View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by LSBU Research Open

A Survey and Analysis on Sequence Learning Methodologies and Deep Neural Networks

Yingxu Wang[1], Omar Zatarain[1], Daniel Graves[2], Marina Gavrilova[1],
Newton Howard[3] and Shushma Patel[4]

[1] International Institute of Cognitive Informatics and Cognitive Computing (ICIC)
Schulich School of Engineering, Hotchkiss Brain Institute, and Faculty of Sciences
University of Calgary
2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4
yingxu@ucalgary.ca, omar.zatarainduran@ucalgary.ca and mgavrilo@ucalgary.ca

[2] Edmonton Research Center
Huawei Canada
daniel.graves@huawei.com

[3] Oxford Neurocomputation Lab (NCL)
University of Oxford, UK
Email: newton.howard@nds.ox.ac.uk

[4] School of Engineering
London South Bank University, UK
Email: shushma@lsbu.ac.uk

Abstract — Sequence learning is one of the hard challenges to current machine learning and deep neural network technologies. This paper presents a literature survey and analysis on a variety of neural networks towards sequence learning. The conceptual models, methodologies, mathematical models and usages of classic neural networks and their learning capabilities are contrasted. Advantages and disadvantages of neural networks for sequence learning are formally analyzed. The state-of-the-art, theoretical problems and technical constraints of existing methodologies are reviewed. The needs for understanding temporal sequences by unsupervised or intensive-training-free learning theories and technologies are elaborated.

Keywords — Sequence learning, neural networks (NNs), deep NNs, recurrent NNs, analytic methodologies, denotational mathematics, cognitive systems, visual sequence learning, language sequence learning, applications

I. INTRODUCTION

Recent technologies of deep neural networks (DNN) and recurrent neural networks (RNNs) [Gers & Schmidhuber, 2001; Sutskever et al., 2014; Widrow et al., 2015] for deep learning [Rumelhart et al., 1986; Salakhutdinov & Joshua, 2012; Bengio et al., 2015; Schmidhuber, 2015] provide a promising approach to generic machine learning. However, it is recognized that a number of problems and constraints remain in current supervised learning technologies [Widrow & Lehr, 1990; Raytchev & Murase 2003; Widrow et al., 2015; Barbu, 2013; Wang, 2015, 2016a-d, 2017a-d] as follows:

a) Unsuitable for temporal and real-time sequence learning due to the need for supervision and human intervention;

b) Mathematical models are merely a special solution for a trained domain rather than a general solution in the universe of discourse for a category of problems;

c) A convergent mechanism suitable for pattern classification ($m << n$) rather than discriminative object identification ($m = n$) given arbitrary numbers of input vectors (n) and recognized outputs (m);

d) Exponential growth of topological and weight fitting complexities among inter-locked layers in deep and recurrent structures;

e) Data-driven rather than knowledge-driven thus requiring significantly large set of training data, intensive data labeling, and expensive human-aided data preprocessing;

f) Restricted processing power by dummy artificial nodes and networks underpinned by least square regression functions not sharable for fitting individual input vectors;

g) No inductive learning power to create and retain cumulative knowledge;

h) A brute-force philosophy ignoring problem contexts due to the lack of semantic comprehension ability and long-term knowledge base.

This paper presents a literature survey on sequence learning and neural network methodologies. It addresses the problems in current sequence learning technologies, the challenges of over complicated recurrent neural network solutions and the weaknesses of underpinning theories for sequence learning. In the remainder of this paper, the cognitive foundations of neural networks and machine learning are reviewed in Section II. A set of classical statically structured neural networks is reviewed for supervised machine learning in Section III. Dynamic neural networks such as deep, recurrent and long-short-term-memory neural networks for supervised learning are analyzed in Section IV. Some potential pitfalls and theoretical constraints in traditional neural networks for machine learning are formally analyzed in Section V.

II. THE COGNITIVE FOUNDATIONS OF NEURAL NETWORKS AND MACHINE LEARNING

In order to understand the central nervous systems of the brain and human learning mechanisms, the cognitive foundations of neural networks and learning are explores in this section towards sequence learning.

2.1 Cognitive Foundations of Neural Networks and the Nervous Systems of the Brain

Although there are various anatomic, neurological, and physiological models of neurons [Wilson & Keil, 2001; Hertz et al., 2006; Widrow et al., 2015; Wang, 2016d, 2017b; Wang & Wang, 2006; Wang & Fariello, 2012, 2013; Wang et al, 2017], there was a lack of formal models for them as a rigorous base of studies, particularly for mathematical neurology, neuroinformatics and computational intelligence.

Neurons are the basic unit of natural intelligence as information receptors, transmitters and servos in the brain and the nervous system throughout the body. A fundamental property of neurons is their dynamic connectivity to other neurons via synapses in order to form neural clusters and networks. The taxonomy of neurons is classified into three *functional categories* known as the *association, sensory*, and *motor* neurons. It is recognized that over 95% of neurons in the nervous system are association neurons. However, traditional artificial neural networks in AI and computational intelligence may have been modeled an artificial form of data-driven neurons that is not fully biologically accurate for explaining the neural foundations for machine learning, knowledge representation, reasoning thread establishment and behavior generation [Wang, 2016d, 2018; Wang & Fariello, 2012].

2.2 Cognitive Models of Machine Learning Based on Neural Networks

Learning is commonly perceived as a process of association of a certain form of object with existing knowledge in the memory of the brain [Reisenhuber & Poggio, 1999; Wilson & Frank, 2001; Wang, 2010, 2012c/d, 2013, 2015, 2016c]. A various forms of learning mechanisms have been identified in cognitive science and computational intelligence such as the classic conditioning learning, reinforced learning, supervised learning, latent learning, and social learning on the basis of behaviorism and associationism [Olshausen, 1996].

Definition 1. *Learning* is a cognitive process that cumulatively acquires knowledge or adaptively generates behaviors and skills.

Learning is an interaction among multiple fundamental cognitive processes such as object identification, abstraction, search, concept establishment, comprehension, memorization and retrieval. Learning is closely related to other higher cognitive processes of the brain such as deduction, induction, abduction, analogy, explanation, analysis, synthesis, creation, modeling and problem solving according to the Layered Reference Model of the Brain (LRMB) [Wang et al., 2006].

