



AFRL-RI-RS-TR-2019-019

**A HOLISTIC APPROACH TO UNDERSTANDING AND
BENCHMARKING OF COGNITIVE AND ARCHITECTURAL
CHARACTERISTICS FOR NEUROMORPHIC ARCHITECTURES**

STATE UNIVERSITY OF NEW YORK AT BINGHAMTON

FEBRUARY 2019

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2019-019 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

THOMAS RENZ
Work Unit Manager

/ S /

JOSEPH A. CAROLI
Chief, High Performance Systems Branch
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) FEBRUARY 2019		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) MAR 2016 – AUG 2018	
4. TITLE AND SUBTITLE A HOLISTIC APPROACH TO UNDERSTANDING AND BENCHMARKING OF COGNITIVE AND ARCHITECTURAL CHARACTERISTICS FOR NEUROMORPHIC ARCHITECTURES				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER FA8750-16-2-0134	
				5c. PROGRAM ELEMENT NUMBER 62788F	
6. AUTHOR(S) Zhanpeng Jin				5d. PROJECT NUMBER T2DN	
				5e. TASK NUMBER BI	
				5f. WORK UNIT NUMBER NG	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SUNY at Binghamton PO Box 6000 Binghamton NY 13902				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITB 525 Brooks Road Rome NY 13441-4505				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2019-019	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This research analyzed, evaluated, and characterized the computing and energy requirements for next-generation autonomous systems of significance to DoD's mission-essential tasks and thus investigated the effective and efficient computational intelligence approaches capable of supporting desired autonomy. This assessment will help determine the processing flow of an autonomous system from the cognitive perspectives, as well as the desired performance and energy requirements from the computing perspectives. Specifically, this study first outlined the necessary cognitive primitives and processing flow for a flexible autonomous system capable of real-time problem solving. Then, it focused on the autonomous target tracking problem and explored multiple computational intelligence methods, including artificial neural network (ANN), reservoir computing (RC), and deep learning (DL) architectures, to achieve the desired autonomy. Third, it investigated the computational characteristics of those intelligence models, assessed the performance metrics in terms of accuracy, speed, and energy consumption, characterized performance and energy requirements according to the scope of the problem, as well as identified the most suitable solutions fitting into the cognitive processing flow. Finally, it explored bio-inspired dynamic ensembles of reservoir networks for multiple pattern recognition, category learning driven classification network, and evolutionary adaptation of reservoir network optimization.					
15. SUBJECT TERMS Dynamical Neuromorphic Computing, Dynamical Neuromorphic Computing Hardware, Neuromorphic Computing Benchmarks, Neuromorphic Computing Metrics, Dynamical Neuromorphic Computing Techniques, Dynamical Neuromorphic Computing Neural Networks					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			THOMAS RENZ
U	U	U	UU	99	19b. TELEPHONE NUMBER (Include area code) NA

TABLE OF CONTENT

Section	Page
LIST OF FIGURES	iii
LIST OF TABLES	vi
1.0 SUMMARY	1
2.0 INTRODUCTION	2
2.1 Task 1: Cognitive Process for Real-Time Autonomous Problem Solving	2
2.2 Task 2: Performance Assessment and Characterization of Computational Intelligence Approaches in Autonomous Target Tracking	2
2.3 Task 3: Dynamic Ensembles of Reservoir Networks for Multi-Object Pattern Recognition	4
2.4 Task 4: Human Category Learning Inspired Classification Network	4
2.5 Task 5: Evolutionary Adaptation for Reservoir Network Optimization	5
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES	6
3.1 Task 1: Cognitive Process for Real-Time Autonomous Problem Solving	6
3.1.1 Dimensions of Current Research.	6
3.1.2 Different Paradigms of Cognition.	7
3.1.3 Cognitive Architectures.	8
3.2 Task 2: Performance Assessment and Characterization of Computational Intelligence Approaches in Autonomous Target Tracking	13
3.2.1 Key Parameters of Reservoir Network.	14
3.2.2 Existing Optimization Methods.	15
3.2.3 Reservoir Computing Based Object Tracker	19
3.2.4 Tracking with the Combination of RC and AE.	23
3.3 Task 3: Dynamic Ensembles of Reservoir Networks for Multi-Object Pattern Recognition	26
3.3.1 RC with Multiple Reservoirs	28
3.3.2 Dynamic Reservoir Ensemble Model Based on Genetic Algorithm	31
3.4 Task 4: Human Category Learning Inspired Classification Network	34
3.4.1 Network Architecture	34
3.4.2 Training Strategy.	35
3.5 Task 5: Evolutionary Adaptation for Reservoir Network Optimization	35
3.5.1 Synaptic Adaption Based on Synaptic Plasticity.	36
3.5.2 Structural Evolution Based on Genetic Algorithm.	38
3.5.3 Neuronal Plasticity Inspired Reservoir Ensemble Optimization	39

3.5.4	Interpretive Reservoir: A Preliminary Study on The Association Between Artificial Neural Network and Biological Neural Network	45
3.5.5	Rationale	46
4.0	RESULTS AND DISCUSSION	52
4.1	Task 2: Performance Assessment and Characterization of Computational Intelligence Approaches in Autonomous Target Tracking.....	52
4.1.1	Result Comparison Among RC-based, SDAE-based and CNN-based Trackers.	52
4.1.2	Speedup on GPU Using Reservoir Computing Based Object Tracker.....	55
4.1.3	Tracking Results of the Combined Method (RCAET)	55
4.2	Task 3: Dynamic Ensembles of Reservoir Networks for Multi-Object Pattern Recognition	58
4.3	Task 4: Human Category Learning Inspired Classification Network.....	61
4.3.1	Experiment Setup.....	61
4.3.2	Results.....	62
4.4	Task 5: Evolutionary Adaptation for Reservoir Network Optimization.....	64
4.4.1	Results of Principal Neuronal Reinforcement	64
4.4.2	Neuronal Plasticity Inspired Reservoir Ensemble Optimization	67
4.4.3	Interpretive Reservoir: A Preliminary Study on The Association Between Artificial Neural Network and Biological Neural Network	77
5.0	CONCLUSIONS.....	82
6.0	REFERENCES	83
7.0	LIST OF ACRONYMS	88

LIST OF FIGURES

Figure	Page
Figure 1 Errors for Output 1 Given Varying Connectivity and Number of Neurons (100-500) ..	16
Figure 2 Errors for Output 1 Given Varying Connectivity and Number of Neurons (600-900) ..	16
Figure 3 Errors for Output 2 Given Varying Connectivity and Number of Neurons (100-500) ..	17
Figure 4 Errors for Output 2 Given Varying Connectivity and Number of Neurons (600-900) ..	17
Figure 5 The Structure of Conventional RC Network	19
Figure 6 Network for Online Tracking	20
Figure 7 First Frame in Video Clip Used for Initialization	22
Figure 8 The Virtual and Actual Trajectories.	24
Figure 9 Prediction Result of RC Predictor (left: x, y coordinates; right: trajectory)	24
Figure 10 (a) Structure with Multiple Input Layers, Single Reservoir and Single Output Layer; (b) Structure with Single Input Layer, Single Reservoir and Multiple Output Layers.	29
Figure 11 Hierarchical Structure with 1N-MR-MO	30
Figure 12 Flat Structure with 1N-MR-MO	30
Figure 13 MN-MR-1O Structure	30
Figure 14 (a) Chromosome of Reservoir Ensemble Model; (b) Corresponding Reservoir Ensemble Structure	33
Figure 15 Echo State Network Based Category Learning Model.....	35
Figure 16 GA Determined Structures 1 (Left) and 2 (Right).....	39
Figure 17 The Shape of Nonlinear Function ϕ and Its Relationship	43
Figure 18 A Diagram of Training Procedures	44
Figure 19 The Spatial Locations of 30 EEG Channels (Electrodes)	48
Figure 20 The ESN like Net Structure Adopted	49
Figure 21 Ground Truth Trajectory vs. Estimated Trajectories of RC-based (left) SDAE-based (middle) and CNN-based (right) trackers	52
Figure 22 Tracking Results of RC-Based Tracker in 9 Selective Frames	53
Figure 23 Tracking Results of SDAE-Based Tracker in 9 Selective Frames	54
Figure 24 Tracking Results of CNN-Based Tracker in 9 Selective Frames	54
Figure 25 Ground Truth vs. Estimated Trajectories of RCAET (left) and SDAE-based Tracker (right)	56

Figure 26 Tracking Results of RCAET in 9 Selective Frames	56
Figure 27 Tracking Results of SDAE-based in 9 Selective Frames	57
Figure 28 Optimization process and best solution of dynamic reservoir ensemble	58
Figure 29 Optimal Reservoir Ensemble Structure Constructed by the Dynamic Reservoir Ensemble Model	59
Figure 30 Testing NRMSE of All Models	59
Figure 31 Performance of Standard Model (a) with 1 I/O Neuron; (b) with 5I-2O Neurons	60
Figure 32 Performance of SRE Model without (a) and with (b) Inter-Reservoir Correlations	60
Figure 33 Performance of DRE (a) Prediction Result of High Temp.; (b) Prediction Result of Low Temp	61
Figure 34 Model Input (a) and Output (b)	62
Figure 35 Test Results on Category 1 (a) and Category 2 (b) for One-time Training.....	63
Figure 36 Test Results on Category 1 (a) and Category 2 (b) for Iterative Training.	64
Figure 37 Comparison of NRMS Error Pre/Post-Retraining Performance	65
Figure 38 Trained ESN Performance from Randomized Initialization	65
Figure 39 Retrained ESN Performance from Updated Weights.....	66
Figure 40 NRMSE Pre/Post-Retraining for NARMA Dataset	66
Figure 41 Comparison of NRMS Error Pre/Post-Retraining Performance	67
Figure 42 Output 1 and 2 Optimized with GA (40 neuron).....	69
Figure 43 Output 1 and 2 Optimized with GA and PNR (40 neuron)	70
Figure 44 Reservoir Ensemble Structure Constructed by DRE Process.....	72
Figure 45 Color Map of Anti-Oja	73
Figure 46 Learning Trend of Anti-Oja.....	73
Figure 47 Hebbian Learning's Best Case for Each Decay Rate.....	74
Figure 48 Color Map of Hebbian Learning (decay rate = 0)	74
Figure 49 Learning Trend of Hebbian Learning (decay rate = 0).....	75
Figure 50 BCM's Best Case for Each Tau	75
Figure 51 Color Map of BCM (tau = 260).....	76
Figure 52 Learning Trend of BCM (tau = 260)	76
Figure 53 The Best Improvements of All Learning Rules.....	77
Figure 54 The Avg. Correlation Coefficients on Input Term (Scenario I)	78

Figure 55 The Avg. Correlation Coefficients on Internal Weights (Scenario I)	79
Figure 56 The Avg. Correlation Coefficients on Input Term (Scenario II).....	79
Figure 57 The Avg. Correlation Coefficients on Internal Weights (Scenario II)	80
Figure 58 PDF of Correlation Coefficients for Input Term (left) and Internal Weights (right) (Scenario I & II).....	80
Figure 59 Avg. Correlation Coefficients for the Input Term (Scenario III)	81
Figure 60 Avg. Correlation Coefficients for the Internal Weights (Scenario III)	81

LIST OF TABLES

Table	Page
Table 1 Two-Dimensional Categorization of Cognition Researches	7
Table 2 Configuration of RC-Based Tracker.....	21
Table 3 Configuration of SDAE-Based Tracker.....	21
Table 4 Configuration of CNN-Based Tracker.....	21
Table 5 Network Configurations of SDAE and Online Tracking Network.....	25
Table 6 All possible configurations of RC structures.....	28
Table 7 MSE of Each Removal for Structure 1 (40 neurons for each reservoir).....	40
Table 8 MSE of Each Removal for Structure 2 (40 neurons for each reservoir).....	40
Table 9 MSE of Each Removal for Structure 3(100 neurons for each reservoir).....	40
Table 10 Computational Efficiency of Three Trackers on CPU.....	53
Table 11 Speedup of GPU Compared with CPU.....	55
Table 12 Computational Efficiency of Three Trackers on CPU.....	57
Table 13 Test Errors (NRMSE) on Both Categories for One-time Training and Iterative Training	64
Table 14 Average NRMSE Over 100-900 Neurons After Plasticity Rule (Toolbox Dataset).....	67
Table 15 Average NRMSE Over 100-900 Neurons After Plasticity Rule (NARMA Dataset)....	67
Table 16 Strengthening Principal Neurons with Respect to Ensemble (40 neurons per reservoir) - Instance 1.....	68
Table 17 Strengthening Principal Neurons with Respect to Ensemble (40 neurons per reservoir) - Instance 2.....	68
Table 18 Strengthening Principal Neurons with Respect to Ensemble (100 neurons per reservoir)69	69
Table 19 Strengthening Principal Neuron with Respect to Each Reservoir (40 neurons per reservoir)	70
Table 20 Strengthening Principal Neuron with Respect to Each Reservoir (100 neurons per reservoir)	70
Table 21 Strengthening Principal Neurons within Dominant Reservoirs (40 neurons per reservoir)	71
Table 22 Strengthening Principal Neurons within Dominant Reservoirs (100 neurons per reservoir)	71

Table 23 Strengthen Connections to all Neurons within Dominant Reservoir (40 neurons per reservoir)	71
Table 24 Strengthen Connections to all Neurons within Dominant Reservoir (100 neurons per reservoir)	71
Table 25 The Best Performance for Each Rule	77

1.0 SUMMARY

Although we have seen the von Neumann model's influence on some of today's high-performance computers, the principles the model espouses are not adequate for solving many problems of great theoretical and practical importance. Lots of real-life intelligent problems cannot meet the conditions of von Neumann model (e.g., accurate data and well-defined algorithms), which makes conventional computing architectures less effective in capturing the complexity of the problem. In this study, we investigated several emerging, neuromorphic computing paradigms, which will play key roles in supporting next-generation autonomy.

The objective of this research was to analyze, evaluate, and characterize the computing and energy requirements for next-generation autonomous systems of significance to DoD's mission-essential tasks, and thus to investigate the effective and efficient computational intelligence approaches capable of supporting desired autonomy. This assessment will help determine the processing flow of an autonomous system from the cognitive perspectives, as well as the desired performance and energy requirements from the computing perspectives.

Specifically, in this project, we outlined the necessary cognitive primitives and processing flow for a flexible autonomous system capable of real-time problem solving. Then, we focused on the autonomous target tracking problem and explored multiple computational intelligence methods, including artificial neural network (ANN), reservoir computing (RC), and deep learning (DL) architectures, to achieve the desired autonomy. Third, we investigated the computational characteristics of those intelligence models, assessed the performance metrics in terms of accuracy, speed, and energy consumption, characterized performance and energy requirements according to the scope of problem, as well as identified the most suitable solutions fitting into the cognitive processing flow. Finally, we explored bio-inspired dynamic ensembles of reservoir networks for multiple pattern recognition, category learning driven classification network, and evolutionary adaptation of reservoir network optimization.

2.0 INTRODUCTION

2.1 Task 1: Cognitive Process for Real-Time Autonomous Problem Solving

The term “cognition” dates back to the 15th century when it meant “thinking and awareness” and mainly focused on human minds in philosophy. Humans never stop exploring the way our brain works. Cognition or cognitive processes are analyzed from different perspectives and in different contexts, in linguistics, education, neurology, psychology, philosophy, computer science, etc. Generally speaking, cognitive systems are either natural or artificial systems which may acquire some or all of the capacities of sensing, data processing, learning, reasoning, planning, decision making, acting and representing, etc.

In the 20th century, there was remarkable growth and development of computer science and engineering, which provided powerful tools and methods to research related to cognition. Since then, significant efforts have been made to simulate the capabilities of human minds and to achieve expected properties of artificial or synthetic intelligence using computational technologies combined with neuroscience, psychology and some other disciplines. For instance, researchers have been working on a variety of methods like machine learning and neural networks to achieve artificial cognitive abilities, and there is no doubt that those methods and algorithms play an important role in cognitive systems. Cognitive systems and the related problems form a broad topic. Excess focus on high payoff but limited scope capabilities can lead to getting lost in pursuing the improvement of some specific methods without being aware of that such methods may not always be the most optimal one, or not even the right one to achieve broad cognition. Focusing on specific capabilities is a bottom-up process and follows reductionism at the methodology level. This can indeed achieve better performance for a particular task but may also lose efficiency and breadth in the long run.

In order to address the aforementioned problems, one reasonable approach is to view from a higher level and address things in a top-down manner. This means that we need to have overall planning. What are the essential capabilities of the sub-systems? What are the potential relations among those sub-systems? For each sub-system, what kind of methods and technologies can be adopted to achieve the capability? What are the applicable conditions for each method? This process follows holistic thought at the methodology level.

In this task, we mainly focused on artificial cognitive systems and figured out a general picture of cognitive studies, analyzed from different perspectives and different levels, summarized existing methods and models, understood the state of the art, and proposed a comprehensive guideline in this area from the systematic perspective.

2.2 Task 2: Performance Assessment and Characterization of Computational Intelligence Approaches in Autonomous Target Tracking

As one of the most critical missions for cognitive recognition, recognizing and tracking targets in images/videos from battlefield or search-and-rescue (SAR) unmanned aerial vehicles (UAVs) have been a challenging problem due to various types of image distortion. Moreover, the significantly high computational overhead of existing image/video processing techniques and the limited computing resources available onboard UAVs require most of the processing tasks to be performed in an off-line manner. To achieve fast and autonomous target identification on UAVs, we investigated several computational intelligence approaches that can fulfill real-time processing requirements and evaluated their performance.

Deep Neural Network Approach: Recently, inspired by the mammalian visual system where a simple algorithm is invoked in the neocortex through a recursive hierarchy [1, 2], deep neural network (DNN) approaches have been extensively investigated and advanced for classification and categorization problems [3, 4], given the fact that DNNs can effectively model higher level feature representations. In our research, deep learning architectures-based target tracking approaches were investigated and developed, including the convolutional neural network (CNN) and stacked denoising autoencoder (SDAE). Our initial results showed the superior performance advantages of deep learning architectures in accurately recognizing the targets out of the background, and particularly in precisely tracking multiple moving targets. For instance, the average variation errors of the three moving targets have remained at a stable level for over 80 video frames, that is, the tracking routes consistently match the ground-truth trajectories.

