Emergent Turing Machines and Operating Systems for Brain-Like Auto-Programming for General Purposes

Juyang Weng,[1,2,3,6] **Zejia Zheng,**[1,2,6] **Xiang Wu,**[4] **Juan Castro-Garcia,**[1] **S. Zhu**[1,6]**, Q. Guo,**[5] **X. F. Wu**[5]

[1]Department of Computer Science and Engineering, [2]Cognitive Science Program, [3]Neuroscience Program
Michigan State University, East Lansing, MI, 48824 USA
[4]School of Automation, Nanjing University of Science and Technology, Nanjing, 210094, China
[5]Department of Electronic Engineering, Fudan University, Shanghai, 200433, China
[6]GENISAMA LLC, 4460 Alderwood Dr. Okemos, Michigan 48864 USA

Abstract

In Artificial Intelligence (AI) there is a wide gap between the symbolic school and the connectionist school, but in both schools, different task-purposes require different learning methods. Different sensory modalities, such as video, sound, and text, also require different learning methods. In a larger scale, there is further a wide gap between brain scene and computer science — human brains automatically generate programs for general purposes but computers still cannot do so. Our Developmental Networks (DN) here are meant for bridging the gap in AI by further bridging the larger gap between brain science and computer science. The DN learning engine automatically emerges Turing Machine logic into its neural network. The AI Machine Learning (AIML) Contest 2016 is the first machine-learning contest that used task-nonspecific and modality-nonspecific learning engines. It used the DN engine for a variety of tasks and modalities. The organizers and contestants independently verified a prototype of DN for vision, audition, and natural languages acquisition and understanding (English and French co-acquisition). The Auto-programming Operating Systems (AOS) developed by GENISAMA LLC is meant to facilitate developers to train for many different applications using the same DN engine by following a standard for the body setting file.

Introduction

Since Cresceptron published 1991, the first deep learning neural network for natural images of clustered 3D world (Weng, Ahuja, and Huang 1997), a wide variety of deep neural networks have increased the domains of demonstration (LeCun, Bengio, and Hinton 2015; Schmidhuber 2015; Jordan and Mitchell 2015). Neural networks numeric computation and learning allow interpolative approximation in high-dimensional parameter spaces that seems to be more uniform for large AI problems than traditional symbolic methods.

However, there is a misconception that confuses symbolic networks with neural networks. Note that every symbolic algorithm may correspond to a (non-neural) symbolic network. Let us consider camera as an example but the ideas apply to any sensing modality: A robot takes an image from its 3D cluttered world but its context is to find and recognize an object of interest: Jim. In symbolic nets, each input line is *monolithic*: It must by itself represent a (human handcrafted) monolithic object of interest (e.g., Jim, David, a tree, an object, or an event). For this reason, many early networks are symbolic networks, e.g., (Carpenter, Grossberg, and Rosen 1991) and (Siegelmann 1995).

In neural nets, each input line is *non-monolithic*: In our vision example, each line is only a pixel in the image of a cluttered 3D world which contains many objects. A pixel by itself does not have any monolithic symbolic meaning of an object of interest (e.g., Jim). The entire input image is *contaminated* (e.g., pixels that contain information of Jim are contaminated by many factors such as lighting, viewing angle, and where Jim is), *partial* (e.g., only visible part of Jim), and *cluttered* (e.g., in each input image there are many objects irrelevant to Jim). As we know, all humans and robots must take the lines of sensors/motors that are non-monolithic, because they interact with physics (i.e., physical worlds).

This non-monolithic concept is useful for us to understand biological brains and to approach strong AI (i.e., AI that are not focused on a narrowly defined task). Furthermore, intractable challenges in three AI areas — vision, audition, and natural language understanding — call for strong AI. The direction of Autonomous Mental Development (Weng et al. 2001) showed that the learning algorithm must also be task-nonspecific.

Biologically, the modality-general architecture of the Developmental Network (DN) was motivated by the following studies that demonstrated surprising plasticity of the brain early in life:

1. Cells in the V1 area of the cat selectively respond to the left eye, the right eye, and both eyes in the normal kitten; but if one eye is stitched from birth, they respond only to the other eye (Wiesel and Hubel 1965). It appears that from which eye each V1 neuron receives sensory input is plastic.

2. An amputation of auditory pathway early in animal life enabled the *visual* projections to grow into the *auditory* pathway. The *auditory* cortex emerged *visual* representations and performed *visual* capabilities (VonMelchner, Pallas, and Sur 2000). Namely, not just which eye, but the modality of a pathway is plastic.

3. In humans born blind, the visual cortex is recruited by audition and touch (Voss 2013). It appears that not only non-human animals, human visual areas were recruited by other modalities.

