrafted boundaries for concepts about the extra-body concepts.

Many basic models of ANNs (e.g., Self-Organization Maps (SOM), feed-forward networks with gradient-based learning) use emergent representations but symbolic models use task-specific, handcrafted representations. A hybrid model, partially emergent and partially handcrafted, still belongs to the category of symbolic model. The brain seems to use emergent representations which emerge autonomously from learning experience, regulated by the genome in the nucleus of every cell (e.g., see Purve et al. 2004 [33] and Sur & Rubenstein 2005 [37]). All cells, other than the original zygote, in the body of a multi-cellular eukaryotic life are emergent from the zygote, whose emergence is regulated by the genome in the nucleus of every cell.

It seems to be the emergence of such network represent-tation – the process of autonomous development – that makes it hard to address the criticism “neural networks do not abstract well”. However, autonomous emergence of brain’s internal representation seems also the essential process for an animal brain to do what it does well as we know it.

In this article, I introduce a recent theory that maps a class of brain-inspired networks – Developmental Networks

(DNs) to any Finite Automaton (FA), a "common-denominator" model of all practical Symbolic Networks (SNs). From this FA, we can see what is meant by "abstraction". This mapping explains why such a new class of neural networks abstract at least as well as the corresponding SNs. This seems to indicate that our humans, collectively, have passed "neural networks do not abstract well."

The additional properties discussed in this paper include: (1) In contrast with an SN where the meanings of each node are *hand-selected* and boundaries between conceptual modules are *handcrafted*, there is a class of Generative DNs (GDNs) whose learning is fully *autonomous* inside each network, using the signals in the sensors and effectors. (2) In contrast with an SN whose expansion requires a manual re-design by the original human designer, the expansion (growth) of a GDN is fully autonomous inside the network, through observing an FA which collectively represents the human society's consistent knowledge. Such learning by the DN from the FA is incremental, immediate, and error-free. (3) The input symbols and output symbols of an FA are static, but the representations of input vectors and output vectors of a GDN are emergent from the natural environment (e.g., natural images, and natural arm motions). (4) The consideration of performance requires optimality for both types of models, symbolic (e.g., Markov models based on FA) and emergent (i.e., GDN). While the probability version of FA is limited by the static design of the input symbol set and the output symbol set, the outputs from the GDN at any time are optimal in the sense of maximum likelihood (ML), conditioned on the limited number of internal nodes and the limited amount and quality of the learning experience so far.

2. Two Types of Models

In this section, we discuss two types of models, symbolic and emergent.

2.1. Symbolic networks

Given a task, a human designer in Artificial Intelligence (AI) [21], [10] or Cognitive Science [1], [39] handcrafts a Symbolic Network (SN), using handpicked task-specific concepts as symbols. 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).

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.

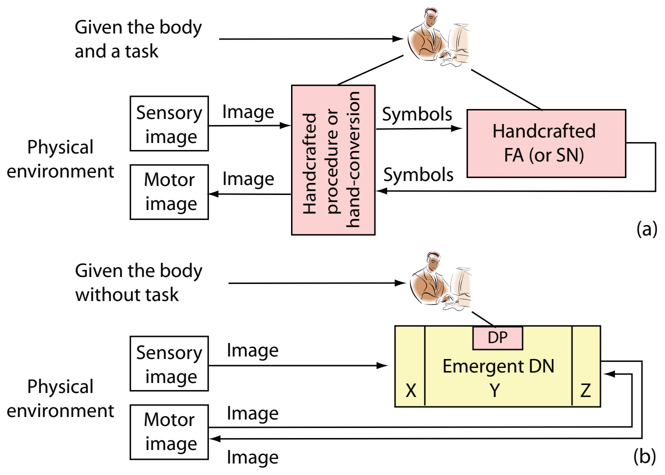

Figure 1. Comparison between a symbolic FA (or SN) and an emergent DN. (a) Given a task, an FA (or SN), symbolic, handcrafted by the human programmer using a static symbol set. (b) A DN, which incrementally learns the FA but takes sensory images directly and produces effector images directly. Without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

2.2. 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 [18] used neural networks to name SNs, and Tenenbaum et al. [40] used SNs to model the mind.