Most DNNs follow the multi-stage Hubel-Wiesel architecture, which contains a hierarchy of layers consisting of filtering, non-linearity and pooling stages. Given the dramatic complexity of those DNN processes, most of existing solutions are hardly applicable to real-time applications with the simultaneous restrictions of extremely limited computational capability and energy efficiency. As we anticipated, the deep architectures demand a relatively high level of computation and suffers from the challenges of limited labeled training data sets.

Reservoir Computing Approach: Given the limitations of the aforementioned approaches, reservoir computing is a promising alternative, because of its superior nature in terms of higher-level feature representation (reservoir), temporal state information (recurrent network), and simple training mechanism (one readout layer to be trained). Following our previous efforts, we investigated a RC-based approach for autonomous target tracking. It is clear that the characteristics of the reservoir network, in terms of number of neurons, topology, and connectivity, play critical roles in determining the performance of the entire system. However, most of the previous research in reservoir computing simply chose a popular topological structure and the number of neurons according to experiences. Little research has shown a comprehensive methodology for selecting the most effective and appropriate reservoir network towards the target application [5]. Thus, in this research, we sought to investigate the potential impacts of changing the reservoir network. Specifically, we used the popular random connectivity as a baseline and varied the amount of connectivity. Lowering the average degree of connectivity decreased sensitivity, while unfortunately; also decreasing the strength the network has in representability and, in the persistence of the signal. (That is, a low degree of connectivity causes the activity to die down quickly because of the lack of feedback and thus the network cannot recognize an “older” input signal.) On the other hand, a higher connectivity will give a larger set of “filters” that separate signals, but also make it more sensitive to changes. Moreover, we also explored other topological structures.

Performance Benchmarking and Characterization: Our primary goal was to explore and understand the rationale and characteristics of multiple computational intelligence approaches and the corresponding solutions in addressing the autonomous target tracking problem. Accordingly, we assessed the performance metrics of those solutions, in terms of accuracy, speed, and energy consumption, characterized their respective performance and energy requirements according to the scope of problem, and identified the most suitable solutions fitting into the cognitive processing flow. Specifically, a cross-platform generic evaluation was performed:

- **Cognitive Accuracy:** The capability of correctly recognizing and tracking the target(s);
- **Computational Demands:** The required execution time, computing platform and power consumption;

- **Scalability:** The estimated changes of accuracy and computational demands, corresponding to different performance requirements, and available resources.

2.3 Task 3: Dynamic Ensembles of Reservoir Networks for Multi-Object Pattern Recognition

Motivated by the natural principles of local neural adaptation and development in cognitive neuroscience, artificial reservoirs whose internal structure can be closely represented [5] as cortical microcolumns (or cortical template) in the brain have been shown to be able to represent rich and complex neural dynamics and perform well when involving temporal information [6, 7]. However, in the practice of conventional reservoir computing paradigm, a single reservoir computing model is not omnipotent. As shown in [8], it was barely possible for a single reservoir to be trained to work as a multiple superimposed oscillator, even when the function consists of only two sine waves (with different phases). Some observations rooted in neurobiological studies have shown that cortical neural networks have a distinctive modular and laminar structure which provides powerful computational functions [6]. Inspired by this specific neuroscience study, in this research, we aimed to develop a new model using dynamic ensembles of reservoir computing, which consist of several sub-reservoirs. The dynamic ensemble model can also lay a foundation for future exploration of energy-efficient, modular neuromorphic architectures, where only a selected set of sub-networks are active for a certain type of cognitive task, similar to the regional cerebral blood flow distributions in the brain for various brain functions.

Lateral inhibition is a well-known mechanism, which was originally discovered in neurobiological systems including the retina, cochlear, and pressure sensitive nerves in skin. Specially, when a neuron is stimulated, the neural activity of its surrounding region is suppressed via lateral inhibitory synapses. In this way, different neurons would possess various receptive fields, i.e., they will respond to stimulations of different conditions such as orientation, frequency and pressure. Therefore, in this research, the dynamic ensembles of reservoir computing were designed to emulate this kind of neurobiological system through the adaption of synaptic connections.

A dynamic ensemble model of reservoir computing may consist of several sub-reservoirs. For one specific pattern, only one or a few sub-reservoirs would produce a strong response while the other sub-reservoirs will be inhibited and present much less response on the associated pattern. This proposed method would implement the competitive mechanism through dynamic connection routing associated with different input patterns and ever sparser connections between multiple sub-reservoirs, and integrate the internal states of all the sub-reservoirs to output the action potentials. In this way, the correlation of neural dynamics between different sub-reservoirs is reduced, and multiple internal states are thereby generated to cooperatively recognize multiple patterns.

2.4 Task 4: Human Category Learning Inspired Classification Network

Category learning has long been considered as a basic component of human intelligence and a key ability towards concept generalization, which is the foundation of sophisticated thoughts [9-12]. Nevertheless, in the field of cognition, no agreement has been reached on the mechanism of category learning.

In recent years, remarkable advances have been achieved in solving classification tasks from the computational models' side. However, it has been argued that many sophisticated models being proposed and studied seem to gradually deviate from the nature of intelligence and primarily focus

on specific tasks and data mining. More and more efforts have been made in seeking tiny optimization tricks for a specific algorithm rather than looking at the fundamental problems from the true perspectives of cognitive and neural sciences. Emphasis is on the quality and representativeness of the target data rather than focusing on the intrinsic but flexible learning ability/mechanism that can grow to learn the learning algorithm of doing different things by itself. To this end, this effort explored fundamental category learning utilizing the chaotic state of reservoir computing and the unsupervised characteristics of synaptic learning.

In previous tasks, we demonstrated the learning ability of reservoir computing composed of a single reservoir or multiple reservoirs when dealing with either relatively simple problems or some complex problems that may incorporate different types of dynamics. In this task, we investigated the potential of a single-reservoir echo state network (ESN) in learning the categories. To make it more biologically and theoretically plausible, unsupervised Hebbian learning of internal synapses was applied to the reservoir, intending to learn the conjunctions between the features and the boundaries among diverse categories at the same time.

2.5 Task 5: Evolutionary Adaptation for Reservoir Network Optimization

One cannot expect any reasonably robust performance from any autonomous system that is incapable of learning. Furthermore, it seems reasonable to contend that any hand-crafted system will fall short of expectations, and so accommodating some form of continuous improvement cycle will be needed. This improvement cycle may be manually implemented, but we strongly argue that some form of automated evolution will perform better.

It can be useful to identify a data structure in any computerized implementation of a cognitive function that serves as the repository for control of that function. Let us call this data structure, memory. Learning may then be applied to making updates to that memory. Design of such learning systems requires identification of mechanisms for identifying performance errors (to serve as, among other things, the impetus for learning), for deciding what changes to the memory are permitted and desirable, and for testing that any changes considered, are effective for overcoming the observed errors, and in addition, do not contribute to disrupting the well-functioning aspects of performance.

For reservoir computing, a critical part of memory is rooted in the reservoir structure, wherein the intricate connections with recurrent circuits represent complex relations of current states and past experience. However, the original concept of RC uses a fixed randomly initialized reservoir to reduce the training overhead. It means that the internal “memory” of the reservoir is randomly created without further update. Lacking the effective internal adaption and evolution, the learning ability of reservoir is more or less limited. For this reason, it is argued that evolving the reservoirs in a dynamic manner can improve performance on a given application. One train of thought is to find an optimal reservoir given a certain task. This idea is based on the fact that the performance of randomly chosen reservoirs forms a distribution. Given some search algorithms, such as genetic algorithms (GAs), it is thus easy to perform better than average by choosing the right reservoir. Another possibility is to start out with a large reservoir and to enhance (or prune away) principal (or redundant) nodes given a certain task. In this task, we investigated both approaches and figured out an effective way to enable the evolutionary adaption of reservoirs.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Task 1: Cognitive Process for Real-Time Autonomous Problem Solving

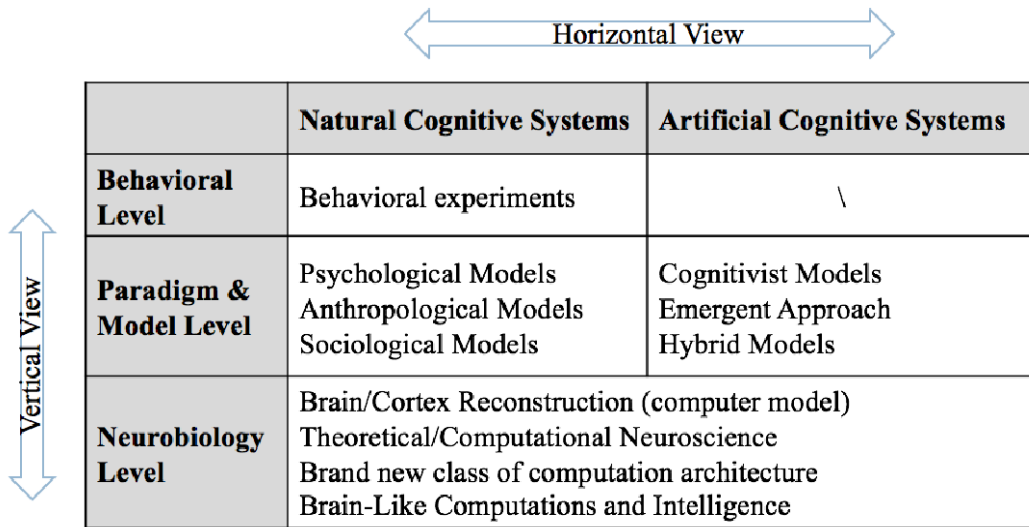
Over the past few decades, research on cognition and cognitive systems has been well addressed in various contexts. No matter in psychology, ethology, anthropology, sociology, or in neuroscience and computer science, unremitting research efforts have been made to keep pushing the development of cognitive science forward step by step. A complete understanding of the mind/brain cannot be attained by studying at one certain level. In this study, a two-dimensional categorization of cognition related research is proposed, with the goal of providing a clearer view of the overall picture of the human cognitive processing.

3.1.1 Dimensions of Current Research.

As shown in Table 1, the research areas are divided into two dimensions: the horizontal dimension and the vertical dimension. The horizontal dimension represents different objects of study, which mainly contains natural cognitive systems (animals and human) and artificial cognitive systems (artificial intelligence). The vertical dimension represents different levels of study, ranging from behavioral level (surface level) to neurobiology level (deeper level).

- **Behavioral Level:** The goals at this level are usually to describe the constitution and process of cognition by observing experimental outcomes (in ethology, anthropology, sociology, and sometimes psychology), like psychophysical responses and reaction time. As a result, they are usually in the domain of natural cognitive systems.
- **Paradigms and Model Level:** Studies are conducted in both natural systems and artificial systems, pursuing extracted general models of cognition. Research work on natural cognitive systems has been performed from the perspective of psychology, anthropology and sociology based on experimental data and results in the behavior level. Adopting these as a foundation and integrating knowledge from neurology and computer science, more unified theories are integrated to achieve cognitive behaviors in artificial systems. There are three paradigms of artificial cognitive systems: cognitivist approach, emergent approach and hybrid approach. They will be addressed in detail in the following discussion.
- **Neurobiology Level:** When it comes to the neurobiology level, the boundary between the natural and the artificial cognitive systems is rather ambiguous. The two parts are so highly dependent that none of the work is launched only for a single type of system. The goal of this stage is to reveal the real mechanism of human mind leveraging brain imaging, neurobiological and many other computational technologies, so as to create human-level artificial systems and next generation neural computing technologies.

Table 1 Two-Dimensional Categorization of Cognition Researches



	Natural Cognitive Systems	Artificial Cognitive Systems
Behavioral Level	Behavioral experiments	\
Paradigm & Model Level	Psychological Models Anthropological Models Sociological Models	Cognitivist Models Emergent Approach Hybrid Models
Neurobiology Level	Brain/Cortex Reconstruction (computer model) Theoretical/Computational Neuroscience Brand new class of computation architecture Brain-Like Computations and Intelligence	

3.1.2 Different Paradigms of Cognition.

In order to clearly understand and be able to take advantage of the concept of cognition, we must first discuss and answer the following questions. What motivates the cognition? How are the actions selected? How are knowledge and experiences represented? How is perception dealt with and how is the information processed? In which way will the actions be evaluated and selected? What drives the development of the system? Undoubtedly, there are different ways to answer those questions, based on people’s position within particular fields. Each position takes significant attitudes from different perspectives on the nature of cognition, the operation of a cognitive system, the mechanism of achieving cognition and the way of analysis and synthesis [13, 14].

It has been widely accepted that the stances of cognition go into two broad classes: the *cognitivist approach* and the *emergent approach*. The most significant difference lies in their distinct perspectives on knowledge representation and processing, where the cognitivist approach is based on symbolic rule systems and the emergent approach relies on emergent properties of self-organized processing units [15]. In addition to this, there are still more distinctions to differentiate these two types of approaches.

The cognitivist proponents view cognition in a functional way in which cognition and mind are analogous to computers. That view comes from both systems using symbolic representations. The emergent approach, which embraces connectionist, dynamical, and enactive systems, and is more likely to treat cognition as an emergent, self-organizing, and dynamical process [16-18].

David Vernon, et al. [19] proposed a comparison between the two paradigms. However, his statements seem obscure in concepts. In this report, the distinctions are compared in a more straightforward way. We will now take a brief look at some crucial differences between the cognitivist approach and the emergent approach.

- ***Symbolic vs. Sub-symbolic:*** The debate over this topic is the fundamental contradiction between cognitivism and emergence. The contrast can further be divided into three aspects. a) *Representation:* Cognitivist systems use symbol tokens to represent either internal or external events (knowledge and situations). This kind of representations is usually provided by human designers and thus can be directly read and understood by humans. Representations of emergent systems are usually self-produced global system states encoded in the dynamic distributed network structures. These representations are not directly readable by humans. b) *Computational operation.* The operations in a cognitivist system are typically rule-based, typically but not necessarily in a sequential manner. In contrast, emergent systems specify no such priori rules and rely on interacting of distributed processing units (nodes in networks). c) *Perception and Action.* In cognitivist systems, perception occurs by extracting symbolic representation from sensory data so as to make the symbolic representation of the external world. Actions in turn, are selected through a consequent processing of internal symbolic representations. For emergent systems, perception is achieved in the form of system state changes caused by external perturbations, while actions are the system generated perturbations towards environment.
- ***Fixed vs. Dynamic:*** The cognitivist approach has its root in functionalism, which determines that aspects of cognition are constant over time and so are the components of structure. This demonstrates that the cognitivist systems represent the fixed part of cognition. This is the reason why they cannot accomplish anything in their own right and need prior knowledge to perform given tasks. For the emergent approach, which is based on connectionism and self-organization, mechanisms for perception, action, adaptation, anticipation, and motivation are embedded, so that dynamic processes of self-production, self-maintenance, and self-development over the system's lifetime can be enabled.
- ***Independency vs. Dependency of Cognition:*** Due to the foundation of functionalism and the feature of fixed structure, cognition is independent of the architecture. On the contrary, the properties of emergence and dynamical processes cause physical instantiation of the emergent systems be closely tied with the cognitive process. In other words, cognition in the emergent approach is agent-dependent.
- ***New knowledge vs. New Dynamics:*** This distinction is determined by their forms of adaption. For cognitivism, adaption performs in the manner of acquisition of new knowledge, while for the emergent approach, adaption implies a structural alteration or re-organization to affect a new set of dynamics [20]. Ultimately, the difference on adaption is also determined by the first two comparisons.

3.1.3 Cognitive Architectures.

Although in the 20th century, researchers had gained supportive theories, data, and experiences about cognition from the perspectives of cognitive ethology and cognitive psychology, it had always been a difficult task to integrate existing knowledge towards the implementation of cognitive systems. However, it was seen as an important step for cognitive science when Newell published his work in 1982 [21]. This is believed to be the origin of the term cognitive architecture. Since then, research on cognitive architectures became an important trend in the following decades and a number of famous works, which will be introduced later, came on the scene.

A cognitive architecture specifies the infrastructure or overall arrangement of a cognitive (intelligent) system. It is worth noting that what a cognitive architecture includes should only be

those aspects of a cognitive agent that are constant over time and across different application domains [22]. In other words, models and knowledge are applied within some specific cognitive architecture but can't be seen as the part of the architecture because of they are user-defined and time-variant. Moreover, distinct cognitive architectures may differ in the assumptions they make. One simple example may briefly illustrate the idea within the definition of cognitive architectures. Different vehicles may be provided with different shapes or layouts due to different assumptions they were endowed with, just like different cognitive architectures. For one specific type, no matter how the decoration or accessories may change for different purposes, its basic architecture will always remain the same.

Research on cognitive architectures is important because it is no longer driven by seeking solutions for specific tasks, instead it is driven by the objective of unified theories of cognition [23]. Even though cognitive architectures may differ in paradigms and models, they take the same effort to achieve the synthesis of a broad range of cognitive issues by integrating interdisciplinary knowledge from psychology, philosophy, neurology, and computer science.

Based on different paradigms of cognition, cognitive architectures also include cognitivist architectures and emergent architectures. In addition, a hybrid type of architectures is developed to combine the former two types. Due to the large number of cognitive architectures developed, we are not able to present and discuss all of them. In this report, we will introduce some well-known cognitive architectures which cover all three paradigms and give a brief discuss about their features and abilities.

3.1.3.1 Soar

Soar, which was first proposed in the early 1980s, is one of the first cognitive architectures [24, 25] and has been well developed. The mission of Soar is to support all capabilities of general intelligent agents. To accomplish this big goal, it is endowed with a wide range of problem solving models as well as the ability to learn, represent and use tasks related knowledge [26]. It has been widely used on verity of tasks in artificial intelligence and cognitive science.

Under the guidance that a cognitive architecture must help produce cognitive behavior, Newell et al. abstracted 6 common characteristics from everyday behaviors [23]:

- It is goal-oriented;
- It takes place in a rich, complex, detailed environment;
- It requires a large amount of knowledge;
- It requires the use of symbols and abstractions;
- It is flexible, and a function of the environment;
- It requires learning from the environment and experience.