Definition 2. Machine learning can be classified into six categories known as object identification, cluster classification, pattern recognition, functional regression, behavioral (game) generation and knowledge acquisition as follows [Wang, 2015, 2016e]:

$$\begin{cases} L_i(\mathbf{x},\mathbf{P} \mid \mathbf{x} \subset \mathbf{X}) \triangleq \mathbf{x} = \mathbf{P}.\mathbf{x} & \text{// Object identification} \\ L_k(\mathbf{X},\mathbf{P}) \triangleq \mathbf{X} \subset \mathbf{P} & \text{// Cluster classification} \\ L_r(\mathbf{X},\mathbf{P}) \triangleq \mathbf{X} = \mathbf{P} & \text{// Pattern recognition} \\ L_g(\mathbf{X},\mathbf{P}) \triangleq \mathbf{X} \Rightarrow \mathbf{P}(\mathbf{X}) & \text{// Functional regression} \\ L_b(\mathbf{X},\mathbf{P}) \triangleq \mathbf{X} \Rightarrow f(\mathbf{P}(\mathbf{X})) & \text{// Behavior generation} \\ L_k(\mathbf{X},\mathbf{P}) \triangleq \mathbf{X} \Rightarrow c(\mathbf{X}) \uplus \mathbf{K} & \text{// Knowledge acquisition} \end{cases} \quad (1)$$

where $\boldsymbol{X}$ is a given variable vector or matrix of characteristic attributes of a pattern $\boldsymbol{P}$ such as a frame of image, a segment of voice, a stream of video and a sequence of sentences; f a certain function on $\boldsymbol{X}$; $c(\boldsymbol{X})$ a formal concept; and $\uplus$ a composition of a concept c with existing knowledge $\boldsymbol{K}$.

The sixth category of machine knowledge learning revealed by Wang [2016e] is the main form of human learning and an important type of sequence learning. A recent discovery in knowledge science is that the basic unit of knowledge is a *binary relation* (*bir*) [Wang, 2016a, 2017c] as that of binary digit (bit) for information and data. Knowledge learning is a life-long endeavor of humans that challenges current machine learning technologies.

III. STATICALLY STRUCTURED NEURAL NETWORKS FOR SUPERVISED MACHINE LEARNING

The classical structures of static neural networks for supervised machine learning are reviewed in this section, while its counterpart on dynamic and adaptive neural networks will be explored in the next section. The term of static neural

networks refers to those with fixed topological structures and node functions as well as calibrated weights after training.

3.1 Single-Layer Single-Output Artificial Neural Networks

A basic model of the node of classic artificial neural network (ANN(n,1,1)) is illustrated as shown in Figure 1, which represents the simplest neural network with a single node and a single-output [Hopfield & Tank, 1985].

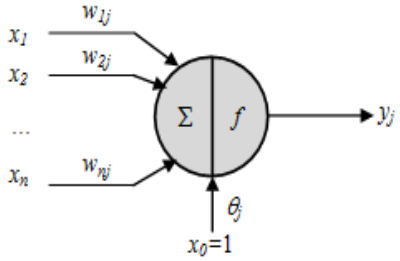

Fig. 1 A single-layer single-output ANN(n,1,1)

Definition 3. The *single-layer single-output artificial neural network* (ANN(n,1,1)), $\aleph^1_{ANN}(n,1,1)$, is the basic convergent node, as shown in Figure 1, that implements a weighted sum between the input vectors $\mathbf{X}(t) = \underset{i=0}{\overset{n}{R}} x_i(t)$ and the weight vector $\mathbf{W} = \underset{i=0}{\overset{n}{R}} w_i$, i.e.:

$$\begin{aligned} \aleph^1_{ANN}(n,1,1) &\triangleq \aleph^1_{ANN}(\mathbf{X}, \mathbf{W}, \mathbf{y}),\ \mathbf{X} = \underset{i=0}{\overset{n}{R}} x_i,\ \mathbf{W} = \underset{i=0}{\overset{n}{R}} w_i \\ &\triangleq \mathbf{y}(t) = f(\sum_{i=0}^{n} \mathbf{W}_i\, \mathbf{X}_i(t)) \end{aligned} \tag{2}$$

where the output $\boldsymbol{y}(t)$ at a time point t is transformed from a certain activation function $f(\sum_{i=0}^{n} \mathbf{W}_i\, \mathbf{X}_i(t))$ such as a *signum, step* or *sigmoid* function determined by a trained (calibrated) weight vector $\mathbf{W}^*$ (Eq. 3).

In Definition 3, the symbol $\underset{i=0}{\overset{n}{R}} X_i$ is known as the *big-R* notation [Wang, 2007] for denoting recurrent structures (|S) or repeated behaviors (|F) such as $\underset{i=0}{\overset{n}{R}} X_i|\mathrm{S} = (x_0, x_1, \ldots, x_n)$ or $\underset{i=0}{\overset{n}{R}} X_i|\mathrm{F} = (f_0 \to f_1 \to \ldots \to f_n)^{\cdot}$.

It is noteworthy that a neural network is a data-driven structure. The key methodology for building an artificial neural network is not only describe by the network function as given in Eq. 2, but also by its training function that determines how the neural network is calibrated for fitting an expected function.

Definition 4. The *training function* $\Gamma(\aleph^1_{ANN}(n,1,1))$ for the neural network $\aleph^1_{ANN}(n,1,1)$ determines the learning mechanism by $\gamma(\mathbf{W})$ that calibrates the weight of each neural link $\mathbf{W}^*$ in order to optimize the neural network for the expected function $f(\tau)$ for all input vectors $x(\tau)$, i.e.:

$$\begin{aligned} \Gamma(\aleph^1_{ANN}(n,1,1)) &\triangleq \Gamma(\aleph^1_{ANN}(\mathbf{X}, \mathbf{W}, \mathbf{y} \mid \mathbf{W} = \mathbf{W}^*)) \\ &= \mathbf{W}^* = \underset{i=0}{\overset{n}{R}} w_i^* \\ &= \gamma(\mathbf{W} \mid \underset{i=0}{\overset{n}{R}} \{ \lim_{w_i \to w_i^*} \underset{\tau=1}{\overset{|\mathfrak{U}|}{R}} [|y_i(\tau) - f(\sum_{i=0}^{n} w_i^* x_i(\tau))|] \to 0 \}) \end{aligned} \tag{3}$$

where $\mathfrak{U}$ is the universe of discourse of input vectors, which is infinitive in the domain of real numbers, i.e.:

$$|\mathfrak{U}| = n^{|\mathrm{K}|} = \infty,\ K = \mathbb{R} \tag{4}$$

Eq. 3 and 4 indicate that a neural network, even in the simplest single node configuration, may not always be trained or fitted for an expected function by the same set of weights in $\mathfrak{U}$.