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 [36], [34] and recurrent [12], [49] networks, use images (numeric patterns) as representations. Recurrent networks can run continuously to take into account temporal 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 [30] 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. This paper explains that this hard problem now has a solution — DN. The internal representations of a DN emerge from a single cell (zygote) through experience, regulated by the Developmental Program (DP). An artificial DP is handcrafted by a human, to short cut extremely expensive evolution.

2.3. Innate problem-specific structure?

Neuroanatomical studies, surveyed by Felleman & Van Essen as early as 1991 [5] 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. Are such areas problem-specfiic?

Computationally, feed-forward connections serve to feed sensory features [31], [38] to motor area for generating behaviors. It has been reported that feed-backward connections can serve as class supervision [12], attention [3], [4], and storage of time information [49]. What developmental mechanisms enable the brain to establish feed-backward connections, as well as feed-forward connections? Are such developmental mechanisms problem-specific?

Gallistel reviewed [9]: “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 problem specific way if we analyze how a Generative DN (GDN) dynamically establishes 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.

3. Symbolic Networks

The brain’s spatial network seems to deal with general temporal context without any explicit component dedicated to time as argued by [26], [20], but its mechanisms are still largely elusive.

3.1. Finite automata

FA is amenable to understanding the brain’s way of temporal processing. An FA example is shown in Fig. 2(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 abstract. For example, our FA example treats “young cat” and “kitten” the same in its state output.

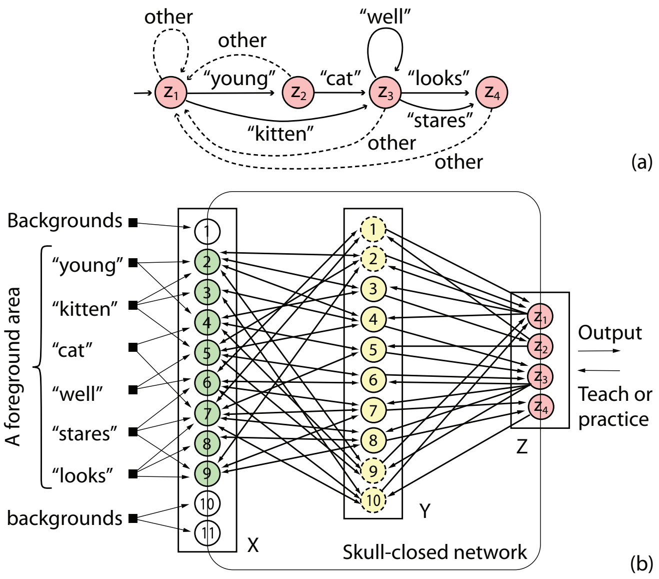

Figure 2. Conceptual correspondence between an Finite Automaton (FA) with the corresponding DN. (a) An FA, handcrafted and static. (b) A corresponding DN that simulates the FA. It was taught to produce the same input-out relations as the FA in (a). A symbol (e.g., z_2) in (a) corresponds to an image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

A finite automaton (FA) has been defined as a language acceptor in the traditional automata theory [13]. To model an agent, it is desirable to extend the definition of the FA as a language acceptor to an agent FA. An agent FA (AFA) M for a finite symbolic world is the same as a language acceptor FA, except that it outputs its current state, instead of an action (accept or not accept), associated with the state. In the following, an FA means an AFA by default.

The input space is denoted as $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_l\}$, where each σ_i representing an input symbol, whose meaning is only in the design document, not something that the FA is aware of. The set of states can be denoted as $Q = \{q_1, q_2, ..., q_n\}$. Like input symbols, the meanings of state q_i is also in the design document, but the FA is not “aware” the meanings. Fig. 2(a) gives a simple example of FA.