To reflect these characteristics, they picked memory, perception, action, and cognition as primitive features, connect them with a mechanism and formed the Soar cognitive architecture. Interaction with the environment is enabled by linking the perception and action with a decision-making mechanism. Information from the environment is represented in the working memory (WM) through perception. The knowledge in WM can then be used to match and retrieve relevant knowledge from long term memory (LTM) to WM according to a set of production rules. During this procedure, procedural knowledge plays a key role of controlling behaviors and is mapped directly onto the operator knowledge in WM, while semantic and episodic knowledge are employed as supplements in the procedure of solving problems. Finally, preferences are made as recommendations to evaluate and select operators and trigger the steps of actions, the environment

can be influenced as well. It is in such way that a fixed decision cycle is formed as the essence of problem solving mechanism: input, elaboration, decision, application, and output.

However, it may occur that no appropriate knowledge can be found from the LTM. Applicable operators will not be found and Soar will make no decisions. An impasse is encountered [24]. The situation is not a waste of time, for it provides Soar with an opportunity to learn from its failure [27]. This is a brilliant mechanism built inside the architecture. It is inspired by the human's behavior of "practice": the act of doing the task strengthens our ability to do the task in the future.

In Soar, there was originally a single learning mechanism called chunking, three more were added later: reinforcement learning, semantic learning, and episodic learning. The chunking mechanism allows the expansion of production rules (in LTM), preventing the repetition of the same impasse. Reinforcement learning gives Soar rewards for successes and punishments for failures, helps to improve future performance. The episodic and semantic learning help to gain knowledge from past experiences for the use of additional cues in operator selection. Each of the learning mechanisms introduces a different source of knowledge. In coordination, they ensure the self-improvement of the system.

3.1.3.2 ACT-R

ACT-R, short for Adaptive Control of Thought – Rational, is another effort to build the unified theories of cognition, although it uses an alternative way of expression, an integrated theory of mind [28]. By putting forward this programming language like cognitive architecture, researchers intend to simulate and understand how human cognition works based on assumptions and facts derived from psychology experiments.

The goal of exploring the organization of human brain and the origin of intelligent behavior makes ACT-R unique in two aspects. First, it focuses on modular decomposition of cognition [19] and achieves cognition in the manner of harmonious integration of the modules. Second and more importantly, it allows researchers to collect quantitative measures of the model and then compare directly with the quantitative measures obtained from human participants, even including neurological data obtained from functional Magnetic Resonance Imaging (fMRI).

The ACT-R architecture comprises 3 main components: modules, buffers and a central production system. A set of modules are employed to process different kinds of information. Generally, they can be divided into two broad types: perceptual-motor modules, which take care of the interface with the real world; and the memory modules which store the knowledge. Five developed modules are included in the above structure. Visual and the manual modules are perceptual-motor modules. The visual module is designed to recognize and detect the position of objects. The manual module is designed to control actuators and perform actions. The declarative module helps to retrieve information from memory. The goal module is important for keeping track of current goals and intentions. [28]

ACT-R accesses its modules through buffers. For each module, a dedicated buffer serves as the interface with that module. The buffers also represent the internal state of ACT-R at the moment. In order to achieve coordinated behaviors, all of the buffers are connected through a central production system, in which the production rules are implemented.

In the architecture, the whole set of components are analogous to the regions of human brain, while the connections and working mechanism are designed on the basis of cognitive psychology. The central production system serves as the basal ganglia in human brain. Within the production system, production rules are divided into matching, selection and execution mapping to the function of striatum, pallidum and thalamus. The goal buffer plays the role of the dorsolateral

prefrontal cortex (DLPFC). The retrieval buffer is mapped to the ventrolateral prefrontal cortex (VLPFC). The manual buffer is associated with the motor and somatosensory cortical areas, which are the two areas responsible for controlling and monitoring hand movements. The visual buffer is identified with the abilities of locating and identifying the object corresponding to dorsal path and ventral path respectively.

ACT-R operates in a mixed manner of parallel processing and serial processing. The buffers first represent the information determined by the external world and internal modules. In this stage, they work simultaneously within and between the modules. Then the production rules will match the state of the buffers and search for a production. On occasion, an estimation and selection process is needed when several productions match the state of the buffers. When a production fires (is executed), the buffers can be updated and thus the state of the system is changed. The matching, selection and execution in the central production system are processed in succession. A production rule in ACT-R corresponds to a specification of a cycle from the cortex, to the basal ganglia, and back again.

There are two levels of serial bottlenecks that the researchers of ACT-R already recognized. The first one, in their own words, the content of any buffer is limited to a “single declarative unit of knowledge”, called a chunk in ACT-R. Thus, at one moment, only a single memory can be retrieved or only a single object can be encoded from the visual field. Second, only a single production can be selected and fired at each cycle. For the second bottleneck, one development has been made, called production compilation. It enables the combination of separate production rules into one. But for the first bottleneck, it’s more like a deliberate design to explain the phenomenon that people are usually not aware of all the information in the visual field but only the object they are currently attending to. A similar thing happens for long-term memory when people try to retrieve something to deal with the facing situation.

However, the biggest potential threat comes along with the basic hypothesis of ACT-R and therefore the basic structure. The high-level separation of buffers and modules in ACT-R is based on the hypothesis that the organization of the human brain conforms with functional specialization. This point of view is consistent with the previous research and practice in psychology and neuroscience to partition the brain into segregated, cortico-striatal-thalamic loops. However, findings seem to challenge this view by showing some interdependence between the regions. Some evidence has been proposed by Giacomo Rizzolatti et al in their research about ventral and dorsal pathways [29, 30].

Anyway, ACT-R is still an excellent cognitive architecture, which has improved over the years and has been widely applied in the areas of education, human-computer interaction, cognitive psychology, and neuroscience.

3.1.3.3 The Subsumption Architecture

The subsumption architecture represents an alternative approach to realize cognitive behavior. Unlike the classical cognitivist architectures (such as Soar and ACT-R, etc.) that decompose the system into specific functional components, the subsumption architecture maps different levels of competence onto a multi-layer structure in an incremental and bottom-up manner. It reflects the idea of emergence and was proposed by Brooks in 1985 [31].

The subsumption architecture has a hierarchical structure organized into layers. Environmental information acquired through sensors is allowed to be accessed by all layers and consequently multiple behaviors can be executed simultaneously (in parallel). Also, each layer is capable of controlling the system by itself. This is the most obvious distinction in topology with the

cognitivist architectures. Each layer corresponds to a certain level of competence and is responsible for pursuing a particular goal. It begins with simple systems that can act effectively in simple circumstances (lower layers), more sophisticated systems (higher layers) are then added incrementally from bottom to top, and the lower layer is designed to be subsumed in the layer of higher level. Based upon the hierarchical structure, the lower layers would behave like a quick response mechanism, allowing the agent to quickly adapt to changes of environment. The higher layers take charge of the whole situation driving the system towards the overall goals. As mentioned above, in order to achieve a higher level of competence, a new layer can be simply added without altering existing layers [27, 32]. The inspiration of the working mechanism more or less comes from the biological nervous system, which is explained in Brook's paper [33]. When a new competence of operating more complex function has been achieved, a new section of brain is developed to manage it, while the old sections are still responsible to perform their original tasks.

One benefit of this structure is that the lower level competence can be well tested and adjusted before the higher ones can be added. This makes the debugging easier to handle and also ensures the stability of the system. The adoption of multiple distributed layers can serve as buffer layers as well, so that the performance won't fall dramatically when facing a huge change in environment. Compared to the symbolic processing employed by Soar and ACT-R (cognitivist architectures), there is no explicit representation of knowledge, no production rules for matching and no central control system. These features meet with the inherent meaning of emergent system. Even though no specific computational nodes or clear structure of network can be found in the subsumption architecture, we still see the form of hierarchical organization which is the important feature in some later artificial neural networks.

3.1.3.4 CLARION

CLARION is the abbreviation for "connectionist learning with adaptive rule induction on-line". Simply from the name we can tell that it is a hybrid cognitive architecture with the learning capability fulfilled by the combination of symbolic structure (chunks and rules) and connectionist module (more specifically neural networks).

The architecture is designed to provide the advantages of cognitivist and emergent architectures by evolving both implicit memories and explicit memories and addressing both top-down learning (from explicit to implicit knowledge) and bottom-up learning (from implicit to explicit knowledge).

The main components of CLARION include four subsystems: the action-centered subsystem (ACS for procedural knowledge), the non-action-centered subsystem (NACS for declarative knowledge), the motivational subsystem (MS) and the metacognitive subsystem (MCS) [34]. Within each subsystem, both implicit and explicit representations are allocated. Implicit knowledge is represented in an emergent way like neural networks and explicit knowledge is represented in a symbolic way by using chunks and rules.

The role of the ACS is to control both external and internal actions. Neural networks (called Action Neural Networks) are used to form the bottom implicit memory. The top explicit memory is made with action rules. The rules can be extracted either from the environment or from the bottom implicit memory.

The role of the NACS subsystem is to maintain general knowledge. Additional neural networks (called Associative Neural Networks) are adopted to build the bottom implicit layer. Knowledge and facts stored in the top explicit memory are further divided into semantic and episodic. The semantic representation corresponds to generalized knowledge, and episodic is applicable to more

specific situations. It is notable that in CLARION, declarative knowledge is not necessarily explicit, but can also exist in the implicit layer.

The MS is provided the duty of forming goals and underlying motivations for all behavior of the system as the name suggests. A boundary is set between explicit goals and implicit motivations, dividing MS into goal structure and drivers. In this case, the combined motivational processes which include both explicit and implicit parts is believed to have better performance on keeping the system sustained, purposeful, focused, and adaptive.

The MCS takes the overall control of other subsystems' processes by monitoring, directing, and modifying their operations. Actions include setting goals for MS and setting parameters as well as changing on-going process in both ACS and NACS.

Based on the four-way division structure, learning algorithms are addressed with both implicit and explicit representation. Implicit knowledge can be gradually accumulated through reinforcement learning, which works in a manner of repeated practice. Explicit knowledge is acquired through hypothesis testing rule learning and bottom-up rule learning (from implicit to explicit knowledge). Instead of evaluating only the rules, CLARION also combines the cumulative reward value computed by reinforcement learning and judge in a weighted sum. Then the most appropriate action can be decided.

3.2 Task 2: Performance Assessment and Characterization of Computational Intelligence Approaches in Autonomous Target Tracking

For a comprehensive understanding of reservoir computing, an extensive literature survey has been conducted to investigate the key parameters associated with general reservoir network architectures that influence the performance, including the activation function, connectivity, and number of neurons. Moreover, the existing popular optimization strategies have also been explored and discussed. Specifically, several examples of changing performance resulting from varying connectivity and number of neurons was implemented and presented.

In the autonomous target tracking task, an RC-based tracking algorithm has been developed and implemented for single object tracking, along with the Softmax classifier. With the goal of assessing and characterizing the performance of multiple computational intelligence approaches, including the RC, SDAE, and CNN, we proposed an object tracking flow involving the RC in a similar way as we did with SDAE and CNN. The experiments show that the RC-based tracker (RCT) could achieve comparable performance as the SDAE- or CNN-based methods. The speedup on GPU platforms using RCT was further evaluated. However, because the current processing flow was developed targeting at the conventional neural network structures, it is not able to fully make use of the superior advantage of RC in terms of dynamic representation of temporal states and behaviors.

Aiming at RC's weakness in directly handling image data, we proposed an RC-compatible, temporal-aware object tracking flow through combining RC network, autoencoder (AE) and particle filter together, trying to leverage RC's advantage in handling time series data. The tracking performance and computational efficiency were further examined and discussed. The object tracking task can be managed by the new method and computational efficiency were largely improved.

3.2.1 Key Parameters of Reservoir Network.

As illustrated in Jaegers' work [8, 35, 36, 54], the initialization of the reservoir plays a large part in the performance of the ESN and this was shown to be the case for the liquid state machine (LSM). It was discussed in [36] that due to the random nature of RC systems, it is by definition that the networks are not optimized. This suggests that the variables within these reservoir base networks can be altered to improve the performance of the system. In this section, the primary variables of interest will be discussed and are listed as follows: activation function, connectivity/weights, and number of neurons.

3.2.1.1 Activation function.

The activation function primarily acts as a method for limiting the output of a neuron. It is also referred to as a squashing function in the way that it "squashes" the range of unbounded input to a bounded range of output. As it has been discussed in [37], different activation functions can have different effects on the performance of neural networks in general, and this would be applicable in the different network types within RC. Commonly used types include the spiking neuron activation, sigmoid function activation and hyperbolic tangent functions. The sigmoid function is often used in the non-saturation regions to allow for dynamic representations within the RC.

As the activation function can be any transfer function which takes an unbounded input and transforms it a bounded output, it becomes obvious that the choice of function becomes non-trivial and requires further research to determine the applicability of different types to different applications.

3.2.1.2 Connectivity and weighted connections.

Connectivity means the number of recurrent connections within the reservoir. In artificial neural networks, it is common for the connections to be fully connected from layer to layer. Even structure specific custom network types such as convolutional neural networks, the number of connections from layer to layer is defined, and thus connectivity has no applicable meaning. However, this is not the case in the RC network structure where instead the connections are initialized randomly and sparsely from a preset connectivity. As it has been consistently shown, the connectivity decided upon at the initialization of the RC network affects the complexity of the reservoir and has a direct impact on its performance [38, 39]. Too many connections may affect the ESN's ability to be successfully trained, because an excessive connectivity may result in a stronger coupling effect of internal neuronal states and reduce the diversity of the neuronal states in the reservoir. This suggests that the alteration of the number of interconnections within the reservoir in RC networks has great potential when optimizing the performance of said networks. It is common in practice to maintain a rather sparse connectivity, usually between 0.01 and 0.2 [40].

In any neural network, the system is constructed by connections that pass signals between neurons. These connections are weighted so that the signals being passed through the connections can be scaled and adjusted. In traditional neural networks, these weighted connections are altered during the training phase so the overall system can more closely model the system of interest. This is again not the case in RC networks, where the weights within the reservoir remain constant once they have been initialized. Whilst if initialized with consideration, the network functions well in modeling many time series applications [39-43], it is clear that with the update of these weights it would be possible to improve the network performance.

3.2.1.3 Reservoir size.

Reservoir size represents the number of internal neurons within the reservoir. As with regular artificial neural networks, the number of neurons represents the complexity of the signal it is able to model, and the amount of feature the network is able to capture from the input signal. Deciding the number of neurons to have within the network becomes a non-trivial task when considering performance of the network vs. the computational complexity. Even in cases where computational power is not an issue, over fitting with a large number of neurons can be an issue faced in RC systems. Therefore, this is probably the most significant and challenging factor for RC modeling, because the size of the reservoir determines the maximum number of synaptic connections in the reservoir network. Many studies have shown that the reservoir size represents the potential memory capacity [38]. Reservoirs that are too small will result in the RC not being able to accurately model the task while too large ones will encounter issues with over fitting of the data [8].

3.2.1.4 Spectral radius.

Spectral radius is the length of “memory” the reservoir is capable of retaining. The larger the spectral radius, the longer the past inputs to the RC are able to affect the current output. To ensure the stability of neuronal dynamics, the size of the spectral radius, which is the largest absolute eigenvalue of the connection weight matrix, must be properly controlled [38].

Figures 1 - 4 show the effect of the connectivity on the error of the system for output 1 and output 2. Figure 1 and Figure 2 show the errors for output 1 with networks ranging from 100 to 900 at connectivity between 0.1 to 1 and Figure 3 and Figure 4 show the same for output 2. We can clearly see that for network sizes from 100 to 500 there is a clear increase in error rate when the connectivity is increased. This is observed in Figure 1 and Figure 3 for both outputs. This effect is not seen in Figure 2 and Figure 4. For network sizes of 600 and larger, it can be seen that the error is large (greater than 1) and that there is no clear relationship to the connectivity. This is likely due to the fact that the systems were not converged, and therefore the errors are large regardless of the connectivity.

3.2.2 Existing Optimization Methods.

Having demonstrated the vast nature of the initialization space of RC networks, it is manifest that a brute force method for determining the optimal parameters for the reservoir is impractical. At the same time, due to the limited understanding and lack of a rigorous mathematical representation of the randomized reservoir, directed optimization algorithms are difficult to implement. For this reason, optimization methods which do not require an explicit understanding of the mechanics of the RC networks but is still able to search the state space of the parameters to achieve a better performance have been explored. Below, some of the optimization methods more commonly used with RC networks are outlined.

3.2.2.1 Neural plasticity.

Since neural networks are biologically inspired systems, it makes sense to draw inspiration for their adaptation from biology as well. Neural plasticity describes the change to brain architectures in response to development and learning. This process can be largely described by two main types of plasticity: synaptic and intrinsic plasticity [44].