The DN takes advantage the generality of muscle "states". Skills that are declarable (Sun, Slusarz, and Terry 2005) (e.g., through English) can be declared verbally through muscles in the vocal tract. Non-declarable skills (e.g., riding a bike) are also executed through muscles. These numerical networks do not necessarily have symbols inside.

Theory Brief

Next, we briefly outline the theory due to the space limit. Symbolically, consider the above "states" as symbolic "states" of a Finite Automaton (FA). The transition function of an FA takes the current state $q \in Q$ and the current input $\sigma \in \Sigma$ and maps (q, σ) to the next state $q' \in Q$, but for convenience we write the pair of q and σ vertically as a vector:

$$\begin{bmatrix} q \\ \sigma \end{bmatrix} \rightarrow \begin{bmatrix} q' \\ \sigma' \end{bmatrix}. \tag{1}$$

where σ' is the next input, or the predicted next input when the input is absent. All symbols here are monolithic. This new representation of FA, q at top and σ at bottom, is useful for our discussion below about internal representation in Y.

Why FA? In computer science, the Universal Turing Machines were widely recognized as a model for any general-purpose digital computers (Von Neumann computers). However, (Weng 2015) recently proved that the control of any Turing Machine is an FA. The main idea in the proof (Weng 2015) was to allow each state in FA to include also actions — write a symbol onto the TM tape and move the read-write head of the TM tape. Thus, learning any FA is sufficient for any Universal Turing Machines and sufficient for general purposes.

Below, we bridge the above gap in AI. We enable DN to do abstraction but without a human in the loop using symbolic representations.

We consider the "skull" to be the boundary of DN (brain). Inside the skull is "internal" and outside the skull is "external". Running at discrete times $t = 0, 1, 2, 3, 4, ...$, the sensory area X of DN takes input patterns $\mathbf{x}$ and the muscles area Z of DN takes state pattern $\mathbf{z}$. State/action $\mathbf{z}$ is taught by the environment but very often self-supervised.

To bridge the above gap mathematically, we define *nonmonolithic* vectors $\mathbf{x}$ and $\mathbf{z}$ that correspond to symbols σ and q, respectively.

However, the new notation here is that by a *nonmonolithic* vector $\mathbf{v}$ that corresponds to a symbol a, we mean that the vector $\mathbf{v}$ is *contaminated, partial, and cluttered* in representing the monolithic symbol a. Therefore, in all the notation below, it is essential for the corresponding neural network to automatically *attend* only the relevant and partial parts of components among the *cluttered* vector $\mathbf{v}$ and to effectively deal with the *contamination* and the *partial* nature. Unsupervised clusters of the Lobe Component Analysis (LCA) (Weng and Luciw 2009) with different receptive fields and different weight patterns are necessary to provide candidates or competitors for attention (Weng 2015).

The hidden Y area takes input from vector $(\mathbf{z}, \mathbf{x})$ to produce an internal response vector $\mathbf{y}$ which represents the best match of $(\mathbf{z}, \mathbf{x})$ with one of many internally stored patterns of $(\mathbf{z}, \mathbf{x})$:

The winner-take-all learning rule, which is highly nonlinear and simulates parallel lateral inhibition in the internal (hidden) area Y of DN is sufficient to prove in (Weng 2015) that a DN that has sufficient hidden neurons learns any Turing Machine (TM) perfectly, immediately, and error-free.

The n neurons in Y give a response vector $\mathbf{y} = (y_1, y_2, ...y_n)$ of n neurons in which only the best-matched neuron fires at value 1 and all other neurons do not fire giving value 0:

$$y_j = \begin{cases} 1 & \text{if } j = \underset{1 \le i \le n}{\operatorname{argmax}} \{ f(\mathbf{t}_i, \mathbf{z}, \mathbf{b}_i, \mathbf{x}) \} \\ 0 & \text{otherwise} \end{cases} \qquad j = 1, 2, ...n, \tag{2}$$

where f is a function that measures the similarity between the top-down weight vector $\mathbf{t}_i$ and the top-down input vector $\mathbf{z}$ as well as the similarity between the bottom-up weight vector $\mathbf{b}_i$ and the bottom-up input vector $\mathbf{x}$. The value of similarity is the inner product of their length-normalized versions (Weng 2015). Corresponding to FA, both the top-down weight and the bottom-up weight must match well for f to give a high value as inner product.

The response vector $\mathbf{y}$ in the hidden Y area is then used by Z and X areas to predict the next $\mathbf{z}$ and $\mathbf{x}$ respectively in discrete time:

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{x} \end{bmatrix} \rightarrow \mathbf{y} \rightarrow \begin{bmatrix} \mathbf{z}' \\ \mathbf{x}' \end{bmatrix} \tag{3}$$

where $\rightarrow$ denotes the update on the left side using the left side as input. The first $\rightarrow$ above is highly nonlinear because of the top-1 competition so that only one Y neuron fires (i.e., exactly one component in binary $\mathbf{y}$ is 1). The second $\rightarrow$ consists of simply links from the single firing Y neurons to all firing neurons on the right side. (Weng 2015) has shown that each weight of the firing post-synaptic Z neuron in $\mathbf{z}'$ incrementally accumulates the probability for the Z neuron to fire, conditioned on its firing, supervised by the external teacher or self-taught during autonomous practice. The same is true for predicting components in $\mathbf{x}'$.