Theorem 1. The *overestimated distinguishability* for ANN states that a weighted sum based neural network cannot uniquely identify different input vectors using an identical weight vector.

Proof. Theorem 1 is proved based on Definitions 3 and 4 as follows:

$$\begin{aligned} &\forall \aleph^1_{ANN}(n,1,1), X_a = \underset{i=0}{\overset{n}{R}} x_{a_i}, X_b = \underset{i=0}{\overset{n}{R}} x_{b_i}, f(X_\Sigma), \\ &\exists \Gamma(\aleph^1_{ANN}(n,1,1)) = \mathbf{W}^* = \underset{i=0}{\overset{n}{R}} w_i,\ f(X^a_\Sigma) = f(\sum_{i=0}^{n} w_i x_i^a),\ f(X^b_\Sigma) = f(\sum_{i=0}^{n} w_i x_i^b) \\ &\Rightarrow f(X^a_\Sigma) = f(X^b_\Sigma),\ X_a \neq X_b \quad \blacksquare \end{aligned} \tag{5}$$

Example 1. Let a $\aleph^1_{ANN}(3,1,1)$ be trained with a set of weights W^* = [1, 1, 1]. Then, it cannot distinguish the input vectors X_1 = [1,2,3] and X_2 = [3,2,1] because $\forall f(X_\Sigma) = f(\sum_{i=0}^{n} w_i x_i),\ f_1(X^1_\Sigma) = f_2(X^2_\Sigma) = f(6)$ for any activation function.

Therefore, ANNs may work only for object classification but not good at object identification in an arbitrary domain of input vectors.

3.2 Single-Layer Multi-Output Artificial Neural Networks

A single-layer multi-output (m) artificial neural network (ANN(n,1,m)) is illustrated in Figure 2. The ANN(n,1,3) neural network can be recursively composed by three ANN(n,1,1) in the given topological configuration, so do their mathematical models.

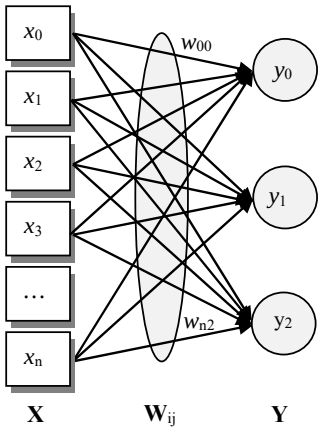

Fig. 2 A single-layer multi-output ANN(n,1,m)

Definition 5. A *single-layer multi-output* ANN(n,1,m), $\aleph^1_{ANN}(n,1,m)$, is a set of multiple $\aleph^1_{ANN}(\mathbf{X}, \mathbf{W}, \mathbf{y})$ for implementing parallel ANNs sharing the same input vector **X**, i.e.:

$$\begin{aligned} \aleph^1_{ANN}(n,1,m) &\triangleq \mathop{R}_{j=1}^{m} \aleph^1_{ANN}(n,1,1) \\ &\triangleq \mathop{R}_{j=1}^{m} \aleph^1_{ANN}(\mathbf{X}, \mathbf{W}_j, \mathbf{Y}_j) \\ &\triangleq \mathop{R}_{j=1}^{m} [\mathbf{Y}_j(t) = f_j(\sum_{i=0}^{n} \mathbf{W}^*_{ji} \mathbf{X}_i(t))] \end{aligned} \tag{6}$$

where $f_j(\sum_{i=0}^{n} \mathbf{W}^*_{ji} \mathbf{X}_i)$ is an activation function that transforms a weighted sum into a set of trained classifications.

Theorem 2. The *training criterion* for an ANN(n,1,m), $\Gamma(\aleph^1_{ANN}(n,1,m))$, is the optimization of its weights, i.e.:

$$\begin{aligned} \Gamma(\aleph^1_{ANN}(n,1,m)) &\triangleq \Gamma(\mathop{R}_{j=1}^{m} \aleph^1_{ANN}(\mathbf{X}, \mathbf{W}, \mathbf{Y})) \\ &\triangleq \mathbf{W}^* = \mathop{R}_{j=1}^{m} \mathop{R}_{i=1}^{n} w^*_{ji} \end{aligned} \tag{7}$$

Proof. Theorem 2 is proved based on Definitions 4 and 5 as follows:

$$\begin{aligned} \Gamma(\aleph^1_{ANN}(n,1,m)) &= \Gamma(\mathop{R}_{j=1}^{m} \aleph^1_{ANN}(\mathbf{X}, \mathbf{W}, \mathbf{Y})) \\ &= \mathop{R}_{j=1}^{m} \gamma_j(\mathbf{W}_j \mid \mathop{R}_{i=0}^{n} \{\lim_{w_{ji} \to w^*_{ji}} \mathop{R}_{\tau=1}^{|\mathcal{I}|} [|y_{ji}(\tau) - f(\sum_{ji=0}^{n} w^*_{ji} x_i(\tau))|] \to 0\}) \\ &= \mathbf{W}^* = \mathop{R}_{j=1}^{m} \mathop{R}_{i=1}^{n} w^*_{ji} \quad \blacksquare \end{aligned} \tag{8}$$

where **W*** is the optimal weight vector that is problem dependent.

It is noteworthy in practice that the training goal as stated in Theorem 2 is usually unachievable because of the nature of problems, the limited sizes of training data and contradictory influences among sample data. Therefore, the calibration of an ANN by training may impossible to fit all input vectors by a single set of static weights according to Theorems 1 and 2, because the calibrated weights may severely deviate from the generally expected least-square regression. For instance, a simple logical AND gate cannot be accurately trained and ideally fitted by ANNs [Mehrotra et al., 2000].