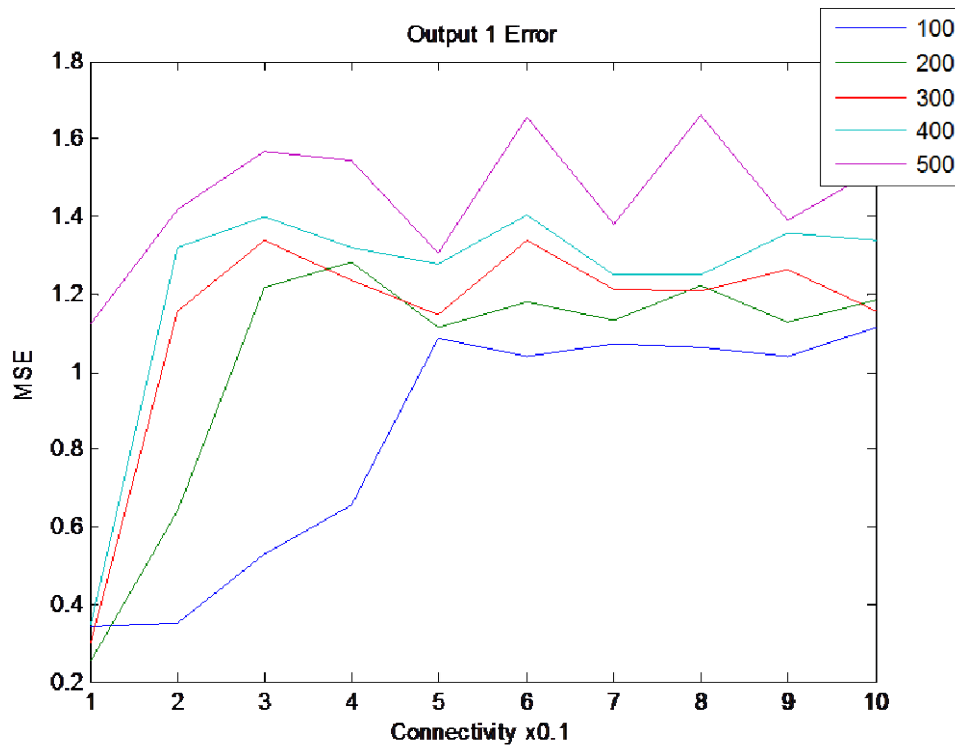


Figure 1 Errors for Output 1 Given Varying Connectivity and Number of Neurons (100-500)

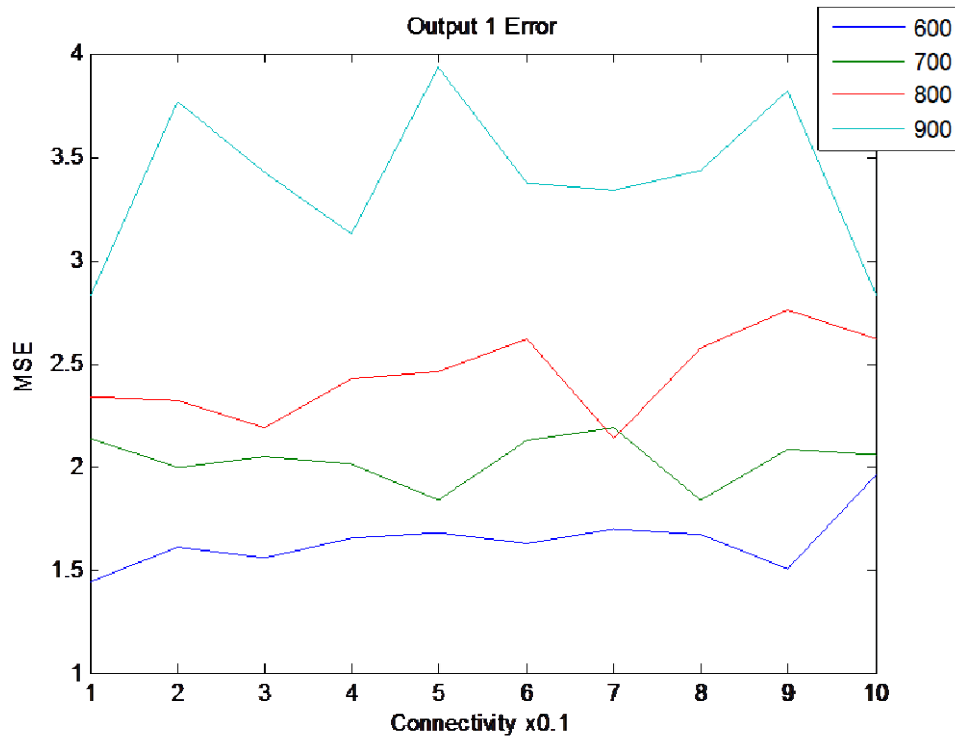


Figure 2 Errors for Output 1 Given Varying Connectivity and Number of Neurons (600-900)

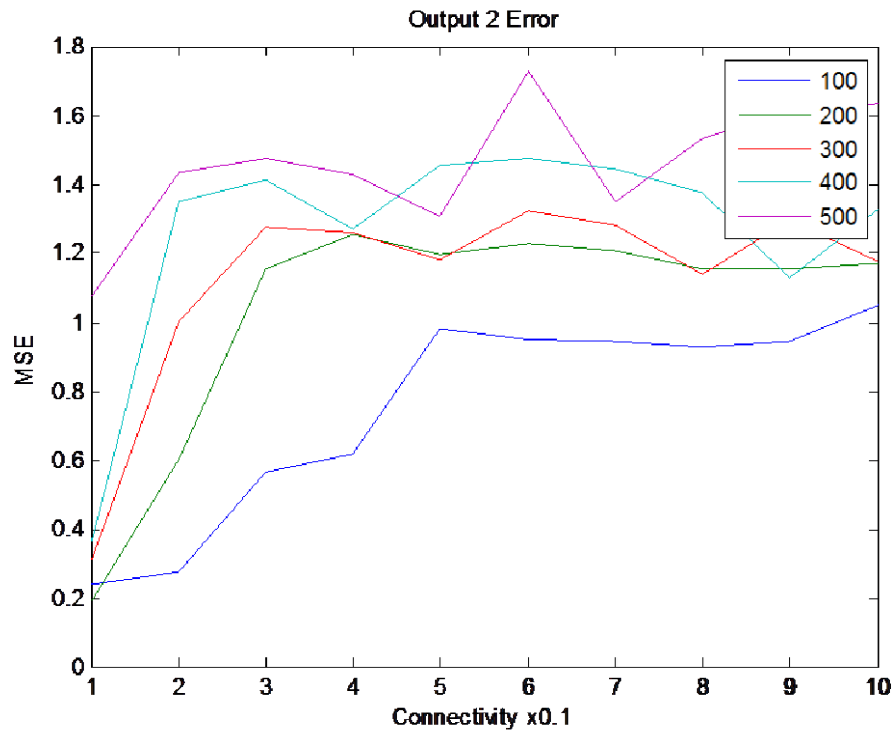


Figure 3 Errors for Output 2 Given Varying Connectivity and Number of Neurons (100-500)

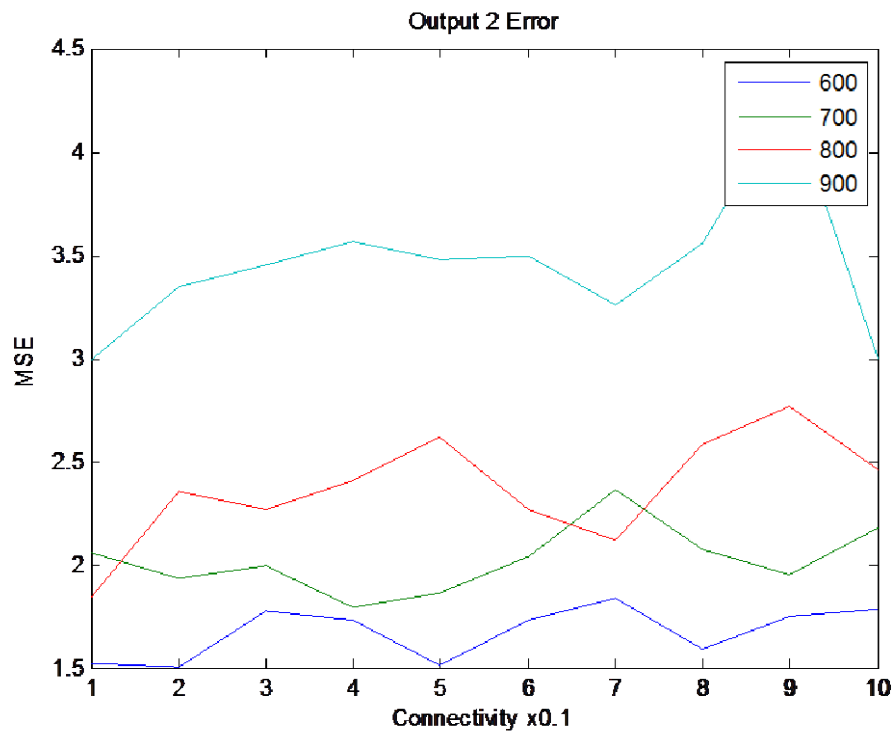


Figure 4 Errors for Output 2 Given Varying Connectivity and Number of Neurons (600-900)

- **Synaptic Plasticity.** Synaptic plasticity refers to the strengthening or weakening of a synapse connection between neurons due to their activity [45]. In neural networks, the strength of the synapse is analogous to the weights of the connections. One popular model that is repeatedly visited in the study of self-organization of neural network (NN) weights is Hebbian theory, which postulates that the synaptic strength between two neurons are strengthened when there is a correlation between their activity [46]. This effect has been thoroughly investigated, and has resulted in more accurate models such as the Oja and Bienenstock, Cooper and Munro (BCM) rules [47, 48, 49]. These have been implemented in algorithm form and have been proven to improve the performance of neural networks. In these cases, the algorithms alter the weights of the RC networks depending on the nature of the input data to achieve better results.
- **Intrinsic Plasticity.** Intrinsic plasticity refers to the change in the way a neuron fires in a network. One frequently focused upon model of the Intrinsic plasticity is the long term potential of intrinsic excitability (LTP-IE). This postulates that more frequently firing neurons will become more excitable over time. This concept when compared to artificial neural networks translates into the activation function changing depending on the type of signal which stimulates the system. It has been shown in a number of studies that the intrinsic plasticity is able to improve the performance of RC systems [36, 43, 50].

3.2.2.2 Particle Swarm Algorithm.

This algorithm observes variables as "particles" moving at velocities. The performance of the variables at the current iteration is assessed to obtain the velocities at which the "particles" move. The new positions of the particles represent the values of the new variables and the performance is reassessed. The new positions are kept if the performance has been improved. This type of algorithm has been implemented in many applications of ESNs to deal with the optimization of the connection weights within the networks. The particle swarm algorithm (PSA) has been shown to improve the performance of neural networks in general, and has been used to improve RC networks in many applications.

3.2.2.3 Genetic Algorithms.

Genetic algorithms (GA) alter the construction of neural networks incrementally to achieve an improvement in performance. The general function of a genetic algorithm involves the generation of "genomes" which are used to alter the parameters of interest. The genomes are also "mutated" to provide a degree of randomness, much like biological evolution. The new parameters are applied and the performance of the system is evaluated, and at the same time, the "fitness" of the solution is assessed. In cases where a better fitness is achieved, the solution is kept. The process is repeated by multiple iterations in order to search for better solutions.

There are certain issues with GAs when being applied to NNs. In particular, they are often seen as brute force methods, and therefore are computationally expensive. It was shown that GAs did not perform well with large networks [51]. Another issue is that depending on the type used, the way the network is altered may vary drastically. Investigation has been conducted on altering the weights, the neuron population, the connections, and a varying combination of these, with varying results [52, 53]. However, there has been no consensus on the best practice using GA algorithms.

3.2.3 Reservoir Computing Based Object Tracker

Object tracking has been generally recognized as a very challenging problem in practical application scenarios. In this task, we developed a robust object tracker based on RC networks, so as to evaluate the feasibility of RC in image representation and object tracking. In the RC-based tracker (RCT), a standard RC network plays the role of feature extractor and is used to directly process image inputs. A particle filter helps to generate probable candidates and to identify the object being tracked among them. Softmax regression is adopted as a binary classifier, explicitly distinguishing the object from its background.

3.2.3.1 Feature extractor of RC-based tracker.

In this task, our objective is to investigate the capability and applicability of the RC network in handling image data and consecutive video frames. Given RC's intrinsic temporal properties, instead of using RC as a classifier, we propose to make use of the RC network as a feature extractor and seek to take advantage of its rich collection of dynamical input-output mapping.

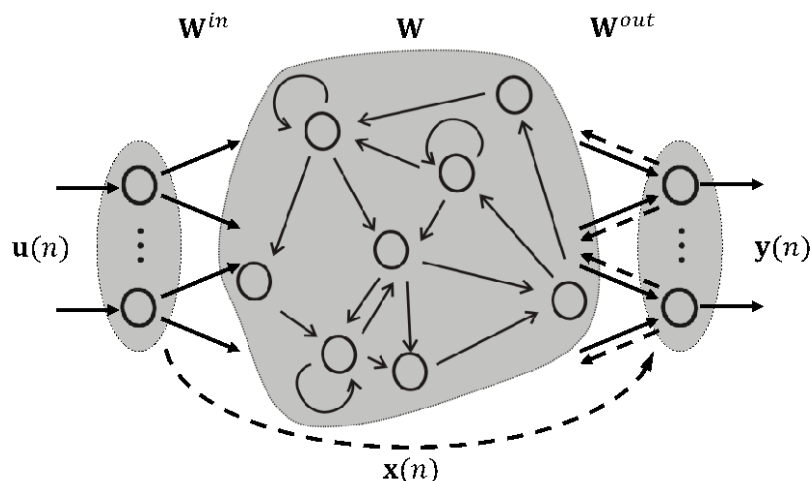


Figure 5 The Structure of Conventional RC Network.

As shown in Figure 5, for each observation, the input data $\mathbf{u}(n)$ is fed into the network through the input connections and drive the reservoir to update its internal activations $\mathbf{x}(n)$ (also called internal states):

$$\mathbf{x}(n+1) = f(\mathbf{W}_{in}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}_{back}\mathbf{y}(n)) \quad (1)$$

f is the activation function of the internal neurons, which is typically the sigmoid function or the linear function. Due to the recurrent connections inside the reservoir, internal states at step $n+1$ are related to the internal states of the previous step. The output of RC network $\mathbf{y}(n)$ is then calculated as:

$$\mathbf{y}(n+1) = f^{out}(\mathbf{W}_{out}(\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n))) \quad (2)$$

where f^{out} represents the activation function of the output neurons. Among those four weight matrices \mathbf{W}_{in} , \mathbf{W} , \mathbf{W}_{back} and \mathbf{W}_{out} , which directly determine the topology and connections of the RC structure, the first three are generally derived using normal or uniform distributions. By forcing the output values towards the expected teacher data, \mathbf{W}_{out} is commonly trained and updated either

online or offline using one of the linear regression algorithms according to the specific application tasks and requirements. In this way, a linear combination can be learnt to model the implicit relationships and approximately represent the structures of input data.

3.2.3.2 Classifier of RC-based tracker.

Softmax regression is a widely-used classifier. It can be viewed as the generalization of logistic regression with which we can handle multiple classes as opposed to only the binary classification. In other words, label \mathbf{y} can take K different values, $\mathbf{y}(i) \in \{1, 2, \dots, K\}$. Since the purpose is to estimate the probability of each possible label becoming the classification result when given testing data, the Softmax function is defined as:

$$h_{\theta}(x) = P(y = j|x) = \frac{\exp(\theta_j^T x)}{\sum_{k=1}^K \exp(\theta_k^T x)} \quad (3)$$

where j stands for the one out of K possible classes, θ represents a vector of weights and x is the vector of inputs. For the convenience of computation, the weight vectors are always arranged row-wise into a matrix.

3.2.3.3 Network configuration.

The overall network architecture is shown in Figure 6.

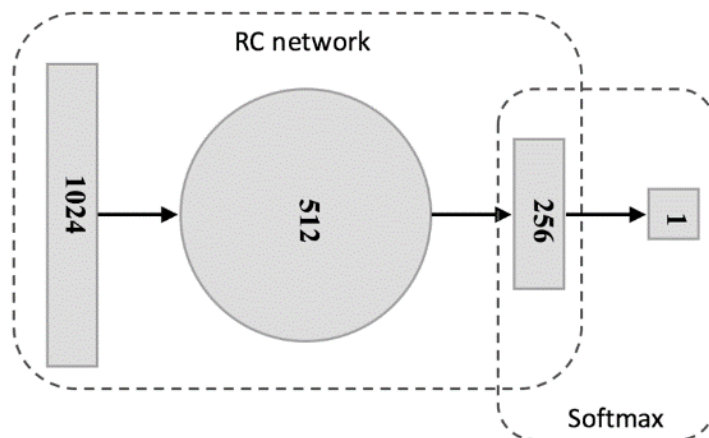


Figure 6 Network for Online Tracking

For a reservoir network, input weights and internal weights are randomly generated with uniform distribution. Sigmoidal activation functions have been used. The size of each layer is designed as foregoing structure in Figure 6 to make it comparable to the SDAE architecture and be capable of progressively extracting image features. Connectivity is set to 0.1 and spectral radius takes the mid-value as 0.5. The configurations of a CNN and a SDAE are inherited from our previous experiments. As a summary, Tables 2, 3 and 4 respectively give the configurations of the proposed RC-based tracker, previously implemented SDAE-based tracker and CNN-based tracker.

3.2.3.4 Training and tracking.

So far, we have introduced the two main parts of the object tracker, the feature extractor and the classifier, as well as the network configuration. Now we will explain the training method and how the previous two parts can work together.

Table 2 Configuration of RC-Based Tracker

Layers	Description	# of Neurons
1 st layer	image input	1024
2 nd layer	reservoir	512
3 rd layer	reservoir output	215
4 th layer	softmax layer	1

Table 3 Configuration of SDAE-Based Tracker

Layers	Description	# of Neurons
1 st layer	image input	1024
2 nd layer	hidden layer	2560
3 rd layer	hidden layer	1024
4 th layer	hidden layer	512
5 th layer	hidden layer	215
6 th layer	softmax layer	1

Table 4 Configuration of CNN-Based Tracker

Layers	Description	Feature Map Size	Kernel Size/Pooling Ratio
1 st layer	image input	32×32	/
2 nd layer	hidden layer	$28 \times 28 \times 12$	kernel size: 5×5
3 rd layer	hidden layer	$14 \times 14 \times 12$	polling ratio: 2
4 th layer	hidden layer	$10 \times 10 \times 24$	kernel size: 5×5
5 th layer	hidden layer	$5 \times 5 \times 24$	polling ratio: 2
6 th layer	fully connected layer	/	/
7 th layer	softmax layer	/	/

As shown in Figure 6, the front portion of the architecture is a standard RC network, followed with a Softmax layer as the classifier. The output layer of the RC network is trained to extract features of the object being tracked. However, unlike the traditional way of training, the features yet to be learnt cannot be precisely described as the teacher data. Consequently, linear regression is not feasible in this case and back propagation is used instead.

In this application, we used a testing dataset of 80 video frames, the same as our previous work. For simplifying the implementation and verifying feasibility of RC in such task, only one trailer out of three is to be tracked. The position of the object to be tracked is specified manually in the first frame of video with a bounding box, as shown in Figure 7 (the numbering on the image starts from 0). Some negative examples from the background are also selected to train the tracker.

