On Machine Thinking

Xiang Wu[1], Zejia Zheng[2] and Juyang Weng[2,3,4]
[1]School of Automation, Nanjing University of Science and Technology, Nanjing, 210094, China
[2]Department of Computer Science and Engineering, [3]Cognitive Science Program, [4]Neuroscience Program
Michigan State University, East Lansing, MI, 48824 USA

***Abstract*—Artificial Intelligence (AI) has made much progress, but the existing paradigm for AI is still basically pattern recognition based on a human-handcrafted representation. An AI paradigm shift seems to be necessary to address the machine thinking question raised by Alan Turing over 90 years ago. As a necessary subject of our new conscious learning paradigm introduced 2020, this work deals with general-purpose machine thinking, with planning as a special case, based on new concepts of emergent Super-Turing Machines realized by our proposed neural network models—Developmental Networks (DNs) that have been mathematically proven for its optimally in the sense of Maximum Likelihood (ML). Experimental demonstrations are presented for simulated new mazes in disjoint tests.**

I. Introduction

Much progress has been made in Artificial Intelligence (AI), in both the symbolic school and the connectionist school [1]. However, fundamental difficulties of machine thinking persisted. They seem to root in a lack of understanding what natural thinking is beyond a machine classifier.

Alan Turing predicted [2], "I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted." Unfortunately, this situation did not happen by 1999.

Over 20 years later than Turing predicted, this paper constructively addresses this challenging question based on some major advances that we have made since Developmental Networks (DNs) [3]–[5] benefited from vast knowledge in AI and Natural Intelligence.

We proposed a new paradigm for AI called conscious learning, whose conference version appeared 2020 in [6], [7]. Conscious learning means that an AI machine must be partially conscious throughout its lifetime, since its simpler and partial consciousness at an earlier time is necessary for it to learn more sophisticated and more complete consciousness at a later time in life. However, consciousness is beyond the scope of this paper.

In this paper, we present a model of machine thinking which is a necessary condition for conscious learning. We further apply planning as an example of thinking to experimentally facilitate our understanding.

Although we have seen books [8], [9] that have "machine thinking" in their book titles, these publications did not explicitly define what they meant by their "machine thinking" and the methods in these books are still non-emergent, not satisfying the machine thinking definition here.

We propose a definition of machine thinking here, the first ever as far as we know, that is meant to approach animal-like thinking. In the definition below, we require that a thinking process should be of general-purpose in the sense that any tasks that an agent thinks about are not given by the birth time of the learning agent but must be learned postnatally.

Furthermore, we require the subject of machine thinking is of any nature, open-endedly learned from the living environment after the learner's birth.

In our model for this new definition of machine thinking, all the skills of thinking are learned through motor-supervised learning, during which a human teacher and the environment teaches not only concepts and actions but also covert actions, as well as under what context the learner should start to think and under what context it should stop thinking and act.

The remainder of the paper is organized as follows: Section II introduces a new theory for machine thinking based on the theory of Turing Machines (TMs). Section III briefly reviews the learning engine DN-2 and introduces its new features for machine thinking. The method of planning in DN-2 is described in Section IV. The experimental steps and results are reported in Section V. Section VI provides concluding remarks. Since the subject is new, we suggest that the readers read all the sections and review prior literature about DN.

II. First Theory of Machine Thinking

In order to discuss thinking, we must first build mechanisms on which machine thinking depends. We will first review TMs. Then, we extend Finite Automata (FA) to Agent FA (AFA). Further, we will introduce (1) AFA blocks and (2) emergent AFA, which lead to Emergent TMs (ETMs) learned by neural networks.

A. Review of TMs

It is worth noting that the framework of TMs is fundamentally more powerful than the concept of general functions that many neural network researchers are familiar with. For the paper to be self-contained, let us briefly review TMs.

A TM [10], [11] has a tape, a read-write head, and a control as a lookup table. The tape bears a string of symbols as input starting from the left most special symbol Δ as a blank mark. The tape also serves as a scratch paper during computation. When the machine halts, represented by a special state h, all the symbols on the tape form the output.

Mathematically a TM T is denoted as a 5-tuple: $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where Q is a set of symbolic states, Σ a set of

X. Wu and J. Weng, "On Machine Thinking" in Proc. International Joint Conference on Neural Networks, pp. 1-8, Shengzhen, China, July 18-22, 2021.

input symbols, Γ a set of tape symbols, $\Sigma \subseteq \Gamma$, q_0 the initial state, and δ the transition function (i.e., a program):

$$\delta : Q \times \Sigma \mapsto Q \times \Gamma \times M \tag{1}$$

where $M = \{L, R, S\}$ indicates the head motion, left L, right R, and stationary S. A TM program is a list of 5-tuples: $(q, \sigma, q', \gamma, m)$ that specifies δ. Each 5-tuple forms a transition:

$$(q, \sigma) \overset{\delta}{\mapsto} (q', \gamma, m) \tag{2}$$

Such a TM can carry out any complex sequence of computations, symbolic or numerical. The human designer must know the task and the meaning of every symbol before he designs a TM. Thus, such a TM is task-specific and so are all traditional AI programs as argued in Weng et al. 2001 [12].

B. Universal TMs

A universal TM [10], [11] is such that the input on the tape from a user has two parts, a program and the data for the program. Martin [11] gives a detailed encoding method E for a universal TM.