3.3 Multi-Layer Multi-Output Artificial Neural Networks

A multi-layer (k) multi-output (m) artificial neural network (ANN(n,k,m)) is illustrated in Figure 3. The ANN(n,2,2) neural network can be recursively composed by [3 • ANN(n, 1, 1) ° 2 • ANN(3, 1, ,1)] in the given topological configuration where ° represents a composition between two adjacent layers of the ANN(n, 1, m_1) and ANN(m_1, 1, m) networks.

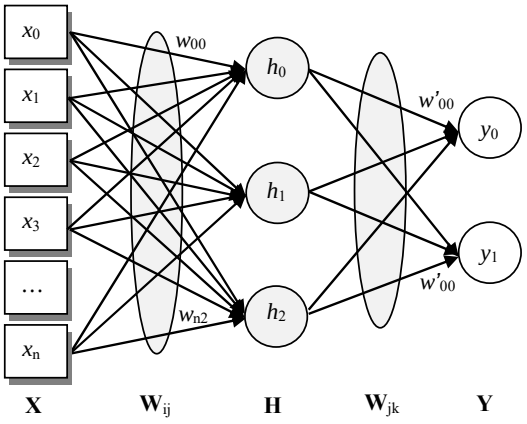

Fig. 3 A multi-layer multi-output ANN(n,k,m)

Definition 6. Let $\mathbf{X} = \mathop{R}_{k=1}^{K} \mathop{R}_{i_k=0}^{n_k} x^k_{i_k}$, $\mathbf{W} = \mathop{R}_{k=1}^{K} \mathop{R}_{i_k=0}^{n_k} w^k_{i_k}$, $\mathbf{Y}^k = \mathop{R}_{j=0}^{m_k} y^k_j$.

An *k-layer m*-output *artificial neural network* ANN(n,k,m), $\aleph^k_{ANN}(n,k,m)$, is a set of recursively configured single-layer multi-output ANNs, which may be coordinately trained through all layers, i.e.:

$$\begin{aligned} \aleph^k_{ANN}(n,k,m) &\triangleq \mathop{R}_{k=1}^{K} \aleph^1_{ANN}(n_k,1,m_k) \\ &= \mathop{R}_{k=1}^{K} \mathop{R}_{j=1}^{m_k} \aleph^1_{ANN}(n_{kj},1,1) \\ &= \mathop{R}_{k=1}^{K} \mathop{R}_{j=1}^{m_k} \aleph^1_{ANN}(\mathbf{X}_k, \mathbf{W}^*_{kj}, \mathbf{Y}_{kj}) \\ &= \mathop{R}_{k=1}^{K} \mathop{R}_{j=1}^{m_k} [\mathbf{Y}^k_{i_k}(t) = f_{kj}(\sum_{i_k=0}^{n_k} \mathbf{W}^{k*}_{i_k} \mathbf{X}^k_{i_k}(t))], \\ &\quad \mathbf{W}^* = \Gamma(\mathop{R}_{k=1}^{K} \mathop{R}_{j=1}^{m_k} \mathop{R}_{i=0}^{n_{kj}} \aleph^1_{ANN}(\mathbf{X}_k, \mathbf{W}_{kj}, \mathbf{Y}_{kj})) \end{aligned} \tag{9}$$

where $\mathbf{X}^1 = \underset{i_1=0}{\overset{n_1}{R}} x_{i_1}^1(t)$ is the initial input vector and $\mathbf{W}^1 = \underset{i_1=0}{\overset{n_1}{R}} w_{i_1}^1(t)$ is the primitive weights corresponding to each input in $\mathbf{X}^1$.

It is found that an ANN is perhaps not a generic analytic model for arbitrary neural structures. It is rather than an approximation model for functional regression by a series of weighted sums of the input vectors. It cannot generally fit an arbitrary polynomial as the universal model of dynamic and convergent functions, because the generic mechanisms of the neurons and neural networks are not a weighted sum, nor an *n*-to-1 convergent structure.

Theorem 2 reveals that ANNs are not generally suitable for dynamic, sequential and unsupervised learning, because the functioning of ANNs is dependent on proper training, which cannot be implemented by random, redundant and low coverage training data in a given domain. In other words, due to the state space of convergent functions is quiet large and of divergent functions are infinitive, a certain set of fixed weights cannot fit all in the domains of arbitrary problems.

IV. DYNAMIC NEURAL NETWORKS FOR SUPERVISED MACHINE LEARNING

Many important and hard problems in machine learning are characterized as dynamic sequences with finite or infinite lengths. Such problems include video stream recognition, speech recognition, language translation, machine knowledge learning and cognitive knowledge base manipulations. These hard AI problems demand dynamic neural networks with both adaptive structures and weights such as deep and recurrent neural networks for supervised learning. However, theories and technologies for unsupervised, non-data-driven, light-training-based and training-free neural networks are yet to be created and developed.

4.1 Deep Neural Networks (DNN)

In order to address the unideal ANN performances as described in preceding subsection, deep neural networks are proposed for implementing deep machine learning towards solving hard AI problems [Hinton et al., 1995; Hinton & Salakhutdinov, 2006; Collobert & Weston, 2008; Ciresan et al., 2010; Arel et al., 2010; Mnih et al., 2015; Schmidhuber, 2015].

Definition 7. Let $\mathbf{X} = \underset{k=1}{\overset{K}{R}} \underset{i_k=0}{\overset{n_k}{R}} x_{i_k}^k$, $\mathbf{W} = \underset{k=1}{\overset{K}{R}} \underset{i_k=0}{\overset{n_k}{R}} w_{i_k}^k$, $\mathbf{Y}^k = \underset{j=0}{\overset{p_k}{R}} y_j^k$. A *deep neural network* (DNN(*n*,k,m)), $\aleph_{DNN}^k(n,k,m)$, is a *k*-layered recursive ANNs where $k > 3$, which can be formally described as follows:

$$\begin{aligned}\aleph_{DNN}^k(n,k,m) &\triangleq \underset{k=1}{\overset{K}{R}} \aleph_{ANN}^k(n,k,m) \\ &= \underset{k=1}{\overset{K}{R}} \underset{i_k=0}{\overset{n_k}{R}} [\mathbf{Y}_{i_k}^k = f(\sum_{i_k=0}^{n_k} \mathbf{W}_{i_k}^k \mathbf{X}_{i_k}^k)] \\ &= \underset{k=1}{\overset{K}{R}} \underset{i_m=0}{\overset{n_m}{R}} \{y_{i_m}^m(t) = f(\sum_{i_m=0}^{n_m} \cdots \sum_{i_2=0}^{n_2} \sum_{i_1=0}^{n_1} w_{i_m}^m(t) \cdots w_{i_2}^2(t) w_{i_1}^1(t) x_i^1(t))\}\end{aligned} \tag{10}$$

where the activation function *f* is normally a continuously differentiable function, typically a *sigmoid* function as given in Eq. 11, in order to facilitate least square optimization during training.