As Eq. (3) is independent of task and independent of modality, it applies to any practical task and any modality. Furthermore, it is able to integrate and share tasks and modalities. It can also transfer task-to-task and modality-to-modality skills.

Like the transition function of the FA in Eq. (1), each prediction of $\mathbf{z}'$ in Eq. (3) is called a *transition*. but now in real-valued vector, without any symbols. The same $\mathbf{y}$ can also be used to predict the binary (or real-valued) $\mathbf{x}' \in X$ in Eq. (3). The quality of prediction of $(\mathbf{z}', \mathbf{x}')$ depends on how state $\mathbf{z}$ abstracts the external world sensed by $\mathbf{x}$. The more mature the DN is in its "lifetime"' learning, the better its predictions. In the following, we use two paragraphs to explain each arrow.

Here, we would like to give a top-level mathematical ideas of the theory but the rigorous proof is available in (Weng 2015). There are two arrows in Eq. (3). First, from sensory input $\mathbf{x}$ and motor state $\mathbf{z}$ to hidden response $\mathbf{y}$. Second, from the hidden response $\mathbf{y}$ to predict the state $\mathbf{z}$ and input $\mathbf{x}$. Although these two arrows compute in parallel without waiting for the other, Eq. (3) presents them in serial just to facilitate our understanding.

The first arrow in Eq. (3): When $k \geq 1$ in top-k competition in the hidden area Y, we have a committee of $k > 1$ members who cast votes, giving relatively more reliable voting results if the committee members are all experts in this $(\mathbf{z}, \mathbf{x})$ case. In other words, if Y have few neurons, $k > 1$ is not a good choice. Now, consider $k = 1$ in top-k competition, namely only one best-matched member in the voting committee who is voting. All Y neurons compete by matching their weights to match the state-input vector pair: $(\mathbf{z}, \mathbf{x})$. In other words, in the high-dimensional space of $(\mathbf{z}, \mathbf{x})$, we have many Y neurons as unsupervised clusters learned from incremental clustering from a huge number (potentially unbounded) of observed samples. The firing Y neuron's weight vector is the nearest neighbor of the current $(\mathbf{z}, \mathbf{x})$, among many other Y neurons. When the winner Y neuron fires at value 1, a very important abstraction has taken place: From the highly distributed representation in the form of $(\mathbf{z}, \mathbf{x})$ into a highly concentrated representation in the form of a single value 1 of the winner Y neuron. The $k > 1$ committee case is similar, where abstraction is concentrated on a relatively small k number of voting members.

The second arrow in Eq. (3): We discuss the prediction for the state vector $\mathbf{z}$ only, as the prediction for the input vector $\mathbf{x}$ is similar. The prediction is for not only the vector $\mathbf{z}$ that corresponds to a symbol q' in Eq. (1), but the result gives slightly imperfect new vectors that are similar but not the same! Namely, emergence of new actions that are not possible by using only a static set of handcrafted symbols! Each firing neuron in $\mathbf{z}'$, regardless it is supervised by the teacher to fire or the agent self-explores to fire, "wakes up" and starts to update its weights for $\mathbf{y}$. Other non-firing neurons still "sleep" and do not update their weights. The incremental update of LCA (Weng and Luciw 2009) amounts to incremental computation of the probability of each firing Y neuron conditioned for this firing Z neuron! Beautiful math.

Furthermore, the above vector formalization in Eq. (3) is simple but very powerful in practice. The pattern in Z can represent the binary pattern of any abstract concept — context, state, muscles, action, intent, object type, object group, object relation. However, as far as DN is concerned, they mean the same— a firing pattern of the Z area!

Namely, unified numerical processing-and-prediction in DN amounts to any abstract concepts above. In symbolic representations, it is a human to handcraft every abstract concept as a symbol; but DN does not have a human in the "skull". it simply learns, processes, and generates vectors. In the eyes of a human outside the "skull", the DN gets smarter and smarter.