It is worth noting that a universal TM is still a TM. Its main difference from a special purpose TM is the universal task: simulating a user supplied TM program P, represented by a given 5-tuple encoding E for $E(P)$ in Eq. (2), applied to the user supplied data D using a symbolic representation $R(D)$. It is universal because it takes any program P from the user. But a traditional universal TM is still specific to the encoding E and representation R. Our thinking machine will remove these restrictions.

Universal TMs are not sufficient for us to deal with animal-like thinking as we will define, because the programmer of a task must require the purpose be given, including the purpose of the Universal TM (e.g., computer language compilers specific to E and R). To enable a machine to think as we will define, we must not require our machines to understand only a single encoding method E and representation R.

C. FAs as Controls of TMs

It was extremely challenging to bridge "scruffy" neural networks with the well-respected "clean" TM logic as complained by Marvin Minsky [13]. This problem was solved by Weng 2015 [4], by proving that the control of any TM is an AFA.

Motivated by biology, Weng [4] proposed to generalize the state Q, the tape alphabet Γ and head motion M to action $Q' = Q \times \Gamma \times M$, so that the transition function in Eq. (1) is represented as an FA with an extended domain $Q' \times \Sigma$:

$$\delta' : Q' \times \Sigma \mapsto Q'. \tag{3}$$

This extended Q' requires δ' to have some do-not-care components in the domain, but we will see below that do-not-care components for each transition are automatically learned without a need for handcrafting. He called this new kind of FA Agent FA (AFA), which outputs the state as part of action in Q'. This methodology innovation seems to have provided ways to break a long series of well-known impasses.

D. DNs as Controls of Emergent TM (ETM)

Weng 2015 [4] has further proven mathematically that a DN learns any TM optimally, conditioned on the incrementally learning mode, a teaching experience, and a limited number of hidden neurons. The key ideas include:

1) Instead of inputting symbols from a tape, input vectors from real-world sensors.
2) Instead of output symbols to the tape, output vectors to real-world effectors.
3) Instead of requiring a human to design an encoding E and write a program P, the DN learns directly from the physical world.
4) Attention to relevant subsectors in $\mathbf{x}$ input vectors and in $\mathbf{z}$ action vectors in the transition function of the TM is also automatically learned by the learner DN, using the power of LCA Hebbian learning [14].

Jointly, the above four points address fundamental limitations of human-in-the-loop symbolic representations in computer science, including their basis—TMs, and traditional AI systems.

Definition 1 (ETMs): Running at time t, $t = 0, 1, 2, ...$, an Emergent TM $M(t)$ learns from a 6-tuple (X, Y, Z, M, B, W) at time t, where X denotes the sensory space, Y the hidden space, Z the effector space, M the learner's memory space, B the space of learner's body that contains sensors and effectors, and W the world space, using an incremental network algorithm.

Let $V = X \times Y \times Z \times M$ be the time varying brain space without the body B and world W since they are sensed via X and Z. In real time t, starting from initial $\mathbf{v}(0) \in V$, update the time-varying network mapping function f:

$$\mathbf{v}(t+1) = f(\mathbf{v}(t)) \tag{4}$$

to simulate the network's lifelong developmental learning.

We define a context as $\mathbf{c} \in C$ where the space of contexts is defined as $C = X \times Y \times Z$. The network mapping function f recursively realizes the context transition using memory $\mathbf{m}$:

$$\mathbf{c}(t) \triangleq (\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t)) \overset{f,\mathbf{m}}{\longrightarrow} \mathbf{c}(t+1), t = 0, 1, 2, ... \tag{5}$$

These ETM transitions are very different from the symbolic transitions not only because of the use of emergent vectors but also the thinking-enabling internal representations in $\mathbf{y}$ that are absent from TMs.

Consider the four entities W, Z, Y, X at sample times $t, t = 0, 1, 2, ...$, with a sampling rate 30-60Hz:

TABLE I
UNFOLDING TIME IN THE DEVELOPMENT OF DN

Time sample index	0	1	2	...
Actable world W_z	$W_z(0)$	$W_z(1)$	$W_z(2)$	...
Motor Z	$Z(0)$	$Z(1)$	$Z(2)$	...
Skull-closed brain Y	$Y(0)$	$Y(1)$	$Y(2)$	...
Sensor X	$X(0)$	$X(1)$	$X(2)$	...
Sensible world W_x	$W_x(0)$	$W_x(1)$	$W_x(2)$	...

Unlike many neural networks that suffer from the so called wait-for-convergence problems, each column in the table can use only the information on the adjacent left column so that no iterations are allowed and the network responds in real-time.

The first row indexes the discrete sample times from inception $t = 0$. The second row denotes the actable world W_z, such as the body which acts on W_z. The third row is the motor Z, which has muscles to drive effectors, such as the arms, legs, and glottal. The fourth row is the skull-closed brain Y. The computation inside the brain Y must be fully autonomous postnatally, without intervention from any external teachers, as autonomous development [12]. The fifth row is the sensor X, such as cameras, microphones, and touch sensors (e.g., skin). The last row is the sensible world W_x, such as surfaces of objects that reflects light received by cameras. The actable world W_z is typically not exactly the same as the sensible world W_x, because where sensors sense from and where effectors act on can be different.

E. Super-TMs

The following extension [4] turns a TM into a super-TM.