$$f_{sigmoid}(s) = \frac{1}{1+e^{-2s}},\ s = \sum_{i=0}^{n} w_i x_i \tag{11}$$

However, because the basic node of the DNN is still implemented by an ANN, all constraints and problems of ANNs have been inherited as described in Section 3. Therefore, it is unlikely to expect that DNNs may solve dynamic and sequence learning problems. It cannot be proven that the deeper the DNN, the better the performance. Due to feedback and feedforward operations for optimization among a large set of weights at different layers, the cost of training for a DNN is exponentially increasing according to Eq. 10.

Many questions on DNNs are unanswered such as: a) Is the deeper the better in DNNs? and c) How does potential performance gain be balanced with the exponential increase of training cost and the significant decrease of run-time efficiency?

4.2 Recurrent Neural Networks (RNN)

A recurrent neural network is a cyclic network in order to deal with dynamic temporal behaviors of learning problems which may involve previous learning results [Pearlmutter, 1989; Giles et al., 1992; Omlin & Giles, 1996; Schuster & Paliwal, 1997; Siegelmann et al., 1997; Auli, et al., 2013]. Therefore, internal memory is required in recurrent networks in order to hold historical learning information. As a result, the weight of each node is expected to be dynamically determined in each cycle of learning.

Definition 8. A *recurrent neural network* (RNN(*n*,*k*,*m*)), $\aleph_{RNN}^k(n,k,m)$, is a cyclic network with dynamically predicated weights influenced by the previous learning cycle, i.e.:

$$\begin{aligned}\aleph_{RNN}^m &= \underset{k=1}{\overset{K}{R}} \aleph_{ANN}^k(\mathbf{X}^k, \mathbf{W}^k, \mathbf{Y}^{k-1}),\ \mathbf{Y}^0 = \varnothing \\ &= \underset{k=1}{\overset{K}{R}} \{\mathbf{Y}^k = \sum_{i_k=1}^{n_k} \mathbf{w}_{i_k}^k (\mathbf{X}^k + f_{i_k}^k(\mathbf{Y}^{k-1})\}\end{aligned} \tag{12}$$

where $f_{i_k}^k(\mathbf{Y}^{k-1})$ is the introduction of previous learning result transformed by a certain function dependent on the problem.

It is recognized that an RNN is an extend $\aleph_{ANN}^k$ with historical $f_{i_k}^k(\mathbf{Y}^{k-1})$ information of the previous learning cycle. Because all problems of ANNs have been inherited in RNN, it is hard to formally determine how trainings may be implemented on the fly in each cycle. It requires intensive preprocessing for data labeling in order to assistant dynamic calibration of the weight vectors in each application cycle of the RNN. As a result of the intricate complexity, extremely high performance computing power involving multiple GPUs is demanded for even a simple applications.

4.3 RNN with Long-Short-Term Memory (LSTM)

Long short-term memory is a technology for enhancing the conceptual model of RNNs with local (short-term memory, STM) and global (long-term memories, LTM) [Hochreiter & Schmidhuber, 1997, Gers & Schmidhuber, 2001]. LSTM has been applied in pattern recognition, handwriting recognition and speech recognition in recent years. Google's speech recognition has archived a 49% accuracy by LSTM enhanced RNN in 2015 [Sutskever et al., 2014], though it is still far from ideal.

Definition 9. A *recurrent neural network with long- short-term-memory* (RNN-LSTM(*n*,*k*,*m*)), $\aleph_{LSTM}^k(n,k,m)$, is an RNN with the support of local STM and global LTM in order to deal with learning problems constrained by long-range temporal dependencies acquired in all previous steps, *s*, of historical learning $\underset{s=0}{\overset{k-1}{R}}\mathbf{Y}^s$, i.e.:

$$\begin{aligned}\aleph_{LSTM}^k(n,k,m) &\triangleq \underset{k=1}{\overset{K}{R}}\aleph_{ANN}^k(\mathbf{X}^k,\mathbf{W}^k,\mathbf{Y}),\ \mathbf{Y}^0=\varnothing \\ &= \underset{k=1}{\overset{K}{R}}\{\mathbf{Y}^k = f_k(\sum_{i_k=0}^{n_k}\mathbf{W}_{i_k}^k(\mathbf{X}^k + f_{i_k}^k(\mathbf{Y}))\}\end{aligned} \tag{13}$$

As that a classical RNN considers the influence of output in the previous cycle, RNN-LSTM involves all historical results or cumulative learning acquired in the LTM. A typical strategy in RNN-LSTM is to utilize learning results in LTM as a conditional probability $p(\underset{i=0}{\overset{n}{R}}\mathrm{y}_i \mid \underset{i=0}{\overset{n}{R}}\mathrm{x}_i)$ between an input sequence $\underset{i=0}{\overset{n}{R}}x_i$ and the corresponding output sequence $\underset{i=0}{\overset{n}{R}}y_i$. However, the complexity is extremely high and there is no common algorithm to implement an RNN-LSTM. Because RNN-LSTM still requires complex training, it is not suitable for unsupervised learning particularly in real-time sequence processing.

V. PITFALLS IN TRADITIONAL NEURAL NETWORKS FOR MACHINE LEARNING

It is recognized that the entire AI problems in general and deep learning challenges in particular had been extremely persistent, complex and hard, because they were out of the domain of $\mathbb{R}$ (real numbers) and traditional analytic mathematics [Wang, 2012b]. The emerging class of entities are identified as hyperstructure ($\mathbb{H}$) and the contemporary mathematical means for dealing with them are known as denotational mathematics [Wang, 2012a].