Consider learning. Suppose human society together with mother nature as a teacher is a huge and complex TM, including a Universal Turing Machine. Because its control is

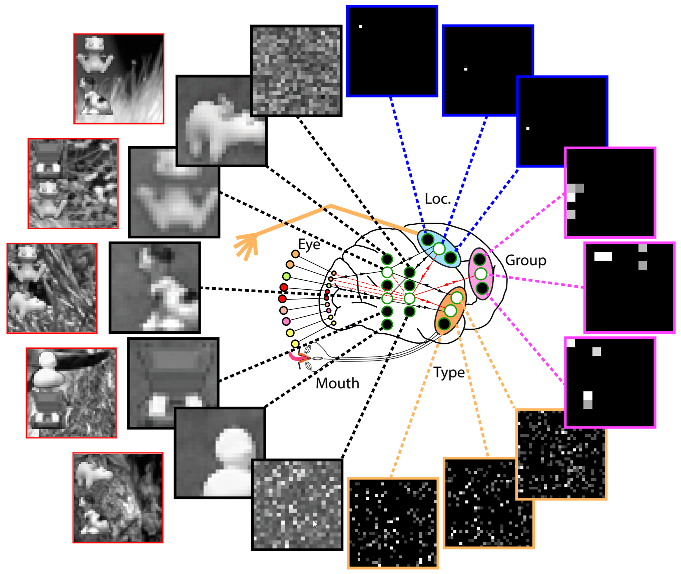

Figure 1: The distributed memory and responses inside a DN that has learned a concept hierarchy — the lower "location" concept (blue) and "type" concept(brown); and the higher "group" concept (purple) .

an FA, represented by a huge FA transition table having r rows and c columns. At each time t, $t = 1, 2, ...$, only the winner Y neuron fires at response value 1 and incrementally updates its weight vector $(\mathbf{z}_i, \mathbf{x}_i)$ as the vector average of attended part of $(\mathbf{z}, \mathbf{x})$. This is called the incremental Hebbian learning rule. Then, the i-th Y neuron memorizes perfectly the i-th distinct input pair $(\mathbf{z}, \mathbf{x})$ observed in life, because the teacher TM has no errors. When the teacher has errors the DN is optimal in the sense of maximum likelihood, as proved in (Weng 2015).

The Z area was taught the next (binary) response vector $\mathbf{z}'$ using the same incremental Hebbian learning rule.

Suppose there are at most rc transitions in TM. The FA needs at most rc hidden neurons to perfectly learns the TM. Because a universal TM is a TM, a DN with sufficient hidden neurons can learn any universal TM perfectly, immediately and error-free.

Because any universal TM is a general purpose computer, so is the corresponding DN. However, DN is emergent (i.e., skull closed) by automatically learning from the real physical world. In contrast, a universal TM is human handcrafted. Therefore, we have established in theory that a DN can automatically learn from the real physical world to gradually become a general purpose machine that auto-programs.

Fig. 1 fives a schematic illustration about how the brain or DN learned *higher* concept "group" from rules that are based on *lower* invariant concepts — type and location. Each input $\mathbf{x}$ is concrete, but motor $\mathbf{z}$ becomes increasingly abstract through learning. Potentially such DNs are capable of auto-programming for general purposes after they become mature through "lifetime" activities.

Cross-modality transfer is one of the goals of our future experimental studies using the same DN learning engine. The information in the motor area Z can be contributed from different inputs from X or different modalities from X. For example, visual "apple" in X_1 and auditory "apple" in X_2 all are bi-directionally linked with spoken "apple" in Z. Thus, visual "apple" in X_1 invokes spoken "apple" in Z which then invokes auditory "apple" in X_2, or vice versa. The WWN-1 embodiment of DN has demonstrated such bidirectional predictions but not yet cross-modality transfer.

Experiments

In AIML Contest 2016, we let a DN to take one of three sensory modalities while it learns in "lifetime": We let the $\mathbf{x}$ be the image at each time instance and $\mathbf{z}$ be landmark location-and-type and action of navigation, the DN became a vision-guided navigation machine. We let $\mathbf{x}$ be the frame of firing pattern of hair cells in cochlea at each time instance and $\mathbf{z}$ be the dense stages and the sparse type of sounds, the DN became a auditory-recognizer machine. We let $\mathbf{x}$ be a time frame of vector of word (either English or French) and $\mathbf{z}$ be the language kind and meaning of each sentence context, the DN became a bilingual language understander.

Learning vision based autonomous navigation: The vision task is autonomous navigation on the MSU campus, where GPS signals are often missing, not accurate enough, and will lead to failures without a sufficient visual capability using a single video camera. Fig. 2 provides an overview of the extensiveness of the training, regular training, and blindfolded testing sessions. [1] The inputs to the DN were from the same mobile phone that performs computation. They include the current image from the monocular camera, the current desirable direction from the Google Map API and the Google Directions API. If the teacher imposes the state in Z, this is treated as the supervised state. Otherwise, the DN outputs its predicted state from Z. The DN learned to attend critical visual information in the current image (e.g., scene type, road features, landmarks, and obstacles) depending on the context of desired direction and the context state. Each state from DN includes heading direction or stop, the location of the attention, and the type of object to be detected (which detects a landmark), and the scale of attention (global or local), all represented as binary patterns. None is a symbol. The dataset used in the AIML Contest contained 2109 gray-scale images of 72×128 pixels that have been converted down to 38×38 pixels for the DN to learn and test. The impressive performance will be reported elsewhere due to the limited space.