Definition 2 (Super-TM Extension): The Super-TM Extension is defined as follows: (1) Extend the symbolic tape to the real world. (2) Input images $\mathbf{x}$'s from sensors, instead of symbols σ's; (3) Input and output the state, output, and head motion in the TM (q', γ', m) to be a motor vector $\mathbf{z}(t)$ in the effector space Z; (4) add the internal and emergent representation $\mathbf{y}$'s that are absent from TMs; (4) avoid handcrafting task-specific features in $\mathbf{x}, \mathbf{y}, \mathbf{z}$ and encoding E; (5) recursively and incrementally run the super-TM in real-time t according to Table I and Eqs. (4) and (5).

Note, point (3) in super-TMs avoids the impasse that brain states are not observable and not teachable, a major departure from traditional neural networks and reservoir computing.

Time unfolding: We treat X and Z in Table I as external because they can be "supervised" by the world as well as "self-supervised" by the network itself. We require the area Y to be internal and hidden—cannot be directly supervised by external teachers. Eq. (4) implies unfolding time $t = 0, 1, 2, ...$:

$$\begin{bmatrix} Z(0) \\ Y(0) \\ X(0) \end{bmatrix} \rightarrow \begin{bmatrix} Z(1) \\ Y(1) \\ X(1) \end{bmatrix} \rightarrow \begin{bmatrix} Z(2) \\ Y(2) \\ X(2) \end{bmatrix} \rightarrow ... \tag{6}$$

where $\rightarrow$ means neurons on the right use the input neurons on the left and compete to fire as explained below without iterations. Namely, by unfolding time, the spatially recurrent DN becomes non-recurrent in a time-unfolded and time-sampled DN. A DN is ML-optimal and has a low time complexity $O(1)$ suited for real-time computation as proven in [4].

F. Sub-TMs

An AFA is the control of any TM, including Universal TM. Because the teaching is in the motor of the agent, namely, the representation is natural, there is no need for the encoder E, so that even the physical world can serve as a teacher when the learner interacts with the physical world using its sensors and effectors as *discovery learning* through machine thinking.

We use a dashed block to enclose a sub-AFA, as a sub-TM. A sub-TM can call another sub-TM, like how a main program calls a procedure.

G. Emergent AFA

Because $\mathbf{x} \in X$ and $\mathbf{z} \in Z$ are no longer symbols, there are too many samples to be listed in an emergent AFA diagram. Therefore, we use subspace as labels instead. Suppose $X = (X_1, X_2)$ where X_1 and X_2 are two subspaces, each representing a different set of sensors or sensory subspaces, an emergent AFA block T_0 may use only a subspace $X_1 \subset X$ as input and X_2 is treated as "don't care".

Likewise, suppose $Z = (Z_1, Z_2)$ where Z_1 and Z_2 are two subspaces, each representing a different set of effectors or motoric subspaces, an emergent AFA block T_0 may use only a subspace $Z_1 \subset X$ as motor input and output and Z_2 is treated as "don't care".

Weng [4] proved the ML-optimality of AFA in every DN for any learned TM, including UTM, without local minima because every update in Eq. (4) is optimal, conditioned on incremental learning, the learning experience and the available number of hidden neurons.

H. Covert vs. Overt

We will model that thinking involves learning of covert actions, like the rehearsal of vocal track actions without activating the glottis.

Definition 3 (Covert and overt): Suppose a motor vector $\mathbf{z} \in Z$ consists of a series of sub-vectors $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n)$. Then, to enable each sub-vector $\mathbf{z}_i$ to think, $\mathbf{z}_i$ is associated with a pair of covert-overt neurons denoted as (c_i, o_i). If and only if $c_i(t) = 1$, $o_i = 0$, $\mathbf{z}_i(t)$ is covert, corresponding to thinking for sub-vector $\mathbf{z}_i$, not displayed to the actable world W_z in Table. I. Otherwise, $c_i(t) = 0$, $o_i = 1$, sub-vector $\mathbf{z}_i$ is being displayed to W_z.

I. Definition of Thinking

We need the definition below since all animals develop.

Definition 4 (Developmental learning): Developmental learning is a process of lifetime interactions, after inception time, between a skull-closed, task-nonspecific, and incrementally learning network and the extra-skull environment that consists of the extra-skull body of the agent and the extra-body environment that may include teachers.

Markov Decision Processes (MDPs) do not satisfy this definition because an MDP requires skull-open batch clustering. A human specifies the correspondence between MDP and the extra-body environment during this batch clustering.

Definition 5 (Animal-like thinking): Animal-like thinking is a process of developmental learning by an ETM inside a neural network during its life time $t \geq 0$, that has the following five properties.

1) **general:** for any open-ended skills/tasks learned from the environment postnatally;
2) **sensorimotor:** taught through the sensory and motoric areas that interact with the external environment;

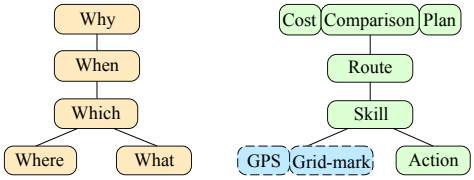

Fig. 1. The hierarchical concepts for machine thoughts. The correspondence between the Where-What Network concept architectures (left side) and the learning context of the DN agent (right side), all represented by the Z motor.

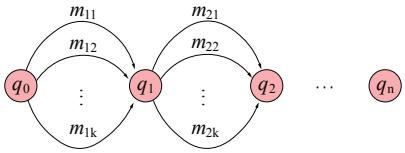

Fig. 2. Abstraction using stage sub-TMs. The i-th stage is abstracted by a sub-TM with starting state q_i, $i = 0, 1, ...n$. Each sub-TM has k sequences to learn, each denoted by an arrow.