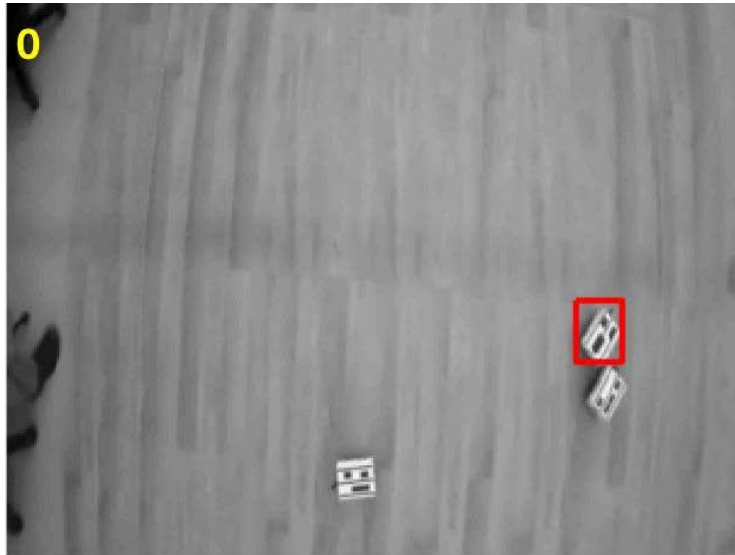


Figure 7 First Frame in Video Clip Used for Initialization

For each frame of the video clip, the input of the tracker is a 32×32 gray image, indicating one possible candidate of object. In this case, each image is represented by a vector of size of 1024 (i.e., number of pixels). The gray-scale values are directly fed into the input layer with each input neuron corresponding to one pixel.

Given the fact that the position of the object in one frame is strongly relevant to its position in the last frame, particle filter can be very helpful in sampling latent candidates from the potential areas. Therefore, when a new frame comes, a set of particles are drawn first and each particle is then passed forward through the whole network to evaluate the confidence. Once all particle confidence levels are calculated, the position of the object can be identified as the particle with the largest confidence.

In case of potential image distortions during tracking, a threshold of confidence τ is set. Every time the largest confidence falls below the threshold, it may indicate some significant object distortions, and the network will be trained and tuned again to adapt to the changes.

Although some prior research has adopted RC in computer vision tasks such as image segmentation, action recognition and facial expression recognition, it is still reported by many, that directly handling image data with a RC network is not easy. In order to simplify the implementation and investigate the basic image processing ability of reservoir computing, we skipped the pre-training of the network which consumes a long time to learn meaningful features

from a large number of general images, and developed a RC-based tracker using only online training for single object tracking.

10 positive and 100 negative samples are selected from the first frame to initialize the latter part of the network, which includes the output layer of the RC network and the Softmax layer. The training repeats for 20 epochs before the tracking starts and the trained network can be used to evaluate the confidence and distinguish between object and background in the following frames.

3.2.4 Tracking with the Combination of RC and AE.

Compared with the stacked denoising autoencoder (SDAE) and the convolutional neural network (CNN) approaches, RCT gave the worst performance on both tracking accuracy and computational efficiency, leading to the conclusion that RC is less effective and favorable on direct and accurate image representation (details see section 4.1). Through analyzing the unsatisfactory performance of RCT, we consider that the echo state property of RC model which significantly benefits the processing of time series data, has become a limitation in image representation, especially in object tracking when directly combining with particle filter. In time series modeling, the echo state property is usually leveraged to handle the temporal relations. However, in object tracking with the adoption of the particle filter, the input data sequence is a cluster of possible candidates established over spatial relations rather than temporal relations. For this reason, the representation of current candidate may be affected by the “memories” of the previous input samples through the feedback pathways. This may result in inaccuracy in image representation and the subsequent classification. In addition, the complex non-linear combinations and recurrent feedbacks inherent in the chaotic structure of the reservoir could cause significant delay when updating the internal activations. “Memories” echoed among the internal connections shall be spread and expressed sequentially, resulting in the expensive time cost. This is especially the case when the dimensions of input and reservoir are extremely huge, as in the application of image processing.

In view of the above reasons, we have explored a new way to leverage RC’s advantage in handling time series data and proposed a paradigm in which RC is adopted to predict the movement of the target, based on the estimated position, where SDAE and softmax are used to recognize the object. In this way, we expect that role of the particle filter can be weakened and the computational efficiency could be improved.

3.2.4.1 Trajectory predictor.

In order to make use of RC’s ability in time series modeling and avoid the weakness of accurate image representation, we combine RC and AE to develop a new tracking paradigm (RCAET for short), in which a standard RC network is adopted as the trajectory predictor. Instead of directly feeding the images into RC, the target location in the previous frame is used as the only input and the output is trained to predict the potential position in the current frame.

From the learning perspective, this approach still faces the same challenge as the other methods in visual tracking, for very limited training samples (usually only one instance in the first image frame) are available ahead of time. In the subsequent frames, the tracker has to learn to predict the trajectory and, based on the estimated location, to identify variations of the tracked object with only unlabeled data available. With nearly no prior knowledge about the object’s history trajectory, it is easy for the tracker to lose track of the target. Thus, it is necessary for the tracker to learn prior knowledge about the object. To address this issue, a virtual trajectory with 1000 data points was

generated in advance. It is designed to be ended with the object position in the first frame and to comply with two restrictions for each step: the maximum step length of 40 pixels and the maximum turning angle of 30 degrees.

A schematic diagram of the whole trajectory which contains both virtual and actual trajectories is provided as Figure 8. It is seen that the actual trajectory marked in the red box is connected with the virtual trajectory. Two input neurons and two output neurons are implemented, representing the X and Y coordinates of the target location.

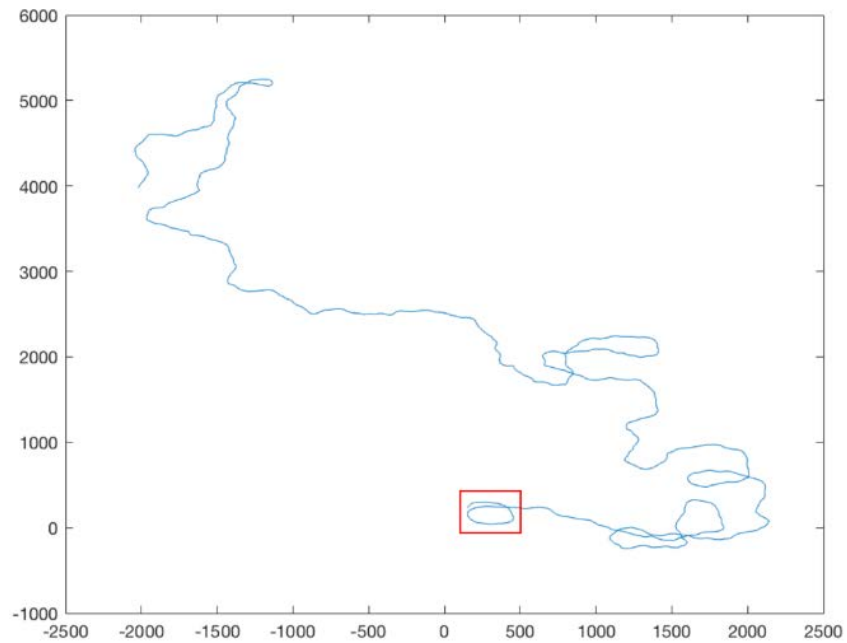


Figure 8 The Virtual and Actual Trajectories.

To verify the feasibility of RC on trajectory prediction, we utilized the whole trajectory in a compliance test. Virtual trajectory for training and actual trajectory for testing. The prediction result using ground truth as input is shown in Figure 9. The blue line draws the ground truth and the red line describes the predicted values.

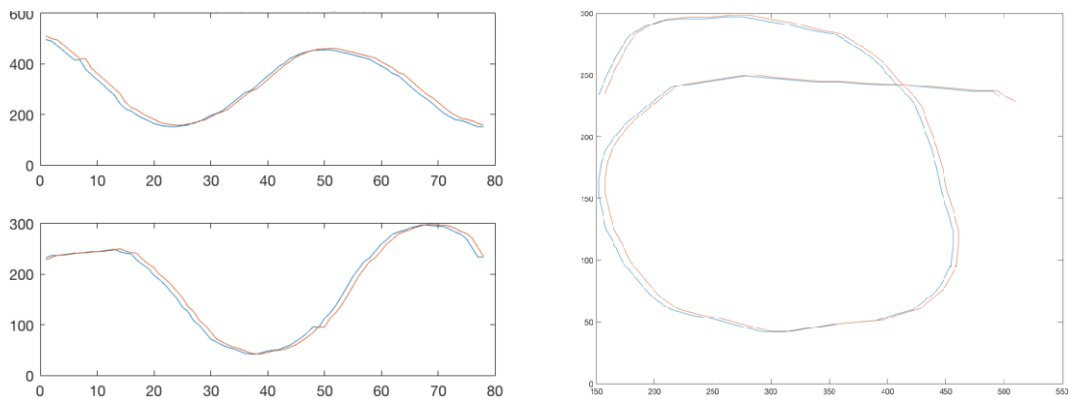


Figure 9 Prediction Result of RC Predictor (left: x, y coordinates; right: trajectory)

Although it has shown very good performance on trajectory prediction, the results are obtained using ground truth data, which can never be available in real cases. Therefore, only the virtual trajectory is used to train the RC predictor and to initialize the internal states before tracking. Starting from the second frame, the determined position identified by particle filter and classifier will be used as input instead of true position.

3.2.4.2 Feature extractor and classifier.

We've concluded that RC is not the most effective option as a feature extractor and it has been observed that SDAE based tracker exhibits the best performance among all three paradigms no matter in accuracy or time efficiency. We hereby choose SDAE as feature extractor and softmax as classifier.

3.2.4.3 Network configuration.

The network configurations of SDAE as well as the network for online tracking are presented in Table 5.

Table 5 Network Configurations of SDAE and Online Tracking Network

Layers	SDAE		Network for online tracking	
	Description	# of Neurons	Description	# of Neurons
1 st Layer	image input	1024	image input	1024
2 nd Layer	hidden layer	2560	hidden layer	2560
3 rd Layer	hidden layer	1024	hidden layer	1024
4 th Layer	hidden layer	512	hidden layer	512
5 th Layer	hidden layer	256	softmax layer	1
6 th Layer	hidden layer	512	/	/
7 th Layer	hidden layer	1024	/	/
8 th Layer	hidden layer	2560	/	/
9 th Layer	image output	1024	/	/

3.2.4.4 Training and tracking.

For the two reference networks SDAE and CNN, neither of them have the capability of dealing with temporal relations over successive video frames. To overcome this fatal shortage for object tracking, a particle filter is adopted so that the intrinsic temporal relations can be transformed and represented as spatial relations. However, along with the employment of a particle filter, a large

amount of computation was introduced into the tracking process. By incorporating temporal information with an RC trajectory predictor, we seek to weaken the role of the particle filter in this tracking task and hence improve the efficiency of RC tracker.

To achieve this goal, the tracking is executed in the following way. First, the RC network is trained and initialized using the randomly generated virtual trajectory. After that, the position of the object to be tracked is specified manually in the first frame. Some positive and negative examples are selected to train the SDAE in a supervised manner. For the first 10 frames, the object is tracked in the old way, with 1000 latent candidates being selected by the particle filter. The position of the object can be identified through the calculation of encode network and softmax. In the meantime, the RC network takes the position of the identified object as input and makes the prediction. Although at this time, the predicted positions don't participate in tracking, some highly accurate input data can be acquired for the RC predictor. This is very important for the accurate prediction in the following frames. Starting from the 11th frame, the particle filter will no longer sample 1000 candidates based on the object position in the last frame, instead, it will search in a smaller area around the predicted object position of current frame (i.e., the output of the RC network), generating a candidate space of 500.

In case of potential image distortions during tracking, an update threshold τ_u is set. Every time the largest confidence falls below the threshold, it may indicate some significant object distortions, and the network will be trained and tuned again to adapt to the changes. It is worthy to note that, the implementation of the RC predictor may bring in some additional errors in the prediction stage. In order to achieve robust tracking, such error must be managed with an extra threshold (a give-up threshold τ_g) to avoid failure from the very beginning. The give-up threshold is set to be lower than the update threshold. Once the largest confidence become even smaller than τ_g , the predicted position derived from the RC network will be given up, the system will redo the tracking of the current frame and the conventional configuration of particle filter with 1000 candidates will be reused based on the previous position. Through utilizing both thresholds, the robustness can be better guaranteed.

3.3 Task 3: Dynamic Ensembles of Reservoir Networks for Multi-Object Pattern Recognition

Reservoir computing (RC), mainly consisting of echo state network (ESN) [54] and liquid state machine (LSM) [55], was proposed as an extension of recurrent neural networks (RNNs). The underlying features of the sparse network illustrate the original ideas of RC, which include: 1) a single "reservoir" within which the connections and weights are randomly generated; 2) the fixed topology of the reservoir after its formation; as well as (3) the only trainable linear combination of all connections in the readout layer.

Thanks to the aforementioned characteristics, RC is relatively easy to implement and has been successfully applied in a broad range of applications, from image processing, pattern recognition, speech recognition to chaotic systems prediction and control [5, 8, 38, 53-59]. The echo state property makes it good at handling time series data. In addition, a bunch of tunable parameters, such as reservoir size, spectral radius, degree of sparsity, the scaling of inputs/feedbacks, and sometimes even the connection weights provide rich sets of dynamics to achieve desired performance. However, requirements of experiences on tuning the parameters and understanding about various dynamic behaviors resulted from different topologies as well as internal connections have driven the research efforts in optimizing the generation of reservoirs to adapt to diverse practical problems.

Keeping those characteristics in mind, it is not surprising to see that majority of the prior research work focused on the application and optimization of RC with the presence of a single reservoir. One considerable issue that has long been ignored is the potential and significance of the network of multiple reservoirs. As reported by many in the literature, the conventional ESN which only contains one reservoir could be powerless when facing some complex multiple superimposed oscillator (MSO) problems and even incapable of modeling the function with two sine waves of different frequencies [54, 60, 61]. That may possibly be because of the inherent coupling among neurons within one reservoir, introducing difficulties in modeling multiple somewhat uncorrelated dynamics [62]. This problem leads to the idea to construct a reservoir ensemble with multiple reservoirs wherein each reservoir is responsible for a specific set of dynamics. The concept of a reservoir ensemble is also inspired by its biological counterpart, the structure of the brain cortex, where different regions (roughly divided into frontal lobe, parietal lobe, occipital lobe, and temporal lobe) are responsible for diverse functions and actions, such as emotion, movement, visual/ auditory/ olfactory processing, memory and speech [63]. Those defined functional regions connected with nerve fibers underneath, form a sophisticated dynamic system and help us to adapt to complex situations as well as to make comprehensive decisions based on all kinds of information. In a word, the reservoir ensemble holds great potential in addressing more complicated applications which involves intricate dynamics and complications.

Reservoir ensemble is not a brand-new concept. Some prior research efforts have been conducted, providing solutions to MSO and multi-object problems with the combination of certain type of reservoir ensembles and other novel techniques.

Yanbo Xue et al. [60] constructed decoupled echo state networks (DESN) with multiple randomly generated reservoirs (termed sub-reservoir) of different configurations. Connections between sub-reservoirs are implemented using lateral inhibition in two ways to overcome the coupling effects within a single reservoir and to reduce the correlation of dynamics between sub-reservoirs at the same time. Internal states of all the sub-reservoirs are integrated at last to output the action potentials. In this way, multiple tasks could be cooperatively accomplished. The model was tested with an MSO problem of two sine waves and was applied in prediction of sea clutter data, good performance was observed.

Jun Yin and Yan Meng [5] proposed a GRN-regulated self-organizing reservoir computing (GRN-SO-RC) model targeting at multi-object behavior recognizing in computer vision. In their model, several sub-reservoirs are constructed in a cortical neural network and connected through inhibitory connections. A gene regulatory network (GRN) was used to regulate the parameters of the RC model and hence each reservoir can be capable to detect one behavior. The output layer is also constructed by combining the internal states of sub-reservoirs.

Fabian Triefenbach et al. [58, 64] introduced deep hierarchical architecture into RC by connecting several “layers” of reservoirs end-to-end. In this case, the readout layer of the former reservoir is treated as the input layer of the latter reservoir. Benefits from this, such as different levels of acoustic units, like phonetic states, phones, syllables and so forth, can be extracted sequentially in the application of acoustic modeling.

Guihua Wen et al. [57] proposed an approach based on RC for facial expression recognition. Multiple independent reservoirs are adopted as basic classifiers. A convolutional neural network is set ahead of each reservoir to extract features that will be fed into reservoirs as inputs. At the readout layer, outputs from multiple reservoirs are fused with a voting strategy.

All the above work has shown some possible structures for reservoir ensembles and have achieved good results in their fields. However, only one structure (either the structure with one

input layer, multiple reservoirs and multiple output layers or the structure with one input layer, multiple reservoirs and one output layer) was selected and analyzed in each research effort for the purpose of solving specific problems. In order to gain a more comprehensive understanding on the potential and significance of reservoir ensemble architecture, more systematic analysis must be addressed over all possible configurations. In the next section, a brief introduction and explanation will be provided on significant structures.

3.3.1 RC with Multiple Reservoirs

It has been well understood that the structure of RC is composed of four main parts: input layer, reservoir, readout layer and all connections therein. Among the four parts, the first three determine the shape of the structure while the connections represent the way in which reservoirs are correlated. Therefore, it is necessary to first list all possible structures defined by the numbers of input layers, reservoirs and readout layers before we take a further look.

Table 6 shows eight different configurations of RC structure. Note here that the number of input/output layers does not mean the number of neurons but individual layers, multiple input/output neurons could exist within one layer.