Audition from a "lifelong" cochlear sequence: For the audition modality, each input image to X is the pattern that simulates the output from an array of hair cells in the cochlea. We model the cochlea in the following way. The cells in the base of the cochlea correspond to filters with a high pass band. The cells in the top correspond to filters with a low pass band. At the same height, cells have different phase shifts. Potentially, such a cochlear model could deal with music and other natural sound, more general than the popular Mel Frequency Cepstral Coefficients (MFCCs) that are mainly for human speech processing. The performance will be reported elsewhere due to the limited space.

[1]Youtube video at `https://www.youtube.com/watch?v=4cc9xk0TaxE`.

Natural languages from a "lifelong" word sequence: As far as we know, this seems to be the first work that deals with language acquisition in a bilingual environment, largely because the DN learns directly from emergent patterns, both in word input and in action input (supervision), instead of static symbols.

The input to X is a 12-bit binary pattern, each represents a word, which potentially can represent 2^{12} words using binary patterns. The system was taught 1,862 English and French sentences from (Scriven, Amiot-Cadey, and Collins 2011), using $2,338$ unique words (case sensitive). As an example of the sentences: English: "Christine used to wait for me every evening at the exit." French: "Christine m'attendait tours les soirs à la sortie."

The Z area was taught two concepts: language type (English, French, and language neutral, e.g., a number or name) represented by 3 neurons (top-1 firing), and the language-independent meanings as meaning states. The performance will be discussed elsewhere due to the space limit.

AOS

Based on the theories, methods, devices, and experiments explained above, this section presents a new kind of OS — Auto-Programming Operating Systems (AOS).

Why AOS? The purposes of AOS are twofold:

First, make strong AI. Weak AI is AI that is for a narrowly defined task. Strong AI is AI that is meant to learn and perform many different tasks in the natural world. Weak AI is not only limited in the scope of the task that the machine executes, but also brittle if the task is a real-world task, such as self-driving in the natural world.

Second, make machines work and learn more like a human. For example, suppose one prints a file that includes 10 pages. A traditional printer is not able to abort the printing task once the task has been started but the toner is run out in the middle, or the long file being printed turned out to be a wrong one. A human can change his goal in real time on-the-fly according to a new situation, but a traditional printer cannot. The AOS abstracts all sensors and effectors as real-time sensors and effectors, so that the DN can change the task at hand within a fraction of second. Changing the goal on the fly is not only important for aborting a task, but also for adjusting sub-goals within a task. For example, how to adjust the pitch, volume, duration and other sound characteristics depending on how the machine likes what it hears from its singing.

By definition, an agent is something that senses and acts. Inside the agent are three types of resources: sensors, effectors, and computational resources. For convenience of the three-type classification, anything that is not sensor or effector is considered computational resources. Therefore, e.g., batteries can be considered part of computational resources.

Because AOS is for auto-programming for tasks without a static scope, we must not assume any task concept. However, AOS should abstract the agent body so that a DN can plug into any computer and start to "live" and learn.