3) **covert:** not always immediately displayed to the environment, like "not voiced";
4) **displayable:** may be overtly displayed by corresponding effectors; and
5) **autonomous:** covert, overt, or mix thereof, all using previously learned skills.

J. Thinking with Abstractions

As thinking is taught through sensorimotor interactions, what about "higher" concepts that are involved in thinking? Intuitively, we need to teach why, when, what, which, where and so on to think about as illustrated in Fig. 1. The teacher just needs to supervise the corresponding covert c_i to fire at 1 or 0 so that whether the display of the corresponding actions is also taught.

We also consider how motor-autonomous learning give rise to higher concepts. The learner needs to experience the real-world results of $c_i(t)$, depending on firing or not and experience the outcome and probably receive punishments and rewards. We have the following theorem.

Theorem 1 (Multi-Stage Abstraction): As illustrated in Fig. 2, suppose a global task has n spatial or temporal underlying sub-TMs as stages and each sub-TM requires k sequences to learn to become error-free and a brute-force search without teaching such n sub-TMs will have an exponential number of k^n sequences to deal with. Then, teaching n sub-TMs moderate kn sequences. The uniqueness of the starting state (context for DN) of each sub-TM guarantees success in dealing with all k^n possible sequences.

Proof: The proof will be available in the journal version; the same is true for other proofs below. ■

We will use sub-TMs below to discuss how to learn a long task.

K. Chaining of Thoughts as Sub-ETMs

When the training of thoughts takes place in an emergent TM, the training corresponds to a temporal sequence of network activities, sampled at the frame rate in Table. I.

The simplest form of chaining is the chaining of overt sensorimotor skills q_i sequentially, each of which is identified by *time*, to realize task transfer [15]. In this work about machine thinking, we extend the chaining for time in [15] to chaining by *sets of motor concepts*, not just time.

Definition 6 (Context Chaining): A chaining of brain activities, that includes covert thoughts and overt displays of actions, is a chaining of emergent sub-TMs based on activated contexts, as a result of competitions among learned contexts.

For successful chaining, it is important for the teacher to teach, or the learner to learn or discover, a hierarchy of concepts, like those in Fig. 1.

Theorem 2 (Winner context): At each lifetime t, multiple $k(t)$ sub-TMs are available, represented by their learned initial contexts $\{\mathbf{c}_i, (t) \mid , i = 1, 2, ...k(t)\}$, as defined in Eq. (5). Concepts taught before t influence what context wins at time t.

A richer set of concepts taught, as illustrated in Fig 1, provides a better uniqueness for the DN to automatically find and call applicable sub-TMs based on winner context, to accumulate sharpened calling statistics, and to generalize when it deals with a drastically different global setting with different sensory inputs.

L. Grand Emergent UTM

Theorem 3 (GEUTM): If each pair of program and data is taught from the environment through interactions with a DN, each pair corresponds to an emergent AFA inside the DN as the control of an Emergent Sub-TM. A Grand Emergent Universal TM (GEUTM) inside the DN is incrementally learned combining all such Emergent Sub-TMs.

It is interesting to see how humans are taught by peers and the physical world as GEUTM-like. Obviously, teaching a pair to a simulated or real robot to become an Emergent Sub-TM is intuitive because the real sensors and real effectors are used. The process of successful teaching is typically harder than writing a computer program into a symbolic UTM because the real world is cluttered and the teaching process must teach how to attend and generalize in the cluttered world.

Theorem 4 (General-Purpose Thinking): An emergent GUTM in DN is taught to think in any task if the thinking task has been taught in the form of sensors and effectors of the DN.

It is worth noting that there is no guarantee that the taught program can always generalize perfectly in a cluttered world. The failures of generalization are important for the learner to observe, to question the learned programs (or rules) and to discover better rules (or new physical laws).

M. Thinking Skills in Planning

We use Fig. 3 as an example of teaching DN for planning in a simulation environment. As a process of scaffolding, the teacher in the example has designed a series of visual settings. The agent is first taught to learn walking forward as TM-F (in Fig. 3 (a)). Then, it is taught how to turn right as TM-R (in