It is recognized that, given any pair of a mathematical model and its implementation, *iff* the mathematical model is correct, it may be implemented in machine learning; However, if the mathematical model is incorrect, it may never be implemented. This is the general reality of traditional neural networks for machine learning. Although experiments in limited domains and simplified applications of DNNs seemed working, a generic theory and rigorous mathematical models are yet to be sought and validated

A number of potential pitfalls and constraints in neural network (NN) based machine learning, including ANNs, DNNs and RNNs are recognized as summarized in Table 1. The learning constraints are evaluated against the six categories of machine learning as formally classified in Definition 2.

In this literature survey and analyses, the following fundamental questions are raised if the mathematical models of NNs are not a general mathematical model to fit arbitrary nonlinear functions: a) What is a neurologically correct NN? and b) What is a mathematically general NN? Because the state spaces of typical problems are too large even infinitive, and too many input vectors may result in the same output in a multi-layer weighted sum, the redundancy in training cannot be avoid and learning accuracy cannot be guaranteed.

It is recognized that traditional deep neural networks are a converging network good at object classification and pattern recognition. However, they are not practical to object identification because the outputs of a trained deep neural network and its required sets of trained weights are always less than the number of input vectors.

Theorem 3. Deep neural network based machine learning may only implement object classification rather than object identification in the universe of discourse of problems, $\mathcal{U}$, constrained by its convergent structures.

Proof. Let *n* and *m* represent the numbers of independent input vectors and the size of the output vector of a deep neural network. Although object classification may allow $m << n$, image identification requires $m \geq n$, i.e.:

$$\begin{cases} m = |\mathbf{Y}_{output}| \geq n = |\mathbf{X}_{input}|, & \text{Identification in } \mathcal{U} \\ m = |\mathbf{Y}_{output}| < n = |\mathbf{X}_{input}|, & \text{Classification in } \mathcal{U} \end{cases} \quad (14)$$

Why are not NNs a generic analytic model for machine learning? In order to explain why NNs does not work well as a general methodology for machine learning and why their training costs are extremely high, consider the following principle.

Corollary 1. An arbitrary problem generally represented by a polynomial cannot be fitted by NNs in the universe of discourse $\mathcal{U}$ of nonlinear functions or arbitrary vector distributions.

Table 1. Constraints of Classic Neural Networks in Machine Learning

No.	Property and constraint	Explanation	Limitation in categories of learning (Definition 2)					
			object identification	cluster classification	pattern recognition	functional regression	behavioral generation	knowledge cognition
1	Special vs. general solutions	The mathematical models of NNs are a special solution in a trained domain rather than a general solution in $\mathfrak{u}$, because $\mathfrak{u}$ cannot be covered by nonindependent or redundant data.		√	√		√	
2	Data-driven vs. knowledge-driven	Data-driven NNs require significantly large set of training data, intensive data labeling, and expensive human-aided data preprocessing.		√	√	√		
3	*Onto* (n-1) vs. partial (1-1) function structures	NNs are a special mechanism suitable for convergent pattern classification ($m << n$) rather than discriminative object identification ($m = n$) given arbitrary numbers of input vectors (n) and recognition outputs (m);		√	√			
4	Exponential growth of complexities	NNs result in exponential growth of topological complexities in deep and recurrent structures.	√	√	√	√		
5	Overloaded least square regressions	Least square regression by training may only fit a few specific input vectors, which is not sharable by others. Thus, no generic fit exist by NNs in $\mathfrak{u}$.		√	√	√		
6	No inductive learning power	NNs lack an inductive learning power to create and retain cumulative knowledge.		√	√		√	
7	No semantic and knowledge comprehension	NNs' brute-force mechanism ignores problem contexts due to the lack of semantic comprehension ability and long-term knowledge base.	√	√	√	√		
8	No support for sequence learning	NNs are unsuitable for temporal and real-time sequence learning due to the need for training, supervision and human intervention.	√	√	√	√	√	

Table 2. Paradigms of Neural Networks towards Sequence Learning

No.	Category	Method	Symbol	Mathematic model	Capability		
					Static learning	Dynamic learning	Sequential learning
1	Static and supervised	Artificial NN	ANN	$\aleph^1_{ANN} = f(\sum_{i=0}^{n} \mathbf{W}_i \mathbf{X}_i)],\ \aleph^k_{ANN} = \underset{k=1}{\overset{K}{R}} \aleph^1_{ANN}(n_k, 1, m_k)$	√		
2	Dynamic and supervised	Deep NN	DNN	$\aleph^k_{DNN}(n,k,m) = \underset{k=1}{\overset{K}{R}} \aleph^k_{ANN}(n,k,m) = \underset{k=1}{\overset{K}{R}} \underset{i_k=0}{\overset{n_k}{R}} [\mathbf{Y}^k_{i_k} = f(\sum_{i_k=0}^{n_k} \mathbf{W}^k_{i_k} \mathbf{X}^k_{i_k})]$	√	√	
3		Recurrent NN	RNN	$\aleph^m_{RNN} = \underset{k=1}{\overset{K}{R}} \aleph^k_{ANN'}(\mathbf{X}^k, \mathbf{W}^k, \mathbf{Y}^{k-1}),\ \mathbf{Y}^0 = \varnothing$ $= \underset{k=1}{\overset{K}{R}} \{\mathbf{Y}^k = \sum_{i_k=0}^{n_k} \mathbf{w}^k_{i_k} (\mathbf{X}^k + f^k_{i_k}(\mathbf{Y}^{k-1})\}$		√	
4		Recurrent NN + LSTM	RNN-LSTM	$\aleph^k_{LSTM} = \underset{k=1}{\overset{K}{R}} \aleph^k_{ANN^*}(\mathbf{X}^k, \mathbf{W}^k, \mathbf{Y}),\ \mathbf{Y}^0 = \varnothing$ $= \underset{k=1}{\overset{m}{R}} \{\mathbf{Y}^k = f_k(\sum_{i_k=0}^{n_k} \mathbf{W}^k_{i_k} (\mathbf{X}^k + f^k_{i_k}(\mathbf{Y}))\}$		√	
5	Dynamic and unsupervised	?	∇NN	?	√	√	√

Proof. According to Definition 3, the mathematical models of a neural network $N(\mathbf{X})$ and of a general problem as a polynomial $P(\mathbf{X})$ are, respectively:

$$N(x) = f(\sum_{i=0}^{n} w_i x_i) = f(w_n x_n + ... + w_3 x_3 + w_2 x_2 + w_1 x_1 + w_0 x_0) \quad (15)$$
$$P(x) = \sum_{i=0}^{n} w_i x^i = w_n x^n + ... + w_3 x^3 + w_2 x^2 + w_1 x + w_0$$

Thus, $P(\mathbf{X})$ can fit $N(\mathbf{X})$, but not vice versa. ■

The methodologies of typical neural networks for machine learning are summarized in Table 2 on the basis of the comparative analyses throughout the paper. By contrasting the categories, topologies, methods, mathematical models, properties and capability of current neural networks, more powerful non-dada-driven and inductive learning methodologies towards machine knowledge learning will be developed.