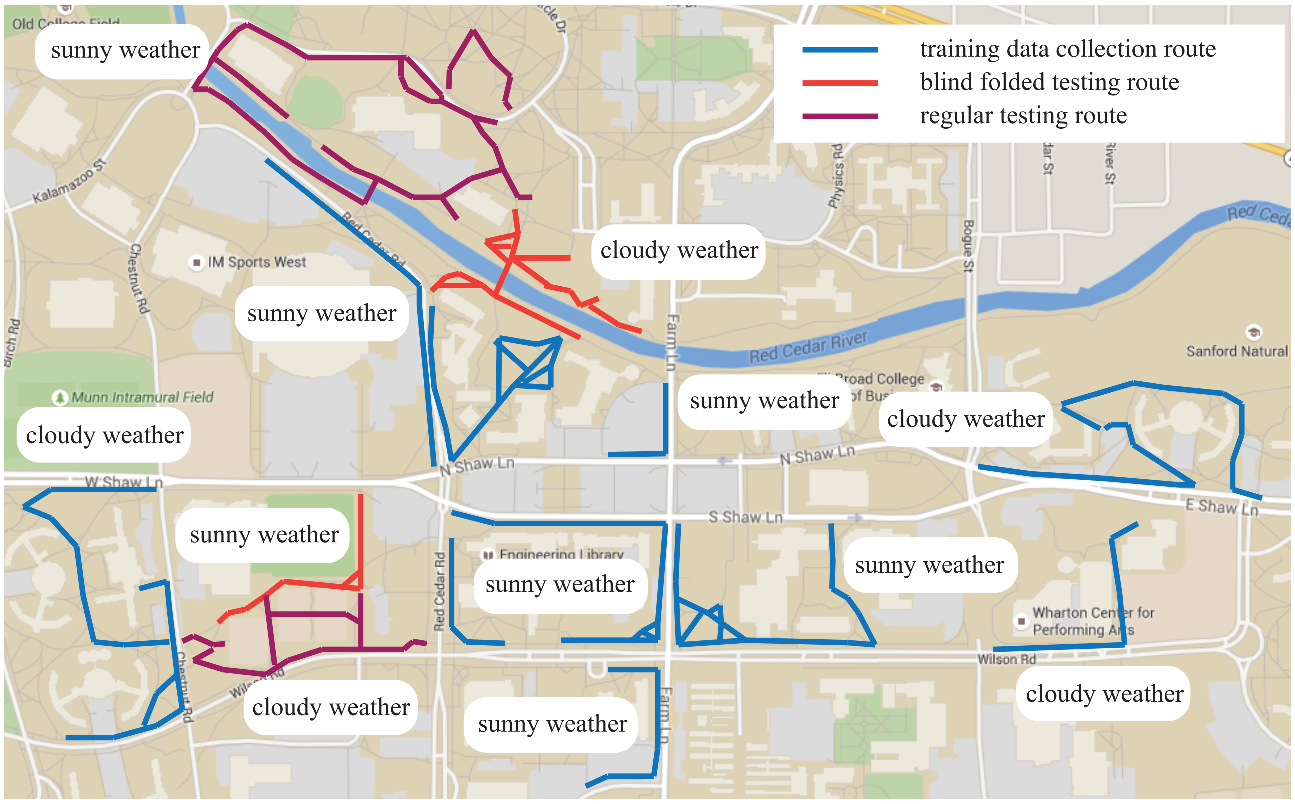

Figure 2: Training, regular testing, and blind-folded testing sessions conducted on the campus of Michigan State University (MSU), under different times of day and different natural lighting conditions with unpredictable shadows from trees, light posts and buildings. The DN automatically learned how to detecting reliable features (e.g., landmarks) while disregard unreliable features (e.g., shadows) based on statistics represented by neuronal weights. Disjoint testing sessions were conducted along paths that the machine has not learned.

The new method and device of AOS deal with the following issues:

1. Convert every input device to a unified sensor with a set of parameters (e.g., the number of pixels).
2. Convert every output device to a unified effector with a set of parameters (e.g., the number of possible values).
3. Convert all computational resources to unified neurons and their connections.
4. Provide a mapping for each change in the physical sensors, effectors, and computational resource in the body so that the trained DN can continue to learn on the new body.

Body changes may take place at different lifetimes of a (machine) brain — the Developmental Network. For example, a machine brain successfully trained on a factory body is copied into many identical machine brains (DNs) each of which is then uploaded to a different machine body by a consumer. A version 1 body is upgraded to version 2 body, and therefore, the machine brain must run on the new body.

Like a human being, the learning of a strong AI system must go through a process of learning many tasks — from simple to complex — so that skills learned for simple tasks assist the learning for complex tasks. For example, learning to stand steady can assist learning walking without falling. Learning walking is useful for learning running. Sometimes, the skills learned for complex tasks can also assist the learning of simpler tasks. For instance, in mathematics, skills in learning derivatives can also assist learning limits (e.g., L'Hospital's Rule).

Traditional OS An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. In the context of operating systems, input devices and output devices are called peripherals. Traditionally, an operating system (e.g., Unix, DOS, Mac OS, Windows, iOS, Android) treat each peripheral differently using a different driver.

A traditional OS treats a keyboard and a camera using two very different drivers because the keyboard and the camera are two very different input devices.

A traditional OS treats a printer and a speaker (or other effectors such as one that controls the steering wheel of a car) using very different drivers because the printer and the speaker are two very different output devices.

A traditional OS treats computer resources very differently, such as memory, disk, CPU and GPU.

AOS The main purpose of AOS is to provide a unified standard for any Developmental Network (DN) that auto-programs for general purposes. Although each computer has different hardware and OS, AOS abstracts all hardware and OS into a single standard in order for any AOS-complaint DN to plug in and start learning as a GENISAMA TM.

Shown in Fig. 3, an AOS is built on top of a conventional OS such as Unix, Android, iOS, and Windows which provides some basic functions about the resources, such as recording, playing back, programming, and search.