3.2. 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 the example in Fig. 2(a). The FA partitions all possible strings into 4 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.

3.3. 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 [41], [16], 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 a state-based symbolic agent.

3.4. 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.

3.5. 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.

3.6. Limitations of SN

In fact, an SN relies on humans to abstract from real world non-symbolic data, from sensors such as images, sounds, and effectors such as motor control signals. Therefore, the power of abstraction does not lie in FA, but in a human designer. An SN has the following major limitations:

(1) An SN is intractable for dealing with input symbols for real physical world. The human designer needs to handcraft Σ — sensory abstraction — to well represent all possible inputs to an acceptable precision. The number of inputs is intractably too large and handcrafting Σ is complex. If each input involves c concepts and each concept has v possible values, the potential number of input symbols is v^c, exponential in c. Suppose that we have $c = 22$ concepts and each concept has v=4 values (e.g., unknown, low, high, do-not-care), the number of possible input symbols is $v^c = 4^{22} = 16^{11}$, larger than the number of neurons in the brain. Here is an example of 23 extra-body concepts: name, type, horizontal location, vertical location, apparent scale, size, pitch orientation, yaw orientation, weight, material, electrical conductivity, shape, color, surface texture, surface reflectance, deformability, fragility, purpose, having life, edibility, usage, price, and owner.

(2) Likewise, an SN is intractable for dealing with output (state) symbols for real physical world. The human designer must handcraft Q — state abstraction — to well represent all possible output states to an acceptable precision. It is intractable for a human to examine many symbolic states for a large problem 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 complexity reason why symbolic agents are brittle.

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

4. Developmental Networks

Weng 2010 [43] discussed that a DN can simulate any FA.

4.1. 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. 2(b). The internal neurons in Y have bi-directional connection with both X and Z.

The DP for DNs is task-specific as suggested for the brain in [47] (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. 2(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.

4.2. 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 muscle neurons. Y potentially can also model any Brodmann area in the brain. According to many studies in detailed review by Felleman & Van Essen [5], each area Y connects in bi-directionally 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.

Although being convenient for studying infinitesimal changes (see, e.g., [17]), a continuous time model seems not very effective to explain network abstraction. Such a weakness is especially obvious for multiple neurons and brain-scale networks. I use a discrete time formulation, which is exact regardless how fast the network is temporally sampled (updated). Let the network update time interval be δ. The smaller the δ, the smaller the latency between a stimulus and the responsive action. The human brain seems to have a latency on the order of 100ms. In the following, δ is consider a unit, so we denote the time by integers $t = 0, 1, 2, ...$

The DN algorithm is as follows. 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 robotic agent or biologically regulated by the genome. Y is skull-closed inside the brain, not directly accessible by the external world after the birth.

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, and initialize the weight vector if the new neuron 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) effector 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-response 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.

4.3. Unified DN area function

It is desirable that each brain area uses the same area function f, which can develop area specific representation and generate area specific responses. Each area A has a weight vector $\mathbf{v} = (\mathbf{v}_b, \mathbf{v}_t)$. Its pre-response value is:

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

where $\dot{\mathbf{v}}$ is the unit vector of the normalized synaptic vector $\mathbf{v} = (\dot{\mathbf{v}}_b, \dot{\mathbf{v}}_t)$, and $\dot{\mathbf{p}}$ is the unit vector of the normalized input vector $\mathbf{p} = (\dot{\mathbf{b}}, \dot{\mathbf{t}})$. The inner product measures the degree of match between these two directions $\dot{\mathbf{v}}$ and $\dot{\mathbf{p}}$, because $r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t}) = \cos(\theta)$ where θ is the angle between two unit vectors $\dot{\mathbf{v}}$ and $\dot{\mathbf{p}}$. This enables a match between two vectors of different magnitudes (e.g., a weight vector from an object viewed indoor to match the same object when it is viewed outdoor). The pre-response value ranges in $[-1, 1]$.