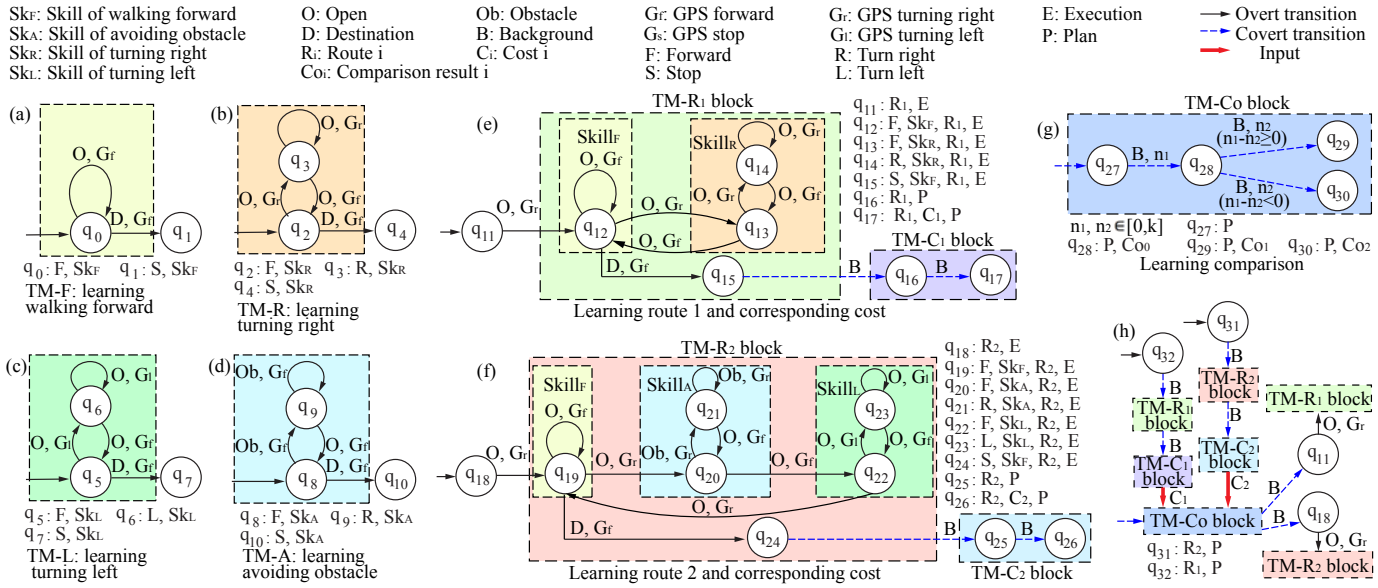

Fig. 3. The eight TMs (a) to (h) that the DN agent is taught in simulated environments as TM transitions. An inward arrow not from a state is the initial state of the TM. When TMs are learned in a long navigation sequence, the last state of the previous TM is taught as the initial state of the following TM so that each TM does not need to re-learn—scaffolding.

Fig. 3 (b)), using skills learned from TM-F. Next, it learns how to turn left as TM-L (in Fig. 3 (c)). To teach how to avoid obstacles, the agent is taught TM-A (in Fig. 3 (d)), using learned skills from TM-F and TM-R. TM-F, TM-R, TM-L, and TM-A are called local behaviors. It is worth noting since each TM corresponds to a subsequence of a lifetime, each TM should contain a sequence to return to its initial or "default" state.

After the agent has learned local behaviors based on what it senses, the teacher lets it explore different routes from the starting point to a destination as a new task. During this process, the agent learns how to link local behaviors to execute the long task. It is taught with the distance and how to count the distance (the number of tiles). These explorations are learned with overt actions. Such explorations of different routes result in the learning of different sub-TMs, TM-R_1 and TM-R_2 in Fig. 3 (e) and (f). For each route, the agent learns a different reward according to the cost of distance, e.g., TM-C_1 and TM-C_2 in Fig. 3 (e) and (f).

Then the teacher teaches covert actions as thinking as the comparison, without actually traversing the two paths. This corresponds to the sub-TM TM-Co in Fig. 3 (g). Finally, the teacher teaches a long thinking process, that involves covertly running TM-R_1, TM-R_1, TM-C_1, TM-C_1, and TM-Co, which results in TM-R_1 as the chosen plan. The entire planning process is shown in Fig. 3 (h).

Because we have the general-purpose model known as the GEUTM that learn each sub-TM as a sub-program, the entire learning process for local behaviors, global behaviors, and planning for a global behavior are only an application of the GEUTM, where there is no encoding necessary because the representation of each sub-TM is natural in the motor area of the DN. The human teacher only designs the instruction lessons, but he does not get involved in the programming of the emergent TMs that the DN autonomously learned inside. The environment taught such large TMs while DN automatically integrates them into a grand TM.

III. Developmental Network-2

This seems to be the first publication for any neural networks, as far as we know including DNs, to address machine thinking. Marvin Minsky [13] complained that neural networks are scruffy. The above theory seems to have solved the scruffiness, if a DN learns the recursive mapping in Table I on the fly optimally without local minima. Unlike error-backprop algorithms, DN is without PSUTS.

DN-2 is the latest general-purpose learning framework in the DN family. In DN-1, the allocation of neurons in each hidden area is handcrafted by the designer. In DN-2, several biology-inspired mechanisms are designed to automatically allocate neuronal resources and generate dynamic hierarchical inner representations during learning, relieving the designer from handcrafting any concept hierarchy.

A. Overview of DN-2

As shown in Table I, the basic structure of DN-2 contains three areas—sensory area X, hidden area Y, and motor area Z. We should mention that the Y area is "skull-closed"—unsupervised after birth. Neurons in the X area directly receive the sensory inputs. The Z was taught using concept zones, receiving supervisions or generate concepts/actions. In the version taught here, each Z neuron in a concept zone corresponds to a distinct value of the concept, in the mind of the teacher. The neurons in Y area learn context-input features by connecting with neurons in X, Y, and Z areas, based on unsupervised Hebbian learning. During each neuron's first

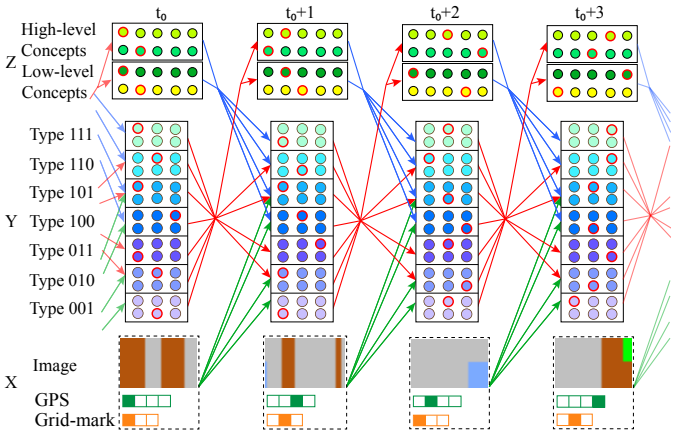

Fig. 4. The sequential learning procedure of DN-2 realizing Table 1. In the Y area, the binary code xyz differentiates seven connection types. The Y neurons incrementally grow when the existing ones of the same type cannot represent the context $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ well. In the Z area, neurons fire like the Y areas but they may be supervised to fire. The neurons with red outlines denote firing neurons.

firing, the neuron memorizes the input directly, according to the Hebbian learning for the ML principle.