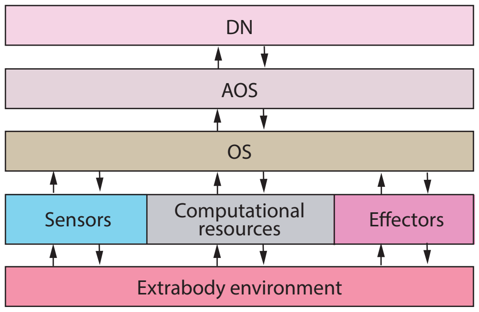

Figure 3: The relation among DN, AOS, traditional OS, hardware (computational resources, sensors and effectors), and the physical extra-body environment. The body includes DN, AOS, traditional OS, and hardware.

The theories, methods and experiments above have given detailed examples about how an AOS converts three types of resources — sensors, effectors, and computational resources — into unified sensors, unified effectors, and unified neurons, respectively. Because there is an open-ended variety of physical sensors, effectors, and computational resources, it is desirable and sufficient to give the following principles of AOS.

AOS unified sensors: AOS provides AOS standards and sample methods to unify all current and future sensors. Examples of sensors include: video camera (real-time image sensors), microphones (real-time sound sensors), touch screens (real-time touch sensors), lasers, radars, sonars (real-time distance sensors), keyboards (real-time finger-touch sensors for body symbols).

Each instance of connected sensor provides an abstract sensor, called AOS body sensor. At each sampled time, each sensor provides a body-sensed pattern, represented as a numerical image (typically 2D for a gray-tone camera or 3D for a color camera) where each pixel corresponds to a body-location of a receptor in the sensor and the intensity of each pixel corresponds to the firing value of that receptor. Two cameras are treated as two sensors whose field of views have partial overlap in the 3D physical world. Pixels in the binocular areas are considerably correlated between the left camera and right camera. It is desirable for DN to automatically form connections in a coarse-to-fine manner through lifetime using synaptic maintenance so that neurons automatically find their locations in the artificial retinas and whether it is a binocular neuron, a left-monocular neuron, or a right-monocular neuron. Other differences between the DN sampling rate and the sensor sampling rate should be treated the same way by AOS.

Because the update rate of a DN (e.g., 30Hz) is considerably lower than the sampling rate of a microphone (e.g., 44,000Hz), a sampled body-sensed image from a microphone integrates the between-frame (e.g., 33ms) spatiotemporal information of all hair cells in a cochlea. Namely, an AOS image from a microphone and an AOS image from a camera are basically the same in data format, called AOS sensory image: e.g., both are images provided at 30 times per second. The major differences between them are the number of pixels and the nature of the physical properties (i.e., sound vs. light).

AOS unified effectors: AOS provides AOS standards and sample methods to unify all current and future sensors. The data format of an AOS effector is also an image, called AOS effector image.

Just like AOS sensors are *body sensors*, all AOS effectors are *body effectors*. By *body effector*, we mean that each component in the pattern of an AOS effector corresponds to a body muscle on the body, instead of value of a concept in the extra-body world. The purpose of this requirement is to avoid handcrafted extra-body concept in the representation of AOS effectors. However, we allow the environment to teach any patterns of world concepts through sensors and effectors. For example, we allow the environment to teach a component in the steering wheel effector that corresponds to a particular speed value or a particular angle value of the steering wheel (i.e., simulating the arm that turns the steering wheel).

The sampling rate difference between DN and an effector (e.g., a loud speaker) is treated in a way similar to sensors. The image generated for a speaker contains information to drive the speaker for the inter-frame time (e.g., 33ms from a 30Hz DN). AOS body speaker is not a conventional static text-to-speech synthesizer. Because it simulates muscles, the DN can generate different sounds (e.g., singing) according to its spontaneous intents or goals.

AOS unified computational resources — neurons: AOS provides AOS standards and sample methods to unify all current and future computational resources. The basis of the standards and sample methods are the architecture of DNs, explained above. All computational resources serve neurons as the basic computing elements. Each neuron requires memory (registers, RAM, disk, etc.) to store its dynamic weights and its dynamic connections with other neurons. Each neuron also requires memory to store its dynamic age and growth rates. Each neuron requires computing resources (CPU, GPU, FPGA, etc.) to carry out its computation for its current response and its current values of neural transmitters through excitatory neural transmitter, inhibitory neural transmitter, 5-HT, DA, ACh, NE (see (Weng 2012)).

The memory hierarchy (e.g., the register-RAM-disk hierarchy) in a traditional OS meets changes here because it is possible that every weight of all neurons would be used to compute the real-time neuronal competition, regardless whether the neuron itself fire after the computation of competition. Such brain-like parallel computations require a high speed of computation and data transmission by hardware (e.g., CPU, GPU, FPGA, dedicated neuronal circuits, and data bus) and a large amount of fast memory.

Hardware changes: Let us consider difference of hardware. Such a difference may occur between a training machine in the robot school run by a factory and a customer machine. This can be treated basically the same as a body change within the robot school: The partially learned DN is downloaded from the old body, recompiled with the AOS on the new body, and continues to run (i.e., learn and perform) the old DN on the new body, in a way similar to a human who changed to a new pair of eye glasses. A minimal degradation of performance may be observed like the human who got a new pair of eye glasses.