It is noteworthy that the persistent hard problems in machine learning are characterized as temporary sequences and dynamic frames underpinned by an extremely large even infinitive universe of discourse in video stream recognition, speech recognition, language translation, machine knowledge learning and cognitive knowledge base manipulations. This set of hard AI problems demands dynamic neural networks powered by adaptive structures and flexible weight vectors for unsupervised, light-training-based or training-free neural networks yet to be explored.

VI. CONCLUSION

This paper has presented a comparative analyses of the theories and methodologies of a variety of neural network technologies as well as their advantages and disadvantages in machine learning. A comprehensive literature survey on neural network technologies towards sequence learning has been reported. The state-of-the-art, theoretical problems and technical constraints of traditional methodologies have been reviewed. Challenges and needs for understanding temporal sequences by unsupervised learning theories and technologies have been elaborated.

It has been found in this survey that classical data-driven and intensive-training-based neural networks are not suitable to sequence learning because they cannot fulfill the basic requirements for unsupervised or fully self-adaptive learning. It has also found that persistent hard problems in machine learning are characterized as temporary sequences and dynamic vectors underpinned by an extremely large even infinitive universe of discourse. This set of hard AI problems will demand dynamic neural networks powered by adaptive structures and flexible weights for unsupervised, non-data-driven, light-training-based and training-free neural networks yet to be developed.

ACKNOWLEDGEMENT

This work is supported by the Huawei Canada HIRPO2016CA23 grant and an associate NSERC CRD grant (pending). The authors and PI would like to thank the support of Huawei Canada and NSERC. The author would like to thank the anonymous reviewers for their valuable suggestions and comments on the paper.

REFERENCES

[1] Arel, I., D.C. Rose and T.P. Karnowski (2010), Deep Machine Learning – A New Frontier in Artificial Intelligence Research, *IEEE Computational Intelligence* Magazine, 5(4), 13-18.

[2] Barbu, T. (2013), Unsupervised SIFT-based Face Recognition using an Autonomatic Hierarchical Agglomerative Clustering Solution, *Procedia Computer Science*, Vol. 22, 385-394.

[3] Bengio, Y., Y. LeCun and G. Hinton (2015), Deep Learning. *Nature*, 521: 436–444.

[4] Ciresan, D.C. et al. (2010), Deep Big Simple Neural Nets for Handwritten Digit Recognition, *Neural Computation*, 22, 3207–3220.

[5] Collobert, R. and J. Weston (2008), A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning, Proceedings of 25th International Conference on Machine Learning (ACM ICML'08), NY, USA, pp.160–167.

[6] Gers, F. and J. Schmidhuber (2001), LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages, *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001

[7] Giles, C.L., C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun and Y.C. Lee (1992), Learning and Extracting Finite State Automata with Second-Order Recurrent Neural Networks, *Neural Computation*, 4(3), p. 393.

[8] Gers, F.A. and J. Schmidhuber (2001), LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages, *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001

[9] Hertz, A.V.M., Tim Gollisch, Christian K. Machens, Dieter Jaeger (2006), Modeling Single-Neuron Dynamics and Computations: A balance of detail and abstraction, *Science*, Vol. 314(5796), 80-85.

[10] Hinton, G.E., P. Dayan, B.J. Frey, R. Neal (1995), The Wake-sleep Algorithm for Unsupervised Neural Networks, *Science*, 268 (5214),1158–1161.

[11] Hinton, G.E. and R. Salakhutdinov (2006), Reducing the Dimensionality of Data with Neural Networks, *Science,* 313, 504–507.

[12] Hochreiter, S. and J, Schmidhuber (1997) Long Short-Term Memory, *Neural Computation*, 9(8):1735–1780.

[13] Hopfield, J. and D. Tank (1985), Neural Computation for Decisions in Optimization Problems, *Biological Cybernetics*, 52, 141-152,

[14] Mehrotra, K., C.K. Mohan and S. Ranka (2000), *Elements of Artificial Neural Networks*, The MIT Press, MA.

[15] Mnih, V. et al. (2015), Human-level Control through Deep Reinforcement Learning. *Nature*, 518: 529–533.

[16] Omlin, C.W., C.L. Giles (1996), Constructing Deterministic Finite-State Automata in Recurrent Neural Networks, *Journal of the ACM*, 45(6), 937-972.

[17] Pearlmutter, B.A. (1989), Learning State Space Trajectories in Recurrent Neural Networks, *Neural Computation*, 1(2):263–269.

[18] Riesenhuber, M. and T. Poggio (1999), Hierarchical Models of Object Recognition in Cortex. *Nature-Neuroscience*, 2(11): 1019–1025.

[19] Rumelhart, D., G.E. Hinton and R.J. Williams (1986), Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536.

[20] Raytchev, B. and H. Murase (2003), Unsupervised Face Recognition by Associative Chaining, *Pattern Recognition*, 36(1): 245-257.

[21] Salakhutdinov, R. and T. Joshua (2012), Learning with Hierarchical-Deep Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35: 1958–71.

[22] Schmidhuber, J. (2015), Deep Learning in Neural Networks: An Overview, *Neural Networks*, 61: 85–117.

[23] Schuster, M. and K.K. Paliwal (1997), Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing*, 45:2673–81.

[24] Siegelmann, Hava T., Bill G. Horne, C. Lee Giles (1997), Computational capabilities of recurrent NARX neural networks, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B 27(2): 208-215.

[25] Silver, D., A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, J. Antonoglou and V. Panneershelvam (2016), Mastering the game of Go with deep neural networks and tree search. *Nature*, 529 (7587): 484–489.

[26] Sutskever, I., O. Vinyals, Q. Le (2014), Sequence to Sequence Learning with Neural Networks, *Proc. NIPS*. Google, pp.1-9.