To speed up the connection patterning, each hidden neuron has an initial default connections type with respect to X, Y and Z, represented by a binary code xyz ("1" connected, "0" not connected).

A synaptic maintenance mechanism is applied to automatically shape the connection patterns of these Y-to-Y connections according to automatically discover neuron-to-neuron statistical dependences.

B. Working of DN-2

To realize the mapping in Eq. (6), DN-2 incrementally generates optimal clusters in the internal feature space during its lifetime learning. Each type of Y neurons or Z neurons in each concept zone have weights matching the corresponding domain inputs and compute the responses. These neurons form the local inhibition zones and only top-k neurons within each local inhibition zone fire and update their weights.

At time t, the Y and Z neurons receive the global context $(\mathbf{x}(t-1), \mathbf{y}(t-1), \mathbf{z}(t-1))$ at the last frame as inputs.

If there is binary supervision from the outside environment to Z, all the supervised Z neurons fire at 1. Without Z supervision, the Z neurons complete to fire like all the hidden Y neurons. Only firing neurons update their post-synaptic weights as the incremental average of the pre-synaptic input vectors. This procedure is shown in Fig. 4 which realizes Eq. (6) incrementally through the lifetime.

As discussed in section II-D, as a GEUTM to execute any learned sub-TM while the lifetime grand TM incrementally learn and integrate.

Therefore, unlike a symbolic TM that must be handcrafted, a DN-2 was taught using emergent and natural representation in the form of its sensors and effectors.

C. Algorithm of DN-2

We list the main DN-2 algorithm as follows.

Input areas: X and Z; Output areas: Z; Hidden area: Y.

1) At time $t = 0$, inception. Initialize the X, Y and Z areas.
2) For time $t = 1, 2, ...$, repeat the following steps forever (executing steps 2a, 2b in parallel, before step 2c):
   a) All Y neurons compute in parallel:
   $$(\mathbf{y}', M_y') = f_y(\mathbf{c}_y, M_y) \tag{7}$$
   where $\mathbf{c}_y = (\mathbf{x}(t-1), \mathbf{y}(t-1), \mathbf{z}(t-1))$, and f_y is the Y area function.
   b) Supervise $\mathbf{z}(t)$ if the teacher likes. Otherwise, Z neurons compute the response vector $\mathbf{z}(t)$ and update memory M_z' in parallel:
   $$(\mathbf{z}', M_z') = f_z(\mathbf{c}_y, M_z) \tag{8}$$
   where f_z is the Z area function and $\mathbf{c}_z = \mathbf{y}(t-1)$.
   c) Replace asynchronously: $(\mathbf{y}, M_y, \mathbf{z}, M) \leftarrow (\mathbf{y}', M_y', \mathbf{z}', M')$. Supervise input $\mathbf{x}(t)$.

For further detail of DN-2, the reader is referred to [16].

IV. Planning as Examples of DN-2 Thinking

Planning is a good example domain to study how the process of thinking takes place. A plan is a procedure for a sequence of future actions, as a result of thinking based on skills learned in the past. The process of configuring a plan typically includes a comparison of alternatives for the same goal.

Specifically, this work investigates how the agent performs planning in autonomous navigation tasks. We use a variety of simulated mazes since they allow a precise record of performance. Although real-world settings are more realistic, natural settings do not allow a precise record of performance like simulations. We also discuss how DN-2's new mechanisms help the agent to chain the learned skills and further plan in the thinking mode for a long-term goal.

A. Review of the Planning Work

The framework of Markov Decision Processes has been often used by a symbolic model for learning and planning. The Reinforcement Learning (RL) methods can also be used in planning for acquiring and exploiting knowledge through interaction with the environment. A time-discount Q-learning model is often used to deal with the explosion of expected value when the length of future time grows. But Q-learning models are handcrafted and symbolic in $Q(s, a)$, which we do not follow because such simbolic models are task-specific (e.g., DNs are not just for planning) and so highly brittle.

In our experiments, the distance can be considered a punishment, but our DN is not handcrafted with a world model. Thus, the DN model is not rigidly coded with any time-discount model. Namely, value perception is the result of learning, not part of the DN program. We do not model how values are processed by a DN so that autonomous thinking for general purpose is now possible.

B. Thinking with New Contexts

When motor-imposed supervision for a new context, $C = (X, Y, Z)$ is not available in a certain time frame, DN-2 generates "casually" using network interpolation.

Theorem 5 (Optimal Thinking): For each new context $C(t)$ not observed for this life, the DN-2 thinks optimally in the sense of maximum likelihood by generalization according to its network equation Eq. (4), conditioned on the restriction of incremental learning architecture, the learning experience including the environment-and-body and the given neuronal resources. Different initial random weights of DN-2 result in the same network and performance.