The hardware changes modeled by AOS include resolution change (increase or decrease), a range change (increase or decrease), a depth change (e.g., from black-and-white to color), or a combination thereof, from the prior pattern to a new pattern in the AOS sensory image or the AOS effector image. AOS specifies a mapping standard and sample methods between the old pattern to the new pattern:

Uniform sensors and effectors: For a sensor, the density of receptors is uniform across the sensing array. For an effector, the muscle neurons are uniform. There is no need to calibrate the new camera, because the DN is able to adapt, like how human eyes learn to adapt to a new pair of glasses. Suppose that the new pattern has doubled the number of pixels, in both row and column, from the old pattern. The AOS handles this case by initially connecting one of every 2×2 pixels in the new pattern to the corresponding pixel in the old pattern. The growth of Y neurons in DN will gradually spread its neurons over the entire new pattern. A similar principle applies if the new pattern reduced the resolution: Linearly spread the pixels in the new camera evenly across the old input pattern by skipping an old connection every n pixels, where $n = 2$ if the resolution is reduced by 2 in the direction from the old pattern to new pattern.

Nonlinear sensors and effectors: The sensing array of the sensor may have non-uniform receptors (e.g.. like the retina where the density of cone and rod receptors in the fovea is higher than the periphery). For an effector, the muscle neurons are non-uniform. Like the uniform case, there is no need to calibrate. Simply connect pixels of new pattern uniformly across the old pattern. The DN will automatically assign neuronal resource according to the density distribution in the new pattern.

Change in computational resources: AOS distributes the additional resource, or strip the resource, uniformly across the entire Y zone. DN will automatically blend in new neurons or fill the holes left out by deleted neurons.

Conclusions

Strong AI goals seem no longer hopeless, based on the theoretical brief here and the DN applicability to vision, audition, and natural language understanding in AIML Contest 2016. The AOS provides a new kind of OS that enables auto-programming for general purposes, hopefully greatly reducing the cost of the development and brittleness of AI applications.

Contributions: JW: major original ideas, theories, and algorithms; ZZ: vision; XW: audition; JC: natural languages; SZ: AOS; QG and XFW: knowledge hierarchy and Fig. 1.

Acknowledgements: The authors would like to thank Z. Ji, M. Luciw, M. Solgi, X Song, Y. Wang, W. Zhang, Q. Guo, and Y. Wu for their contributions to the experimental embodiments of DN: WWN-1 through WWN-9.

References

[Carpenter, Grossberg, and Rosen 1991] Carpenter, G.; Grossberg, S.; and Rosen, D. B. 1991. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks* 4:759–771.

[Jordan and Mitchell 2015] Jordan, M. I., and Mitchell, T. M. 2015. Machine learning: Trends, perspectives, and prospects. *Science* 349:255–260.

[LeCun, Bengio, and Hinton 2015] LeCun, Y.; Bengio, L.; and Hinton, G. 2015. Deep learning. *Nature* 521:436–444.

[Schmidhuber 2015] Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

[Scriven, Amiot-Cadey, and Collins 2011] Scriven, R.; Amiot-Cadey, G.; and Collins. 2011. *Collins French grammar*. Glasgow: HarperCollins.

[Siegelmann 1995] Siegelmann, H. T. 1995. Computation beyond the Turing limit. *Science* 286:545–548.

[Sun, Slusarz, and Terry 2005] Sun, R.; Slusarz, P.; and Terry, C. 2005. The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review* 112(1):59–192.

[VonMelchner, Pallas, and Sur 2000] VonMelchner, L.; Pallas, S. L.; and Sur, M. 2000. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature* 404:871–876.

[Voss 2013] Voss, P. 2013. Sensitive and critical periods in visual sensory deprivation. *Frontiers in Psychology* 4:664. doi: 10.3389/fpsyg.2013.00664.

[Weng, Ahuja, and Huang 1997] Weng, J.; Ahuja, N.; and Huang, T. S. 1997. Learning recognition and segmentation using the Cresceptron. *International Journal of Computer Vision* 25(2):109–143.

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

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

[Weng 2012] Weng, J. 2012. *Natural and Artificial Intelligence: Introduction to Computational Brain-Mind*. Okemos, Michigan: BMI Press.

[Weng 2015] Weng, J. 2015. Brain as an emergent finite automaton: A theory and three theorems. *International Journal of Intelligent Science* 5(2):112–131. received Nov. 3, 2014 and accepted by Dec. 5, 2014.

[Wiesel and Hubel 1965] Wiesel, T. N., and Hubel, D. H. 1965. Comparison of the effects of unilateral and bilateral eye closure on cortical unit responses in kittens. *Journal of Neurophysiology* 28:1029–1040.