Table 6 All possible configurations of RC structures

# of input layers	# of reservoirs	# of output layers	Notation
1	1	1	1I-1R-1O
1	1	Multiple	1I-1R-MO
1	Multiple	1	1I-MR-1O
1	Multiple	Multiple	1I-MR-MO
Multiple	1	1	MI-1R-1O
Multiple	1	Multiple	MI-1R-MO
Multiple	Multiple	1	MI-MR-1O
Multiple	Multiple	Multiple	MI-MR-MO

Obviously, the first configuration in Table 6 is the conventional RC structure, but not all of the listed structures are meaningful and feasible. In what follows, we will discuss the various structures as well as their feasibility and practical significance.

3.3.1.1 Redundant Structures

The structures below are redundant, not only because they are barely workable, but also that they are somewhat meaningless or simply equivalent to the conventional structure even if they appear to have different configurations (be equipped with multiple input/output layers or multiple reservoirs).

1I-1R-MO, MI-1R-1O & MI-1R-MO. It is not hard to point out that most structures with a single reservoir are functionally equivalent with the conventional model, even though multiple input or output layers are equipped. Figure 10 shows two structures with only one reservoir. In both cases (and as well the case with multiple input layers and readout layers at the same time), if we assume that the input and output layers consist of same type of neurons and no difference is introduced into the processing and training procedures among the multiple layers, then they can always be seen as an equivalent structure. This is because each input/output neuron (no matter in

which layer) is created equally, connecting with all internal neurons of the reservoir. This always makes it equivalent with the conventional model.

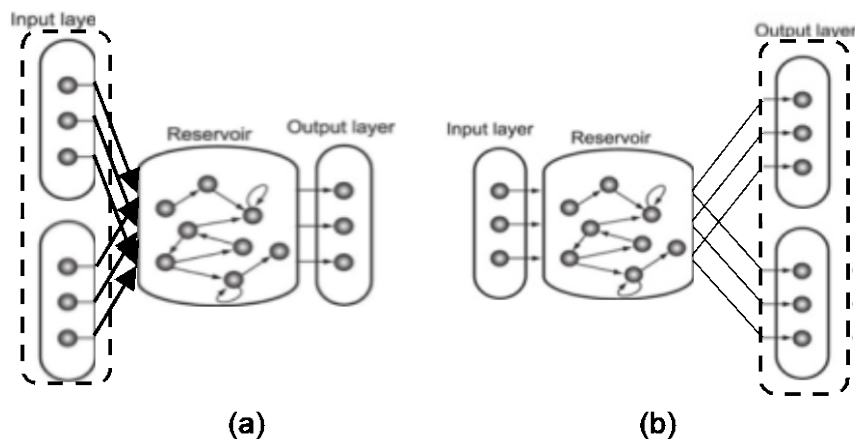


Figure 10 (a) Structure with Multiple Input Layers, Single Reservoir and Single Output Layer; (b) Structure with Single Input Layer, Single Reservoir and Multiple Output Layers.

MI-MR-MO. This model can be seen as a general form of all structures. It requires a mapping from different input layers to multiple reservoirs as well as a mapping from multiple reservoirs to diverse readout layers. Necessary connections between reservoirs are also required, or it may easily break up to several independent networks of other types (conventional structure, 1I-MR-1O, 1I-MR-MO or MI-MR-1O). Since the numbers of reservoirs and input/output layers may not necessarily be equal, it will end up with a large number of complex instances. Moreover, it hardly fits into any application scenarios, nor holds any explicit significance. Therefore, we are not interested in this construction.

3.3.1.2 Significant Structures

1I-MR-1O. Two cases under this configuration should be considered: 1) the structure where reservoirs are independent with each other; 2) the structure where reservoirs are connected with each other at a certain level (connectivity).

For the first case, it's not very significant. Since all the connections within a reservoir are randomly generated, sparse, it can be simply seen as multiple collections of internal neurons that happen to hold no connections (0 weights) with each other. From this perspective, it is also equivalent with the conventional model.

For the second case, evidence has been provided by Jun Yin and Yan Meng [5] and Yanbo Xue et al. [60] that it not only exhibits significance, but also achieves good results in real-life applications. However, not all such structures are practically significant. The constraint condition is that the connections between reservoirs shouldn't be generated randomly, or it would have no difference compared with the conventional model.

1N-MR-MO. This structure has also been investigated in the literature [57, 58, 64] in two types. The first type constructs a hierarchical architecture in a cascade manner (Figure 11). It is similar to deep neural networks in terms that it hierarchically extracts features in different levels. Although only the last readout layer is used as the final result, it reserves all its readout layers in the middle of the chain. So, we still classify it as 1N-MR-MO. The second type is built in a flat

shape (Figure 12). This specific type only makes sense when different reservoirs are provided with different configurations and different tasks or when the connections between reservoirs are something other than random generations (similar as 1I-MR-1O).

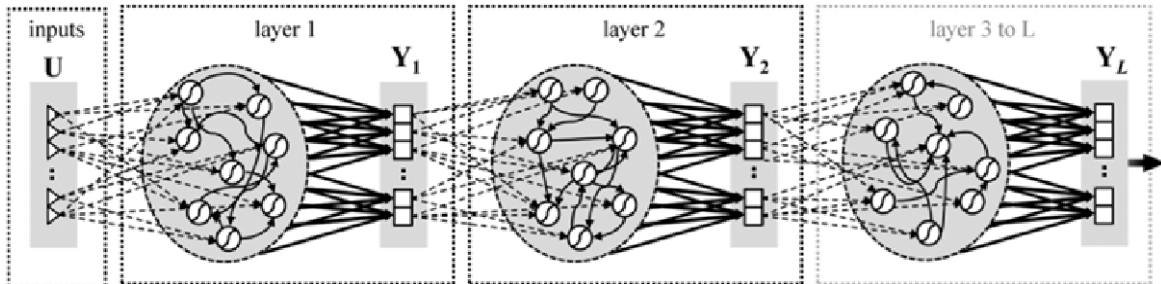


Figure 11 Hierarchical Structure with 1N-MR-MO

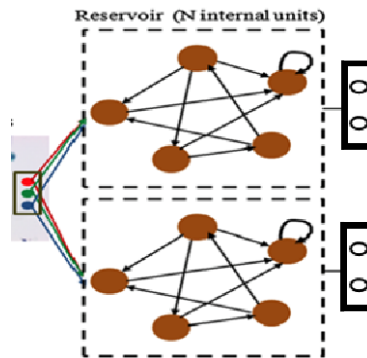


Figure 12 Flat Structure with 1N-MR-MO

MN-MR-1O. This is the structure that we are most interested in, generally because of the motivation from real-life decision making. One objective can be reflected and determined from different aspects, and in turns, better decisions can be made with multiple sources of information. One simple example would be the fusion of different sensory information in our brain. When determining the quality of some food, one may need the information of color, smell and flavor together to make the decision. The schematic diagram of this structure is shown in Figure 13.

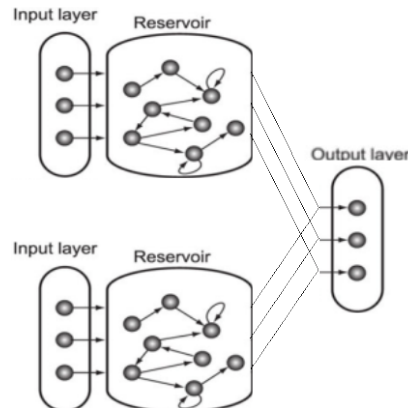


Figure 13 MN-MR-1O Structure

3.3.2 Dynamic Reservoir Ensemble Model Based on Genetic Algorithm

By analyzing the complex structures discussed above, it is not difficult to see that further exploration of reservoir ensemble is confronted with two critical problems. The first problem is the large searching space, caused by the vast number of connection combinations, makes it difficult to manually design a capable structure for a given task, and a considerable amount of domain knowledge may also be required. The second problem is that once a certain structure is picked, the connections are fixed and there is no way to adapt to other tasks. Considering these limitations, we seek to find an approach to save energy from model designing and to adapt the reservoir ensemble to complex tasks through dynamic optimization of the structures. We call this approach the Dynamic Reservoir Ensemble (DRE) model and conversely, we term the models in previous section as Static Reservoir Ensemble (SRE) models.

DRE employs the genetic algorithm (GA) as an optimizer. The newly proposed model is expected to dynamically evolve and adapt itself based on the given task so that complex dynamics can be modeled and designers can be released from the burdensome work of designing appropriate structure as well as exploiting a vast amount of domain-knowledge.

3.3.2.1 Genetic Algorithm

A genetic algorithm (GA) is a commonly used evolutionary algorithm inspired by the process of biological evolution and natural selection [65]. A GA tries to optimize the parameters gradually by inheriting the properties from relatively good parents in the last generation and introducing mutation to escape local optima.

All the properties of a candidate solution (also called individual) are encoded into a vector or string (chromosome or genotype) which is allowed to be edited. Each generation consists of a population of individuals. Usually, the first generation is initialized randomly, which means the chromosomes are generated following the uniform distribution. For each generation, the performance (fitness) of each individual is evaluated using an objective function (fitness function). According to the fitness value, selection is taken throughout the current generation and parents (the group of individuals that have better fitness) will be chosen to contribute their genes. Subsequently, three types of children will be created to make up the next generation. The individuals with best fitness automatically survive to the next generation and become elite children. Crossover children are created by randomly exchanging portions of chromosomes from two or more relatively good parents. Mutation occurs by infrequently, randomly changing the gene of a single parent, so that diversity can be explored and guaranteed and thereby falling into local optima can be largely avoided. Evolution operates following this manner iteratively, pushing the optimization problem towards better solutions until one or more stopping conditions are met, which may include the maximum number of generations, operation time limits, target fitness value, generation number of no improvement, etc.

There are several reasons for the choice of a GA instead of another optimization or evolutionary algorithm. Firstly, we proposed a general model that evolves the near-optimum structure for different applications, so that no explicit formulation or analysis of the target system is needed. In addition, there is no clear functional relationship between network structures and fitness values. In other words, the reservoir ensemble network serves like a black-box simulation model and the objective function is discrete and stochastic. Thus, derivation methods cannot be applied. Moreover, the existence or absence of connections can be easily encoded into binary strings, making GAs more capable of parametric representation than other evolutionary algorithms

such as particle swarm and ant colony optimization, etc. Meanwhile, GAs always come up with an answer, which usually becomes better and better with time. For the aforementioned reasons, we consider the GA an applicable method in our case.

3.3.2.2 Dynamic Reservoir Ensemble Model

In this model, the MI-MR-MO structure is adopted to meet with the most general case. For a specific task, it can be assumed that the types of input data and desired outputs are defined. In this case, with one neuron corresponding to a certain type of input/output data, the number of input neurons and output neurons are fixed. In our model, the input/output neurons are treated as independent, in other words, those neurons are not bounded in one layer anymore and are provided with the freedom of connecting to any or multiple reservoirs. The structure of the reservoir is determined by connections from inputs to reservoirs, connections from reservoirs to outputs and the connections between different reservoirs. To narrow down the size of the search space (i.e., reduce the number of parameters) and to evolve a good solution more quickly, the number of reservoirs as well as the connections between different reservoirs are predefined and fixed. In this way, only the mappings from inputs to reservoirs and from reservoirs to outputs matter. For simplicity, all connection weights will still be generated randomly in accordance with the standard way.

The properties of each candidate structure are encoded into a bit string, in which every single bit represents the existence (encoded as “1”) and nonexistence (encoded as “0”) of a connection. Figure 14 (a) shows the chromosome of a reservoir ensemble with K input neurons, M reservoirs and L output neurons. The chromosome consisting of $(K + L) \times M$ bits can be divided into M segments, corresponding to the M reservoirs. Each segment contains $K + L$ bits, where the first K bits represent the connections from the input neurons to the current reservoir and the last L bits represent the connections from the reservoir to the L output neurons. The bit string hence covers all possible structures in the searching space, and given one specific string, the only structure can be reconstructed accordingly (again, let alone the weights). The structure of reservoir ensemble corresponding to the chromosome given in Figure 14 (a) is shown in Figure 14 (b) (feedbacks and input-to-output connections are not displayed in this figure).

In the proposed model, output error is evaluated to indicate the performance of the structure. Therefore, the fitness function is defined as the normalized root mean square error (NRMSE) of the outputs on testing data.

The initial generation is created by randomly producing a population of bit strings following the uniform distribution. Corresponding reservoir ensemble structures are then generated and trained. According to the NRMSE values of the testing results, a selection is made and elite structures are passed directly to the next generation. New structures can be created through crossover and mutation, heredity and changes consequently take place in different parts of the input/output connection.

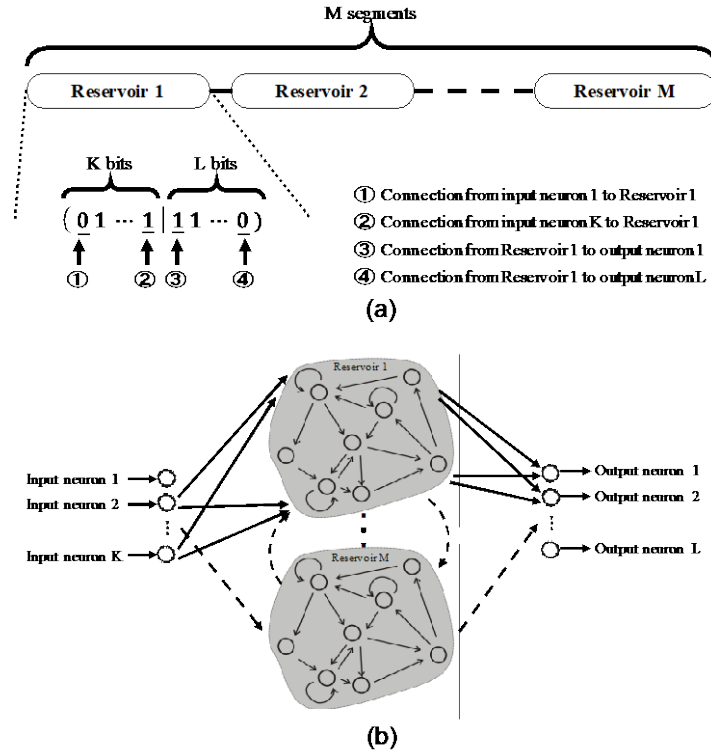


Figure 14 (a) Chromosome of Reservoir Ensemble Model; (b) Corresponding Reservoir Ensemble Structure

3.3.2.3 Time Series Prediction Using Dynamic Reservoir Ensemble

We test and evaluate the performance of the proposed dynamic reservoir ensemble model on a real-life time-series problem, surface temperature prediction. We choose the dataset of historical meteorological observations in the Binghamton Tri-Cities Airport from 1948 to 1952 (downloaded from the website of the United States National Climatic Data Center [66]). From the observations, five variables were picked as the inputs: daily mean dew point (DEWP), daily mean station pressure (STP), daily mean wind speed (WDSP), daily highest temperature (High Temp.) and daily lowest temperature (Low Temp.). The highest and lowest temperatures are to be predicted. 1000 days of data are adopted for training and the successive 100 days for testing. The standard model, static reservoir ensemble (5I-5R-1O structure as an example) and dynamic reservoir ensemble are applied respectively and results of different structures are compared and analyzed.

For the standard model, two types of structures were tested. The first one employed only one input neuron and one output neuron. Under this structure, two sets of experiments were implemented, the history data of either High Temp or Low Temp was adopted as input signal and the reservoir was trained to predict the future value of High Temp or Low Temp. The second type adopts five input neurons and two output neurons. All five input signals were used simultaneously to drive the single reservoir. In this way, both high temperature and low temperature were modeled using one set of dynamics.

For the static reservoir ensemble model, five input neurons, five reservoirs and two output neurons were adopted. Different from the standard model, each input neuron was connected only to one reservoir, while output neurons were connected to all five reservoirs. We tested two variants under this configuration, one exhibited correlations among sub-reservoirs, the other held no correlation.

For the dynamic reservoir ensemble model, we used five reservoirs to model High Temp and Low Temp using the five input signals. However, considering that the complex mechanism behind the climate system and the interactions between variables remained unclear to us, some unavailable factors may have affected temperature values. We kept the connections between reservoirs and input/output layers unspecified and allowed the dynamic reservoir ensemble model to explore an optimal structure that could handle the potential unknown relations by itself.

To implement the reservoir networks, the MATLAB toolbox developed by Jaeger [67] was used to generate the standard model. Tanh function was adopted as the activation function for both internal neurons and readout neurons. Without loss of fairness and generality, the net dimension (reservoir size) was set to be 200 neurons for the standard structure and 40 neurons in each reservoir of the static and dynamic reservoir ensemble models, so that the total numbers of computational units in all three configurations were the same. All the weights were randomly generated. The connectivity (sparsity) within each reservoir was set to be 0.1 and connectivity between different reservoirs (if applicable) is 0.01. For each of the input signals, 15 samples were used for each step of the input. In the genetic algorithm, the population size was 100, 5 elite children were to be generated in each generation, uniform mutation occurred at the rate of 0.2, and 80% of the population were recombined to generate crossover children. A 5-2-bit crossover function was customized to break down the chromosome for each 5 and 2 bits, corresponding to 5 input-to-reservoir and 2 reservoir-to-output legal connections for each reservoir. The purpose was to preserve the full information of input and output connections of a reservoir for inheritance. Since we relied on the two outputs equally, the fitness value was the average of two NRMSEs. The results are described in section 4.4. Evolutionary Adaptation for Reservoir Network Optimization

3.4 Task 4: Human Category Learning Inspired Classification Network

Although not extensively investigated, category learning has been addressed by some early studies [68-72]. Among which, some notable computational models on category learning were established based on the structure of an auto-encoder [71, 72]. In this work, a neural model called divergent auto-encoder (DIVA) was trained in the supervised manner to gain shared information among the connections of the hidden layer and to differentiate different categories over the divided outputs. Inspired from the basic idea, we felt that it would be interesting to explore this issue based on a more biologically plausible combination, RC network internally trained with synaptic learning.

3.4.1 Network Architecture.

We constructed the network as Figure 15. Two divided output layers were employed, corresponding to two different categories. Considering that if there are only forwarding connections in the readout layers, the divergent outputs would have no difference with a (single) joint longer output layer and hence become meaningless, we reserved the output feedback connections. Thus, the output values from both categories will be able to affect the inner states of the reservoir and therewith exhibit influences on the other output as well as itself. This effect, to some extent, also simulates the feedback behaviors in the cognition process. Different from the output layers, a single input layer is built for the purpose of unified inputs and alternative training over each category.