As we will see in our experiments below, when the optimal DN-2 experiences new environments that are different from any of the taught ones, its optimal thought may not rigidly follow the mind of the teacher, as a result of its creative thought process based on the numerical interpolation of dot-products and the ML principle.

C. Skill Chaining to Learn Higher-Level Concepts

High-level concepts can be learned by chaining the low-level concepts together following Theorem 5.

V. Experiments

A. Motivation of the Experiments

Our goals of the experiments are twofold. First, to experimentally realize the model of thinking. Second, to measure the performance of the DN-2 learning thinking through scaffolding. We as teachers for the general-purpose DN, designed multiple simulated environments so that the learner's behavior can be exactly measured and recorded. Experiments on real robots are our future goals. Although real robots do not allow precise measurement of performance as we did earlier for skill transfer [15], future visual observations probably would have to be used to replace the exact recording here.

Although GPS signals are now widely available, driverless cars remain challenging because local collision avoidance or detours are typically inconsistent with GPS signals. Machine thinking is indeed needed and challenging in such situations.

Suppose the teacher teachers a simulated agent to make turns and avoid obstacles using its vision. Can the teacher only supervise occasionally for long navigation in new settings? If the teacher supervises the simulated robot to think in one setting, can the robot think in new settings? The TM based model enables high-level and low-level combined learning by neural networks. Such a new kind of neural network learning, without task-specific programming, seems to be the first time to have the potential to address such extremely challenging questions. No other methods can be compared with yet because this is the first time to address these questions.

In the experiments, a series of simulated mazes are designed by the teachers for an simulated agent equipped with DN-2 to learn local skills, skill chaining, cost computation, cost comparison, plan selection and plan execution. During the simulation, the DN-2 agent is trained to learn these contents through multiple-stage learning. During later learning, the teacher only supervises occasionally while the situation is new. The code and data are publicly available at https://github.com/wux0080/Phoneme-synthesis-and-maze-navigation-based-on-DN-2.

TABLE II
Teaching Rules in Maze

Item	Instructions
1	Follow the GPS except obstacle/red traffic light is in vision.
2	Turn to the open grid If there is an obstacle within 20 pixels.
3	If the traffic light is red, stop until it changes to green.
4	Keep track of distance when walking through adjacent grids.
5	Go back to the starting point after reaches the destination.

TABLE III
Z Concept Zones

Concept zones	Level	Number of neurons	Contained concepts
Z_{action}	Low	4	Forward, Left, Right, and Stop
Z_{skill}	Low	8	Skill 1-7 and none
Z_{route}	High	3	route 1-3, and "go back"
Z_{cost}	High	18	0-16 and none
$Z_{\text{comparison}}$	High	4	less, equal, more, and none
Z_{plan}	High	2	covert and overt

The simulated robot has a simulated color camera, as shown in Fig. 5. A simulated GPS as a sensor tells it the long-term direction, which can be blocked by obstacles (blue) and traffic signals (green or red).

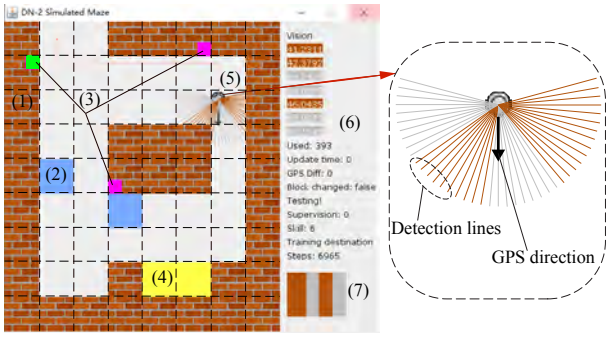

Fig. 5. The GUI of one of many simulated mazes. (1) wall grids; (2) obstacle grids; (3) traffic light grids; (4) destination grids with reward. (5) learning agent. More details of the agent are shown on the right side. (6) the GUI information panel for training and testing. (7) a 2D image seen by the learner, gray if the object is too far.

The teaching rules for the learning agent in the simulated environment are listed in Table II.

The network has six Z concept zones, listed in Table III, in the motor area which receive supervision at different stages.

500 Y neurons are allocated in the Y area of DN-2. When taught low-level concepts, only type 101 neurons are released to learn actions and navigation skills. When taught high-level concepts, type 011 neurons mainly grow to learn individual routes and corresponding costs, comparison and planning.

We randomly generate maze environments and put the agent into random locations of the maze to finish navigation in a specific scene.

At the first local behavior stage, we train the agent seven kinds of basic navigation skills (see Fig. 6).

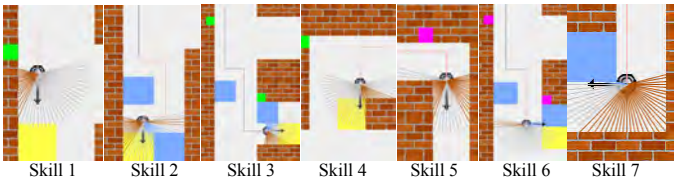

Fig. 6. Navigation skills taught to the learning agent in the simulated maze environment. Skill 1: move forward when the next grid is open. Skill 2: avoid obstacle and correct facing direction. Skill 3: move through the narrow path. Skill 4: turn right when GPS indicates right. Skill 5: turn left when GPS indicates left. Skill 6: avoid the obstacle when turning left. Skill 7: avoid the obstacle when turning right. Other skills: move forward when the traffic light is green, and stop when the traffic light is red.