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 bottom-up part and the top-down part separately 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 $y_j = 1$ (a pike) 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 $r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t})$.

4.4. *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:

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

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

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

where is important for the proof of the optimality of DN in Weng 2011 [44].

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.

4.5. *GDN area functions*

For simplicity, let us consider $k = 1$ for top-k competition.

Algorithm 1 (Y area function):

```
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 less than 0.9999, generate
   a $Y$ neuron using the input $\dot{\mathbf{p}}$ as the initial weight with
   age 0. The new $Y$ neuron is the winner for sure.
4) The winner neuron $j$ increments its age: $n_j \leftarrow n_j + 1$,
   fires with $y_j = 1$, and updates its synaptic vector,
   using Eq. (3).
5) All other neurons do not fire, $y_i = 0$, for all $i \ne j$,
   and do not advance their ages.
```

Algorithm 2 (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 imme-
       diately 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 $i \ne j$,
   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.

5. DN Abstraction

As one can expect, a handcrafted FA does not have any problem of convergence as it is statically handcrafted. However, how well can a DN abstract? Weng 2011 [45] provided the following three theorems, which provide properties about how well a DN can abstract, using FA as a basis. The proofs for the three theorems are available as a report [44], currently under review by a journal.

Since this paper is meant for a general reader of the INNS society journal, let us have an informal explanation of the three theorems and their importance.

5.1. *GDN learns any FA immediately and error-free*

Since FA is a "common denominator" model of many symbolic models (e.g., HMM, POMDP, Bayesian nets, semantic nets, belief nets, and graphical models), it is desirable to show that neural networks can incrementally learn any FA by autonomously organizing its emergent internal representations.

Frasconi et al. 1995 [6] programmed (not through learning) a feed-forward network to explicitly compute a statically given (as a batch) state transition of a fully given FA. They require a special coding of each state so that the Hamming distance is 1 between any source state and any target state. This means that transition to the same state (a loop) is impossible. If such a loop is necessary, they added a transition state to satisfy the requirement for unit Hamming distance. Omlin & Giles 1996 [32] programed (not through learning) a second-order network for computing a statically given (as a batch) state transition of an FA. By 2nd order, the neuronal input contains the sum of weighted multiplications (hence the 2nd order), between individual state nodes and individual input nodes.

The Theorem 1 in Weng 2011 [45], [44] established that goal above has been not only reached, but also two somewhat surprising properties — immediate and error-free. The text version of the Theorem 1 us 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 intended by the human FA designer.

The GDN 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. Conventional networks 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 seen 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.

5.2. GDN optimally performs while frozen

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. 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 the 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 GDN "lives" in the real world and generalizes optimally, going beyond the FA.

5.3. GDN optimally performs while learning

The following is the 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 additional detail about how a GDN simulates any FA.

6. Additional Details

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

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 $t = t_i$.

Let the motor area Z consist of several concept zones, $Z = (Z_1, Z_2, ..., Z_n)$. Within each concept zone only one neuron can fire. For example, each neuron represents a particular amount of contraction of a muscle or the degree of a joint. Neurons in each concept zone compete so that only the winner neuron can fire. If only one concept zone can fire at any time, the GDN can simulate any deterministic FA (DFA). If any number of concept zones can fire at any time, the GDN can simulate any nondeterministic FA (NFA).

If the pre-response value of the winner neuron is higher than a dynamic threshold, then the winner neuron fires. Otherwise, the winner neuron does not fire, like other loser neurons in the same concept zone. The value of the dynamic threshold in each concept zone can change according to the modulatory system (e.g., affected by punishment, reward, and novelty). In the proof of Theorem 1, the dynamic threshold is machine zero, which accounts for the amount of computer round-off noise.

Although each concept zone has only one or no neuron firing at any time, different concept zones can fire in parallel. For example, Z_1 represents the location and Z_2 represents the type of an object. Suppose that each concept zone has 4 positive values plus one value "do-not-care" (when all neurons in a concept zone do not fire), then n motor concepts amounts to 5^n possible actions, exponential in the number of concepts n. A symbolic model requires 5^n state symbols, but the motor area Z needs only $4n$ neurons.