[27] Wang, Y. (2007), *Software Engineering Foundations: A Software Science Perspective*, CRC Book Series in Software Engineering, Vol. II, CRC Auerbach Publications, NY, USA.

[28] Wang, Y. (2008), On Contemporary Denotational Mathematics for Computational Intelligence, *Transactions of Computational Science,* 2, 6-29.

[29] Wang, Y. (2010), Cognitive Robots: A Reference Model towards Intelligent Authentication, *IEEE Robotics and Automation*, 17(4), 54-62.

[30] Wang, Y. (2012a), In Search of Denotational Mathematics: Novel Mathematical Means for Contemporary Intelligence, Brain, and Knowledge Sciences, *Journal of Advanced Mathematics and Applications*, 1(1), 4-25.

[31] Wang, Y. (2012b), Keynote: Towards the Next Generation of Cognitive Computers: Knowledge vs. Data Computers, *12th International Conference on Computational Science and Applications* (ICCSA'12), Salvador, Brazil, Springer, June, pp.3.

[32] Wang, Y. (2012c), On Abstract Intelligence and Brain Informatics: Mapping Cognitive Functions of the Brain onto its Neural Structures, *International Journal of Cognitive Informatics and Natural Intelligence*, 6(4), 54-80.

[33] Wang, Y. (2012d), On the Denotational Mathematics Foundations for the Next Generation of Computers: Cognitive Computers for Knowledge Processing, *Journal of Advanced Mathematics and Applications*, 1(1), 118-129.

[34] Wang, Y. (2013), Keynote: Basic Theories for Neuroinformatics and Neurocomputing, *Proceedings 12th IEEE International Conference on Cognitive Informatics and Cognitive Computing* (ICCI*CC 2013), New York, USA, IEEE CS Press, July, pp.3-4.

[35] Wang, Y. (2015), Cognitive Learning Methodologies for Brain-Inspired Cognitive Robotics, *International Journal of Cognitive Informatics and Natural Intelligence,* 9(2), 37-54.

[36] Wang, Y. (2016a), Keynote: Deep Learning and Deep Thinking by Cognitive Robots and Computational Intelligent Systems, *15th IEEE International Conference on Cognitive Informatics and Cognitive Computing* (ICCI*CC'16), Stanford University, Stanford, CA, IEEE CS Press, Aug. 8, pp. 4-6.

[37] Wang, Y. (2016b), Keynote: Cognitive Soft Computing: Theoretical and Mathematical Foundations of Cognitive Robotics, *6th World Conference on Soft Computing* (WConSC'16), UC Berkeley, CA, Springer, May. 22-25.

[38] Wang, Y. (2016c), Keynote: Brain-Inspired Deep Machine Learning and Cognitive Learning Systems, *8th International Conference on Brain Inspired Cognitive Systems* (BICS'16), Beijing, Nov. 28-30, pp. 2.

[39] Wang, Y. (2016d), Keynote: Cognitive Neuroscience and the Spike Frequency Modulation (SFM) Theory for Neural Signaling Systems, *9th Global Neuroscience Conference (GNSC'16), Melbourne, Australia, Nov. 21-22*, pp. 22.

[40] Wang, Y. (2016e), On Cognitive Foundations and Mathematical Theories of Knowledge Science, *International Journal of Cognitive Informatics and Natural Intelligence*, 10(2), 1-24.

[41] Wang, Y. (2017a), Keynote: Cognitive Foundations of Knowledge Science and Deep Knowledge Learning by Cognitive Robots, *16th IEEE International Conference on Cognitive Informatics and Cognitive Computing* (ICCI*CC 2017), University of Oxford, UK, IEEE CS Press, July 26-28, pp. 5.

[42] Wang, Y. (2017b), Keynote: From Bioengineering and Cognitive Engineering to Brain Inspired Systems, *5th International Summit on Medical Biology and Bioengineering* (ISMBBE'17), C*hicago, USA, Sept. 27-28, pp. 2*.

[43] Wang, Y. (2018), Keynote: Cognitive Foundations and Formal Theories of Human and Robot Visions, *17th IEEE International Conference on Cognitive Informatics and Cognitive Computing*, University of California, Berkeley, USA, IEEE CS Press, July, in press.

[44] Wang, Y., N. Howard, J. Kacprzyk, O. Frieder, P. Sheu, R.A. Fiorini, M. Gavrilova, S. Patel, J. Peng, and B. Widrow (2018), Cognitive Informatics: Towards Cognitive Machine Learning and Autonomous Knowledge Manipulation, *Int'l Journal of Cognitive Informatics and Natural Intelligence*, 12(1), 1-13.

[45] Wang, Y. and G. Fariello (2012), On Neuroinformatics: Mathematical Models of Neuroscience and Neurocomputing, *Journal of Advanced Mathematics and Applications*, 1(2), 206-217.

[46] Wang, Y. and G. Fariello (2013), The Theory of Neural Circuits for Neuroinformatics and Neurocomputing, *Journal of Advanced Mathematics and Applications*, 2(1), in press.
[47] Wang, Y. and Y. Wang (2006), Cognitive Informatics Models of the Brain, *IEEE Transactions on Systems, Man, and Cybernetics* (Part C), 36(2), March, 203-207.
[48] Wang, Y., Y. Wang, S. Patel, and D. Patel (2006), A Layered Reference Model of the Brain (LRMB), *IEEE Transactions on Systems, Man, and Cybernetics* (Part C), 36(2), March, 124-133.
[49] Wang, Y., Lotfi A. Zadeh, Bernard Widrow, Newton Howard, et al. (2017), Abstract Intelligence: Embodying and Enabling Cognitive Systems by Mathematical Engineering, *International Journal of Cognitive Informatics and Natural Intelligence,* 11(1), 1-15.
[50] Widrow, B. and M.A. Lehr (1990), 30 Years of Adaptive Neural Networks: Perception, Madeline, and Backpropagation, *Proc. of the IEEE*, Sept., 78(9), 1415-1442.
[51] Widrow, B., Y. Kim, and D. Park (2015), The Hebbian-LSM Learning Algorithm, *IEEE Computational Intelligence Magazine*, Nov. pp. 27-53.
[52] Wilson, R.A. and F.C. Keil (2001), *The MIT Encyclopedia of the Cognitive Sciences*, MIT Press.