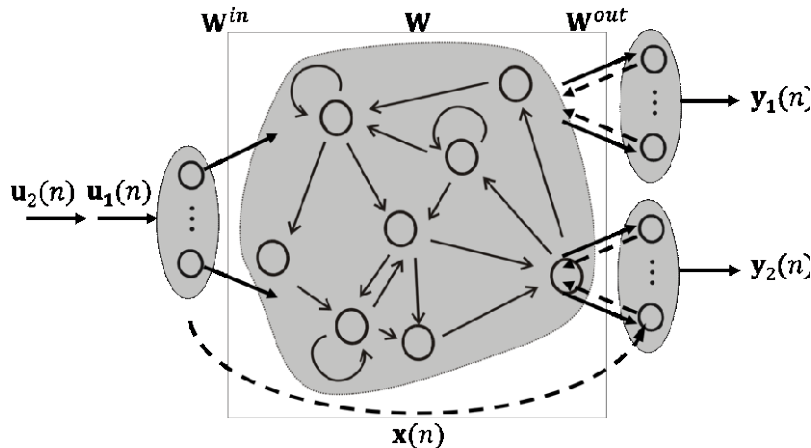


Figure 15 Echo State Network Based Category Learning Model

3.4.2 Training Strategy.

Two stages of training were employed. Unsupervised synaptic learning was applied to the internal weights (inside the reservoir), attempting to learn common features underlying the externally different but intrinsically relevant data (over two categories). The conventional supervised training was implemented as usual on the readout layers so as to learn the boundaries among different categories (two categories in this experiment) and correctly identify them when they appear at the input. Synaptic learning and supervised training on different categories were organized in an alternately interleaving way for several iterations. We did not implement the training phase as a one-time effort because after training the second category, both internal states and the weights in the second readout layer were updated, which would have a significant impact on the first output and disrupt the learned ability on that category (results are shown in section 4.3). For this reason, we trained the model for multiple iterations to obtain stable and balanced internal/output weights over both categories. To achieve this, the training data for each category was split into several segments. For each iteration, a training segment of category one was first fed into the model, then the internal weights were trained with synaptic learning. The supervised learning was only applied on the first readout layer when training category one. Immediately following the training of category one, category two were trained based on a segment of training data in similar way, except that the second readout layer was to be trained instead of the first.

3.5 Task 5: Evolutionary Adaptation for Reservoir Network Optimization

The echo state network (ESN) is a type of recurrent neural network (RNN) that falls under what is known as the “Reservoir Computing” (RC) paradigm. It was first introduced by H. Jaeger in 2001. Its structures and properties have since been thoroughly studied and documented. The appeal of ESNs stems from the fact that it’s reservoir is randomized, and is fixed once initialized. This allows ESNs to have fewer synaptic connections to train, and thus are comparatively computationally inexpensive as opposed to other types of RNN. It has been established that the echo state network has high performance in certain benchmarks, and outperforms regular RNNs

in a number of applications [39, 41]. However, it was also discovered that there are certain limitations or constraints faced by ESNs that require further investigation.

Currently, ESNs face inherent issues present within reservoir computing methods which make it difficult to be applied to practical applications. One of the major drawbacks of ESNs lies with the fact that its properties and performance are contingent on its initialization [39, 42]. However, each task of interest may require a different topology and reservoir size, making the task of choosing initialization parameters a non-trivial endeavor. While there are guidelines that may assist in the selection of these parameters [38], fitting ESNs to a specific task often becomes a game of guess and check. The performance of the networks differs drastically for different starting parameters.

A number of studies have been conducted in an attempt to address these drawbacks and gain performance improvements. Those solutions range from changing the topologies of the initial randomized network to online alterations of the connections and weights within the network, many of which have provided desirable results [39, 43, 73]. However, one of the main issues surrounding the current methods is the inherently complicated nature of the solutions presented. Due to the black box nature of the reservoir structure, there are remarkable difficulties in establishing theoretical models, thus resulting in an incomplete understanding about what aspects of the process are specifically responsible for the performance, whether positive or negative [41].

Another issue is that some original features of reservoir computing (RC) still limit further improvements of such type of networks. For example, although the randomly initialized and fixed weights of the input layer and the reservoir indeed largely reduce the computational complexity and hence the training efforts, this feature also provides a less optimal network for the given task, more or less limiting the performance of the network.

With these issues in mind, this research attempted to search for a bio-inspired approach of reservoir evolution, from the perspective of both structural and synaptic adaption, to deal with dynamic optimization as well as the initialization issues present in ESNs.

3.5.1 Synaptic Adaption Based on Synaptic Plasticity.

3.5.1.1 Principal Neuron Reinforcement.

Inspired by neuroplasticity, an algorithm was proposed to alter the reservoir of ESNs to reach a comparable performance from different starting parameters.

Neural plasticity refers to the process of change which occurs to different aspects of the brain. This change can be triggered by different events which occur throughout a person's life. One of the more specific sub-processes is synaptic plasticity, which refers in particular to the ability of the synapse to strengthen or weaken over time depending on the degree to which they are stimulated [74].

Synaptic plasticity allows synapses in the brain to change and adapt according to the amount of activity in which they are involved. Specifically, the plasticity mechanism is able to modify the strength of the synapses within the reservoir based on the activities stimulated by the input [75]. It is intuitive that the more important a synapse is to the transportation of information, the more it would be used. This reinforcement allows the synapse to be strengthened and therefore be more responsive to future stimulation. This effect can also occur in reverse, where the synapse weakens or degrades due to inactivity.

For the reservoir computing paradigm, specifically echo state networks, while the fixed internal connections of reservoir networks perform adequately for many applications, the

structures' randomized nature results in an amount of unpredictability in the performance. There has thus been motivation to optimize the reservoir structure to achieve better performance in RC networks. In this work, we proposed an alternative synaptic plasticity model, *Principle Neuron Reinforcement (PNR)*, to determine which synapses to modify. The core principles of the PNR is described as follows.

The PNR assumes that in a fixed network structure (as reservoir networks commonly are), there will be neurons that play more important roles in the modeling of target signals. These “principal” neurons contribute critical information to the neurons in the next layer and therefore, if the connections between these principal neurons and the neurons in the next layer are strengthened, more critical information will be passed through. Conversely, there will be neurons that have little impact on target signals, thus their connections should be weakened.

Once an ESN has been trained, it is obvious that the magnitude of the output weights of each neuron in the hidden layer will vary. Here, the larger weights signify strengthened synapses, indicating the connections are frequently used, and that the signal which is passed through these connections are of greater importance in contrast to that passed through weaker connections. From this point, it can be inferred that data passed from the neuron to which the high magnitude weights were connected is important.

On the basis of the findings outlined above, two hypotheses were formed:

Hypothesis 1: *Neurons in the hidden layer with high magnitude output weights contributed more “important information” to the output.*

Hypothesis 2: *The “important information” from these neurons make positive contributions to others in the hidden layer with which these neurons are recurrently connected.*

Built upon these two hypotheses, following the Hebbian learning principle, we proposed the PNR learning rule, a model of how neurons within the reservoir alter the strength of synaptic connections according to the presented input-output patterns.

$$W|_i^j(t+1) = (1 + \alpha)W|_i^j(t), \quad \text{if } W_{out}|_i^k > \xi \quad (4)$$

where α is the learning rate. The synaptic connection from the neuron i to the neuron j within the reservoir, $W|_i^j$, is only updated when the readout weight originating from the neuron i to the neuron k , $W_{out}|_i^k$, is larger than a chosen threshold ξ . The neuron i , in this case, is called the “principal neuron.” The PNR rule regulates that the alternation of synaptic connections can only happen to those ones associated with the principle neurons.

3.5.1.2 Experiment procedure.

In order to test the aforementioned hypotheses and evaluate the performance of the proposed PNR learning rule on ESNs, we performed a set of experiments with varying reservoir size and neuronal connectivity on a benchmark dataset.

The MATLAB toolbox developed by H. Jaeger [67] was used to initialize the echo state networks required for the experiments. The two vector input data is a generated times-series benchmark dataset provided within the toolbox. The equations used to generate the dataset is as follows:

$$\mathbf{y1}(t) = \mathbf{x2}(t-5) \cdot \mathbf{x2}(t-10) + \mathbf{x2}(t-2) \cdot \mathbf{y2}(t-2) \quad (5)$$

$$\mathbf{y2}(t) = \mathbf{x2}(t - 1) \cdot \mathbf{x2}(t - 3) + \mathbf{x2}(t - 2) \cdot \mathbf{y1}(t - 2) \quad (6)$$

where $\mathbf{x1}$, $\mathbf{x2}$, $\mathbf{y1}$, $\mathbf{y2}$ are the input vectors 1 and 2 as well as the output vectors 1 and 2 respectively. For this experiment, $\mathbf{x1}$ was set to be 1 throughout the vector length and $\mathbf{x2}$ was a random vector normally distributed between 0 and 0.5. An input vector of length 2000 was generated, the first 100 of which were used to train the system, and the next 1900 were used in the testing of the trained network.

The ESN was initialized with a connectivity of 0.1 and with configurations of neurons ranging from 100 to 900 in increments of 100. They were then trained using the benchmark dataset. The weights of the internal neurons were bound between -1 and 1. The ESN was initialized with two neurons in the input and output layers respectively.

According to the PNR rule, the recurrent connections to and from the principal neurons contributing the “important information” are strengthened or weakened in a trained ESN. Once the weights have been strengthened and/or weakened, the ESN is retrained and retested. Upon training, the output weights of the ESN were analyzed and ranked from the highest magnitude to the lowest. For the purposes of this experiment, a threshold ξ was chosen by the author for the determination of what is a “high” magnitude. In this experiment, this threshold was chosen to be the top tenth of the weight magnitudes. The threshold was chosen at this point as it is where the weight magnitudes start to rapidly increase.

From here, the recurrent connections associated with these “principal neurons” were strengthened. For this experiment, the increase of the weights was applied incrementally over several iterations at a learning rate of $\alpha = 0.001$. This process was also repeated for weakening the lowest tenth of the weight magnitudes. The performance for strengthened-ESN, weakened-ESN and strengthening+weakening-ESN were compared to the original ESN without neuronal plasticity.

This experiment was also repeated with different settings of connectivity. The connectivity was varied from 0.1 to 1.0 at increments of 0.1 with a setting of 500 neurons. The effects of how connectivity affects error rate and whether the proposed algorithm can counter its effects, was observed.

A similar experiment was applied to a Non-linear Auto Regressive Moving Average (NARMA) dataset to evaluate how well the PNR extended to other types of data. The 10th order NARMA dataset is driven by the equation:

$$\mathbf{y1}(t) = 0.3\mathbf{y}(t - 1) + 0.05\mathbf{y}(t - 1) \sum_{i=1}^{10} \mathbf{y}(i - 1) + 0.5\mathbf{x}(t - 9) \cdot \mathbf{x}(t - 1) + 0.1 \quad (7)$$

We also attempted to evaluate the effectiveness of PRN with respect to current plasticity methods. As established in [75] the Anti-Oja learning rule is effective in improving the rates of error in many time-series prediction applications of the ESN. However, this was under the setting where the initial performance of the ESN was already reasonably high. The Anti-Oja rule was also applied to the training of the aforementioned equations and the results are compared to that of the PNR in section 4.4.

3.5.2 Structural Evolution Based on Genetic Algorithm.

A genetic algorithm was employed to optimize an ensemble of Echo State Networks to achieve the goal of predicting the high and low temperature given historic data. The assumption was that in an application like this, the structure of the ensemble would largely determine the performance of the system. To determine the best structure of the ensemble, i.e., which reservoir should be

connected to the inputs and outputs, GA was used by encoding the input and output connections into genes. It has been seen from experiments that the genetic algorithm employed is able to improve the performance of the system. Details of the methods are described in section 3.3.

3.5.3 Neuronal Plasticity Inspired Reservoir Ensemble Optimization

3.5.3.1 Verification of the Structural Contribution in Reservoir Ensemble

In the submitted paper [76], we proposed a dynamic reservoir ensemble model that consists of multiple reservoirs and is capable of automatically adapting and optimizing the structural plasticity of a reservoir ensemble towards an optimal performance using the genetic algorithm. Implemented in a real-life time series application - temperature prediction, the proposed model demonstrates superior performance over both conventional single-reservoir model and the static reservoir ensemble model with fixed connections.

In order to further verify the contribution of the structure determined by the GA and confirm that the good performance is mainly benefited from the reservoir ensemble, we designed an experiment to remove the least important reservoirs one by one. The performance was then observed.

The determining rule of the least important reservoir is established as follows:

Rule 1: *The reservoir that connects to the least number of output neurons is determined as the least important one;*

Rule 2: *For reservoirs with same number of outputs, the output weights are summarized with respect to each reservoir; reservoirs with larger total values tends to be seen as more important.*

Based on the weather dataset adopted in the IJCNN paper, two trials were played and two relatively good RC ensemble structures were evolved as shown in Figure 16. In both GA determined structures there exists one reservoir that is not connected to both outputs. Those reservoirs are to be removed first. According to the rule, the order of removing reservoirs in two reservoir ensemble models are 2, 4, 5, 3 and 1, 5, 4, 2 respectively. The performance for each removal is collected in Table 8 and Table 9.

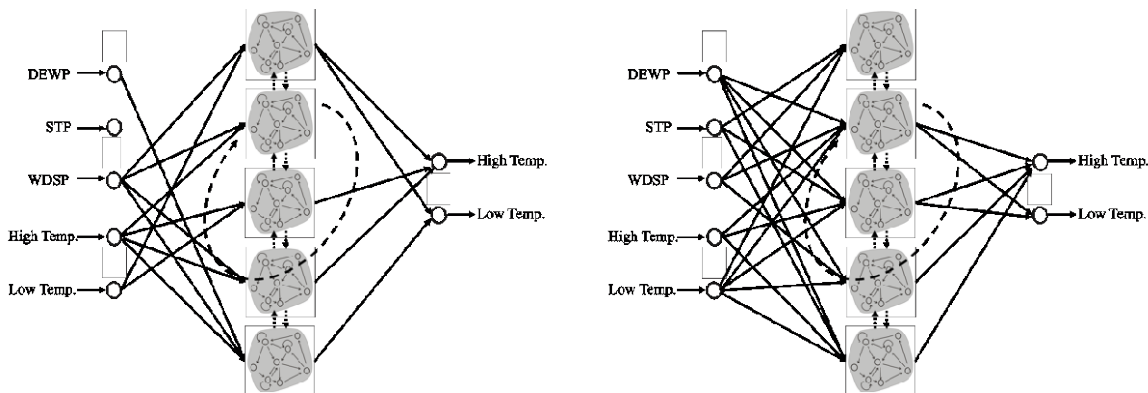


Figure 16 GA Determined Structures 1 (Left) and 2 (Right)

It can be observed from Table 7 and Table 8 that by removing the first reservoir, the normalized root-mean-square error (NRMSE) for both outputs were increased dramatically, which indicates that even holding no output connections, the least important reservoirs still contribute to the output performance through structural connections.

Table 7 MSE of Each Removal for Structure 1 (40 neurons for each reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
Structure 1	0.59932	0.62483
1 Reservoir Removed	0.76741	0.70344
2 Reservoir Removed	0.68458	0.67891
3 Reservoir Removed	0.7113	0.74802
4 Reservoir Removed	0.79227	0.74399

Table 8 MSE of Each Removal for Structure 2 (40 neurons for each reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
Structure 2	0.55844	0.65731
1 Reservoir Removed	0.71061	0.70181
2 Reservoir Removed	0.76111	0.73969
3 Reservoir Removed	0.75541	0.71748
4 Reservoir Removed	0.69503	0.71136

Another interesting observation in both cases is that, as we keep removing less important reservoirs, the performance may improve a little bit for some certain steps, but the NRMSE will never drop below that of the entire ensemble. This observation corresponds to one of the key ideas in the complex systems theory: a reliable system can be formed with unreliable components, conversely, using all optimal components may not necessarily end up with an optimal system.

As we further enlarged the size of each reservoir from 40 to 100, a similar trend was observed, whereas the NRMSE increases for each removal. Result are shown in Table 9.

Table 9 MSE of Each Removal for Structure 3(100 neurons for each reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
Structure 3	0.6689	0.6295
1 Reservoir Removed	0.7157	0.7431
2 Reservoir Removed	0.8227	0.9094
3 Reservoir Removed	0.8738	0.9529

From the above results, it can be further verified and convinced that the optimized structure rather than the principle reservoir(s) is the major contributor to the improved output performance.

3.5.3.2 Enhanced Dynamic Reservoir Ensemble with Principal Neuron Reinforcement

Based on our previous works on dynamic reservoir ensembles (DRE) [76] and principal neuron reinforcement (PNR) [77], we investigated the effects of combining those two techniques

to further optimize reservoir ensembles from both structural and neuronal plasticity perspectives. On one hand, DRE improves the performance by optimizing the structure, resulting in a desirable structure to improve the performance of the system. On the other hand, PNR allows the strength of internal connections to be changed, at the same time removing the difficult task of selecting initialization parameters. Since one method improves the performance by optimizing the structure while the other improves performance by modifying the internal connections, we investigated the potential that both techniques can synergistically produce a comprehensive method, which has the ability to deal with both.

DRE and PNR were combined based on the temperature prediction task. Networks with different reservoir sizes were tested. The optimization results and discussion of their significance will be presented in section 4.4.

3.5.3.3 Enhanced Dynamic Reservoir Ensemble with Classic Synaptic Plasticity

As shown in our previous work, although the dynamic reservoir ensemble model and principal neuron reinforcement can work well independently to enable the adaption of RC and improve its performance, significant improvement was not observed when bringing the two parts together. To deal with such issue, we reexamined the biological process of synaptic plasticity which works in a manner that gradually changes (strengthens or weakens) the synaptic strength over time and sought for some classic synaptic plasticity rules that help.

When applied to artificial neural networks (ANNs), several synaptic plasticity learning rules have been proposed. We identified and analyzed the four most popular plasticity learning approaches:

Hebbian Learning. Hebbian theory is probably the most famous theory in the domain of synaptic plasticity, which gives a plausible explanation for the synaptic adapting mechanism of neurons in the brain. As stated by Donald Hebb:

“When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.”