7. Experiments with DN

Our DN had several versions of experimental embodiments, from networks for general object recognition from $360°$ views [22], to Where-What Networks that detect (in free viewing), recognize, find (given type or location), multiple objects from natural complex backgrounds [23], to Multi-layer In-place Learning Networks (MILN) that learn and process text of natural language [48] (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 [29]. Preliminary versions of the DN thinking process has been observed by [25], [24] for vision as the DN predicts while learning, and by [29] 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.

8. Discussions

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 DN model develops a "brain" internal mapping $X(t-1) \times Z(t-1) \mapsto X(t) \times Z(t)$ to explain the 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.

Using an SN, the human written symbolic text to define 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 the symbols, since these symbols are ineffective to sample an exponential number of sensory images for X and an exponential number of effector images for Z.

Conventional emergent networks, feed-forward and recurrent, were motivated by brain-like uniform, numeric, neuronal computations. However, their learning is slow, not exact, and they do not abstract well.

A GDN is also an emergent network, but is inspired more by characteristics of internal brain area Y as discussed in [43]. It learns any complex FA, DFA or NFA, 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. This means that GDN as an emergent network can abstract as well as any FA, logic and neat. This indicates that we have passed "neural networks do not abstract well".

The GDN theory is also a solution to many well known nonlinear system problems that are well known in electrical engineering and mathematics.

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 [42]). This array of properties indicates that GDN as a new kind of neural networks goes beyond FA and their probability variants SNs.

Much future work is needed along the line of GDN autonomous thinking, such as the creativity of GDN, in the presence of complex backgrounds that are not directly related to the current task or goal.

Acknowledgment

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

References

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

[2] G. Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.

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

[4] 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.

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

[6] P. Frasconi, M. Gori, M. Maggini, and G. Soda. Unified integration of explicit knowledge and learning by example in recurrent networks. *IEEE Trans. on Knowledge and Data Engineering*, 7(2):340–346, 1995.

[7] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136, 1975.

[8] K. I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Network*, 2(2):183–192, March 1989.

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

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

[11] S. Grossberg. Adaptive pattern classification and universal recoding: I. parallel and coding of neural feature detectors. *Biological Cybernetics*, 23:121–131, 1976.

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

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

[14] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79(8):2554–2558, 1982.

[15] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, November 1989.

[16] 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.

[17] E. M. Izhikevich. *Dynamical Systems in Neuroscience*. MIT Press, Cambridge, Massachusetts, 2007.

[18] 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.

[19] M. L. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. the eighth annual conference of the cognitive science society*, pages 531 – 546, Hillsdale, 1986.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] J. L. McClelland, D. E. Rumelhart, and The PDP Research Group, editors. *Parallel Distributed Processing*, volume 2. MIT Press, Cambridge, Massachusetts, 1986.

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

[29] 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.

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

[31] B. A. Olshausen 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.

[32] C. W. Omlin and C. L. Giles. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, 43(6):937–972, 1996.

[33] W. K. Purves, D. Sadava, G. H. Orians, and H. C. Heller. *Life: The Science of Biology*. Sinauer, Sunderland, MA, 7 edition, 2004.

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

[35] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, Massachusetts, 1986.

[36] 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.

[37] M. Sur and J. L. R. Rubenstein. Patterning and plasticity of the cerebral cortex. *Science*, 310:805–810, 2005.

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

[39] 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.

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

[45] J. Weng. Three theorems: Brain-like networks logically reason and optimally generalize. In *Proc. Int'l Joint Conference on Neural Networks*, pages +1–8, San Jose, CA, July 31 - August 5 2011.

[46] J. Weng. Symbolic models and emergent models: A review. *IEEE Trans. Autonomous Mental Development*, 3:+1–26, 2012. Accepted and to appear.

[47] 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.

[48] 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.

[49] 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.