TABLE IV
EXPERIMENT RESULT

Environments	Skill chaining		Planning results
	Route 1	Route 2	
Maze 1	10/10	10/10	Route 1 (10/10)
Maze 2	10/10	10/10	Route 2 (10/10)
Maze 3	10/10	10/10	Route 1 (10/10)
Success rate	100%	100%	100%

The training schedule includes the following lessons: (1) Learning routes and corresponding costs. (2) Learning comparison. (3) Perform long-term planning.

We designed three basic maze environments (see Fig.7) to train and test the DN-2 agents (10 times for each maze).

As shown in the first three columns in Table IV, the DN-2 agents successfully chained the low-level navigation skills together to reach the destination with 100% accuracy in four different maze environments which totally contain nine different routes.

We further trained the agent sequentially to learn the corresponding cost of a specific route and compare the costs of different routes in "thinking mode". In tests, we used three maze environments to make the agent autonomously select the optimal route based on the costs (10 times for each maze).

As shown in the last column of Table IV and Fig.7, the agent successfully chose the optimal routes (marked as solid lines) from the other one (marked as dashed lines) in all tests. In some testing environments, the agent chose to turn more flexibly to pass the obstacles based on the image inputs (circled by red ovals in mazes of Fig 7). This shows to some extent the generalization of DN-2 after learning.

The experimental results demonstrated that DN-2 can associate specific cost with the corresponding route and emphasizing the association by recalling the information in "thinking" mode. And after planning, DN-2 can obtain the knowledge of cost comparison and integrate all learned capabilities and autonomously select the optimal route to reach the destination.

VI. CONCLUSIONS AND DISCUSSIONS

As a necessary condition of the conscious learning paradigm, we presented a model for general-purpose machine thinking. Experimentally, using a variety of simulated mazes. The DN was able to apply its learned skills to new maze settings that it has not observed. We plan to look into non-sensorimotor teaching in the future.

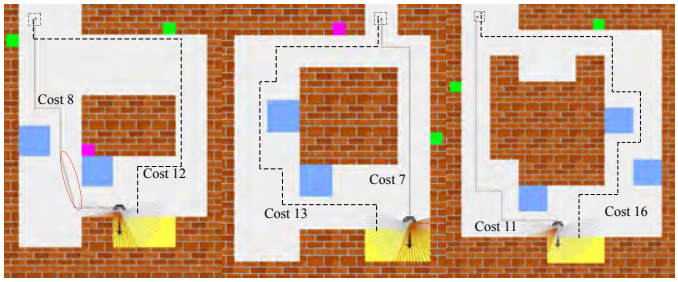

Fig. 7. Three navigation route selection results. The selected route is marked by solid lines, while others by dash lines. The range of random starting locations of the agent is marked by dashed rectangles near the entrance of each maze.

REFERENCES

[1] J. Weng, "Symbolic models and emergent models: A review," *IEEE Trans. Autonomous Mental Development*, vol. 4, no. 1, pp. 29–53, 2012.
[2] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, October 1950.
[3] J. Weng and M. D. Luciw, "Brain-inspired concept networks: Learning concepts from cluttered scenes," *IEEE Intelligent Systems Magazine*, vol. 29, no. 6, pp. 14–22, 2014.
[4] J. Weng, "Brain as an emergent finite automaton: A theory and three theorems," *International Journal of Intelligent Science*, vol. 5, no. 2, pp. 112–131, 2015.
[5] J. Weng, "Autonomous programming for general purposes: Theory," *International Journal of Huamnoid Robotics*, vol. 17, pp. 1–36, August 2020.
[6] J. Weng, "Conscious intelligence requires developmental autonomous programming for general purposes," in *Proc. IEEE International Conference on Development and Learning and Epigenetic Robotics*, (Valparaiso, Chile), pp. 1–7, Oct. 26-27 2020.
[7] J. Weng, "Machines develop consciousness through autonomous programming for general purposes (APFGP)," in *Springer Lecture Notes on Communication, Proc. of IJCAI Workshop on Human Brain and Artificial Intelligence*, (Yokohama, Japan), pp. 1–17, Jan. 7 2021.
[8] P. N. Johnson-Laird, *Human and machine thinking*. Hillsdale, New Jersey: Lawrence Erlbaum, 1993.
[9] R. J. Sternberg, ed., *Thinking and Problem Solving*. San Diego, California: Academic Press, 1994. Chapters 1 and 2.
[10] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Boston, MA: Addison-Wesley, 2006.
[11] J. C. Martin, *Introduction to Languages and the Theory of Computation*. Boston, MA: McGraw Hill, 3rd ed., 2003.
[12] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals," *Science*, vol. 291, no. 5504, pp. 599–600, 2001.
[13] M. Minsky, "Logical versus analogical or symbolic versus connectionist or neat versus scruffy," *AI Magazine*, vol. 12, no. 2, pp. 34–51, 1991.
[14] J. Weng and M. Luciw, "Dually optimal neuronal layers: Lobe component analysis," *IEEE Trans. Autonomous Mental Development*, vol. 1, no. 1, pp. 68–85, 2009.
[15] Y. Zhang and J. Weng, "Task transfer by a developmental robot," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 226–248, 2007.
[16] X. Wu and J. Weng, "Neuron-wise inhibition zones and auditory experiments," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9581–9590, 2019.