To be more specific, Hebb's rule takes the view that a synapse between two neurons is strengthened when the firing of one neuron always (or frequently) leads to the firing of the other one.

In the context of artificial systems or more specifically artificial neuron networks, the Hebbian learning rule is formalized to a method determining how to change weights of neural connections. It can be described as the weight between two neurons increases with high correlation between their activations and reduces if they activate separately.

Mathematically, the Hebbian learning rule can be described as follows:

$$\Delta\omega_i = \eta x_i y \quad (8)$$

$$y = \sum_j \omega_j x_j \quad (9)$$

where $\Delta\omega_i$ is the change of the i th synaptic weight of the postsynaptic neuron, η denotes the learning rate, x_i and y represent the i th input for postsynaptic neuron (namely the response of the presynaptic neuron) and postsynaptic response (output activity of the post-synaptic neuron).

Although it plays an important role in revealing the essence of learning and is often regarded as the neuronal basis of unsupervised learning, it has been shown that the standard Hebbian learning rule is not stable, because the weights will keep increasing or decreasing without bounds for any network with a dominant signal, resulting in a divergence.

Hebbian Learning with Decay. In order to avoid the unstoppable strengthening process in the standard Hebbian learning rule, researchers started to look for ways that can enforce constraints onto the learning process. One successful approach is to directly add a decay term into the learning rule. A general form of this version is

$$\Delta\omega_i = \eta x_i y - \lambda \omega_i \quad (10)$$

where λ represents the decay rate. As seen from the above equation, the weight will be stable (when $\Delta\omega_i = 0$) at the point $\omega_i = \eta x_i y / \lambda$.

Oja Learning Rule. Oja learning rule (or Oja's rule) is another variant of the standard Hebbian learning rule that solves the stability problem and learns to compute the principal component from its input stream. Its mathematical formulation is derived from the standard Hebbian learning rule after some simplifications, for the purpose of a concise statement, we omit the derivation and only provide its final expression here as

$$\Delta\omega_i = \eta(x_i y - y^2 \omega_i) \quad (11)$$

It differs from Hebbian learning with decay only in the decay term. That is, in the Hebbian learning with decay, the decay term is a value of scaled weight, while in the Oja's rule, it is a term proportional not only to the weight but also to the square of the neuronal output. In this manner, Oja's rule guarantees that stronger restrictions will be applied on synapses with larger weights and higher postsynaptic activation levels.

BCM Learning Rule. The BCM rule, named after the authors of the 1982 paper, Bienenstock, Cooper and Munro [47], is an unsupervised learning rule for synaptic plasticity. It originates as a simplified mathematical model that captures and models the selectivity of visual cortical neurons and has been successfully applied to other types of neurons later. Similar to the other plasticity learning rules that we have investigated, it also follows the basic principle of the classic Hebbian learning. Discriminatively, BCM introduces a sliding threshold as the regulator that controls the changing of the synaptic strength. The mathematical formulation of BCM rule takes the form of

$$\Delta\omega = \phi(y, \theta) \quad (12)$$

From the perspective of artificial neural networks, the BCM rule regulates the changes on synaptic weights based on the input pattern vector \mathbf{x} and a nonlinear function ϕ . This nonlinear function determines how to modify the weights given the values of post-synaptic neural activity y and a certain dynamic threshold θ . As shown in Figure 17, the synaptic weight will be reduced (with negative ϕ) for a low level of post-synaptic activity ($y < \theta$) and will be increased (with positive ϕ) when the neuronal output value is large ($y > \theta$):

$$\phi(y, \theta) \begin{cases} = 0, & \text{if } y = 0 \text{ or } \theta \\ < 0, & \text{if } 0 < y < \theta \\ > 0, & \text{if } y > \theta \end{cases} \quad (13)$$

The sliding threshold θ is modified over time based on the temporal moving average over recent past activation values of y . The alteration of the threshold determines the weakening or the strengthening of the neuronal connection, leading to a self-regulation of the neural plasticity. The function ϕ and threshold θ are usually expressed as follows

$$\phi(y, \theta) = y(y - \theta) \quad (14)$$

$$\theta = E[y^q] = \sum_k p_k (y_k)^q, \quad (\forall q > 1) \quad (15)$$

where $E[y^q]$ represents temporal average, p_k is the probability of choosing input pattern y_k from the dataset. This learning rule is proved to be stable for any q larger than 1, however, $q = 2$ is often used for the purpose of better simulation [78]. Therefore, we also follow this convention in our study.

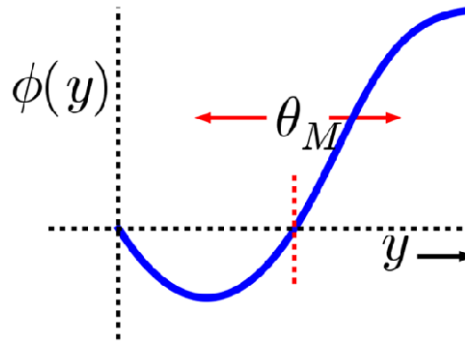


Figure 17 The Shape of Nonlinear Function ϕ and Its Relationship

Practically, for the ease of implementation, the temporal average of neuron activities is always calculated over some recent past. We therefore follow the solution provide by Intrator and Cooper in 1992 [79], as

$$\Delta\theta = \frac{1}{\tau} (y^2 - \theta) \quad (16)$$

where τ is the averaging period.

3.5.3.4 Training Procedure.

To benchmark the capabilities of all synaptic plasticity learning rules mentioned above, the synaptic learning was performed based on the evolution DRE using the weather observations [66]. In the experiments, DRE optimization process was first performed to evolve a good reservoir ensemble structure for the objective problem. Based on the same structure generated, different learning rules were applied separately for a certain number of iterations. Once the weights were strengthened, the whole network was retrained and retested. Diverse learning rates and decay rates (if applicable) were also adopted for each synaptic rule to observe their performance over different settings. Except for data scaling and activation function (will explain later), all other settings of

the DRE model (including settings of the genetic algorithm and the configuration as well as the parameters selection of the network) and data preparation are identical as those in our previous work [76].

After the formation of the ensemble ESN, two types of training are still required for the further synaptic adaptation, i.e., the supervised training of the readout layer and the unsupervised synaptic learning process.

For the readout layer training, it has been proven by many that offline training provides more stable and accurate performance [80], we therefore conducted the readout training in the offline manner, where all neuronal activations stimulated along the training sequence are preserved and collected for the one-off update of the output weights thereafter.

For the synaptic learning, a seemingly online training protocol was adopted. As illustrated in Figure 18, the unsupervised tuning of the reservoir weights is performed simultaneously with the processing of training data. Then the supervised training is carried out as soon as the knowledge of the entire training set has been observed. In other words, the internal weights of the reservoirs are trained prior to the updating of the output layer. Finally, some new data were tested based on the trained internal and output weights to evaluate the learning performance.

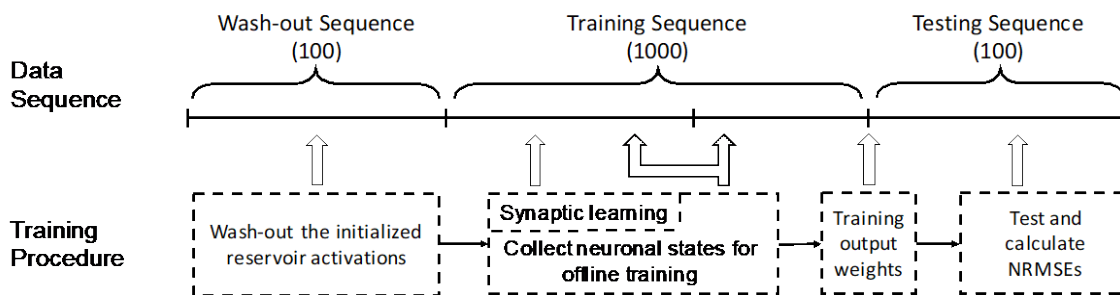


Figure 18 A Diagram of Training Procedures

To search for the optimal solution, synaptic learning rules were tested for various lengths of training iterations ranging from 50 to 900, increasing by 50 steps, at diverse learning rates and decay rates (if applicable). The learning rates of Hebbian learning (with or without decay) and Oja learning rule varied from 0.0001 to 0.003 in increments of 0.0001. For the decay rate of Hebbian learning with decay, it changed from 0 to 0.0001 for every 0.00002. Note that it is actually the classic Hebbian learning when the decay rate takes on value 0. As for the BCM rule, learning rate fell into the range between 0.0005 and 0.01 increment at 0.0005 and the τ took on values from 20 to 300 with increments of 20.

The NRMSEs of the outputs on testing data were calculated and recorded as the measurement of the performance. Results of different learning rules were compared and analyzed.

In order to apply BCM to the reservoir ensemble networks, we have to make some changes to the model settings and parameters to fit the BCM learning rule to the application and benchmark all those synaptic plasticity methods. We then illustrate how we changed the activation function and data scaling of the ensemble model and state the way we train the synapses.

In the previous experiments [76], we adopted the hyperbolic tangent function (also known as tanh function) as the activation function for all computing units. It maps the input values from $(-\infty, +\infty)$ to $(-1, +1)$. This works well for Hebbian learning (with or without decay) and Oja's

rule, however, as for BCM, an inherent characteristic observed in Figure 21 shows that the outputs of neurons should not be negative. For this reason, the tanh function would not be appropriate for BCM learning. We then changed the activation function to one of the sigmoidal membership functions which maps values from $(-\infty, +\infty)$ to $(0, 1)$. The inversion of the activation function was modified accordingly for training the readout layer. The expressions of the sigmoidal function we used and its inverse function are given as follows:

$$f(x) = \frac{1}{1+e^{-2x}} \quad (17)$$

$$f^{-1}(x) = \frac{1}{2}(\ln y - \ln(1 - y)) \quad (18)$$

Furthermore, we also needed to rescale the input and teacher data to match the activation function. We previously scaled both the input and teacher data to $[-0.5, +0.5]$, now, we scaled them to $[0.1, 0.9]$ by multiplying a scaling factor 0.8 and shifting by 0.5. We did not let the data range fill up the entire codomain of sigmoid function $(0, 1)$ to avoid generating excessively large weights.

These new settings for BCM were then applied on all synaptic rules to provide a unified standard and to benchmark their learning capabilities. The results will be analyzed in section 4.4.

3.5.4 Interpretive Reservoir: A Preliminary Study on the Association Between Artificial Neural Network and Biological Neural Network

With several decades of growth, artificial neural network (ANN) and its variants have been widely explored and exploited for a wide variety of applications in different areas, as a state-of-the-art learning technique. In the course of its development, there have been two different perspectives, which in turn, have resulted in two lines of research.

The first research line is primarily promoted by those in the field of computer science. The motivation lies in the great interest in designing novel neural networks and seeking more powerful learning algorithms that can perform complex computational tasks (e.g., classification and functional approximation). Leveraging recent advances in artificial neural networks, many applications, such as image and object recognition, natural language processing, human-computer interaction, games, and autopilot, have already achieved human level or even super-human level performance. In this line of research, most of the prior research were task- or application- driven, with primarily focus on the improvement of result accuracy and precision, from the pure computational perspective. As the neural network architectures became more and more complex, some severe problems emerged. For instance, the predictions or classifications provided by neural networks become more difficult to explain, and the training process gets harder to control in a predictable manner. This increasingly critical issue is well known as the black box problem and many researchers have extensively explored various strategies to better explain the behaviors of black-box classifiers [81-84]. An alternative approach is, instead of examining the mathematical representations in the neural networks, we may look for the answers from the perspective of the neural dynamics of biological neural networks, from which the ANN was originally inspired.

The other research line is related to and motivated by neuroscience, cognitive science, and psychology, etc. Different from the first perspective, what is to be primarily investigated is no longer the computational procedures and mechanisms, but rather the unclear neural and neuronal dynamics (either biological or artificial). Therefore, the other research line combines the computational schemes with the biological behaviors and seeks their underlying associations. It is

expected that this interaction between the biological brain and the artificial algorithm may lead to more insightful understanding about the working principles on both sides. Taking advantage of the most recent neuroimaging technologies, researchers have been able to capture the intensive hidden responses of brain neurons in several ways. In particular, functional magnetic resonance imaging (fMRI) and electroencephalography (EEG) (or a closely relevant method called event-related potential, ERP) are the two most popular measurements of brain activities in the domains of neuroscience and cognitive science. In this line of research, prior studies have primarily focused on investigating and interpreting the cognitive process of humans and have used various types of neural networks as the analytic tool. fMRI has been widely used for those studies, because of its outstanding ability of identifying brain regions involved in the reaction to specific stimuli and constructing the brain functional mapping with rich spatial information by detecting changes associated with blood flow [85]. Some research has used neural networks to segregate and analyze fMRI data to achieve quantification as well as to model brain dynamics. Others have sought to build up the relations between fMRI patterns and certain cognitive processes so as to identify brain activities and even read or reconstruct the visual stimuli seen or imagined by a human subject (known as brain decoding) [86]. On the other hand, given the bi-directionality of the interaction between the biological brain and the artificial network, another branch within this line of research has been under-explored, that is, interpreting the learning scheme (or the black box issue) of ANNs with the support of real brain activity data.

Since the black box issue has limited the development of ANNs for quite some time, investigation on this issue using brain activity data may help to bridge the gap between the biological and artificial neural networks and promote our understanding of the learning scheme associated with ANNs.

3.5.5 Rationale

Memory is fundamental to cognitive learning and the representation of the intrinsic memory mechanism in the context of artificial neural networks has been a long-standing and unexplained problem. We argue that we can partially emulate the intrinsic neural and neuronal dynamics exhibited within the biological brain by indirectly examining extrinsic brain activity characteristics. The objective of this study was to preliminarily investigate and interpret the memory mechanism of a neural network based on some biological observations about brain activity, especially how the intrinsic knowledge is represented and evolved in a biological neural network.

The rationale of this study was derived from a widely-accepted assumption that different people have different memory built upon their distinct knowledge base and diverse experiences. The different experiences result in the changes in brain by creating and eliminating neuronal connections (synapses) as well as strengthening and weakening existing synaptic connections to continuously learn and remember [80, 87, 88]. Those in turn greatly contribute to the very different reactions of people to the surroundings, and consequently result in the significant differences among individuals' brain activities when facing even the same stimulus. It can be therefore assumed that the diversity of individual memory is rooted in the varied brain structures and partially reflected as the ever-changing brain activities measured through many ways (e.g., EEG/ERP). Using these brain activity signals as the training data, we attempted to generate a neural network with certain topology and connectivity so that it could serve as a simplified brain model and approximately mimic the observed brain activities. With the brain signal recorded from different subjects, diverse network models were generated, and through analysis and comparison,

we were able find some measurable indices to explain the representation and principle of memory in artificial neural networks.

3.5.5.1 Brain Activity Measurement

For the measurement and representation of brain activities, the two most common and most frequently used techniques are fMRI and Electroencephalography (EEG).

fMRI, as a hemodynamic technique, measures brain activity by imaging changes associated with blood flow, primarily in the form of blood-oxygen-level dependent (BOLD) contrast. It works based on the finding that cerebral blood flow is spatially coupled with neural activation. Consequently, fMRI is widely used in the study and localization of region-related brain activity.

EEG is an electrophysiological method to monitor the electrical activity of the brain. It is usually operated by placing a number of electrodes on the fixed locations along the scalp and measuring voltage fluctuations resulting from ionic current within the neurons of the brain. Therefore, each channel (a time series signal recorded from one electrode) of EEG signals represents the integrated brain activations of a certain area. However, as an ongoing signal, EEG can be collected without the presence of any particular stimulation. Therefore, it cannot represent and be associated with any specific cognitive process.

Alternatively, ERP as the measured brain response (measured by means of EEG) that is the direct result of a specific sensory or cognitive event, is usually adopted in cognitive neuroscience research to allow some experimental controls over the cognitive state of the subject and to measure dedicated brain response to a specific stimulus. It is calculated by averaging several trials of EEG responses time-locked to the event of interest. Through this way, brain activity not related to the event of interest is mitigated and that related only to the event of interest is emphasized.

For our purposes, EEG/Event Related Potential (ERP) have unique advantages over fMRI, mainly from the following aspects:

- Compared with fMRI, EEG/ERP offer very direct measurements of neural electrical activity. The external appearance and the partial internal mechanism of neuronal activities are the propagation of electrical signals (or spikes), while fMRI is based on the indirect relationship between neural activity and BOLD. EEG is more suitable of the direct expression of brain activities;
- EEG/ERP have excellent temporal resolution (determined by the sampling rate) and relatively low spatial resolution, while fMRI provides relatively low temporal resolution (inherently limited by the slow speed of the BOLD response) and high spatial resolution. Considering the facts that memory itself is a sophisticated time-related mechanism, and additionally, non-volitional memory-related brain activities usually occur and last for a very short time period (in the order of milliseconds) [89], EEG/ERP are thus more capable than fMRI for continuously measuring and recording the time-sensitive target brain activity;
- EEG can be directly collected and sampled as quantized data in a relatively convenient way at a lower cost. However, the recording of fMRI requires large specialized medical facilities, which is much more expensive and introduces difficulties in quantification.

3.5.5.2 Data Acquisition.

For this study, to measure and model the diverse brain activities, we utilized individually unique experience and memory. Data was obtained from another, related project where we collected EEG readings. In that project an ERP biometric protocol called Cognitive Event Related Biometric REcognition (CEREBRE) [90, 91] was employed in the data acquisition process. The

CEREBRE protocol was designed to elicit differential responses across subjects from multiple functional brain systems. It applies control over the cognitive state of participants using carefully designed categories of image stimulation that are likely to cause unique patterns of functional brain activity. With the help of CEREBRE protocol, a more stable measurement is provided, and the disadvantage of remarkable inconsistency and noise level associated with raw EEG signals can be largely diminished.

Figure 19 demonstrates the electrode locations in the CEREBRE protocol, among which, 26 are placed on the scalp, 3 around the eyes to record electrooculogram (EOG) and 1 at the right mastoid as reference.

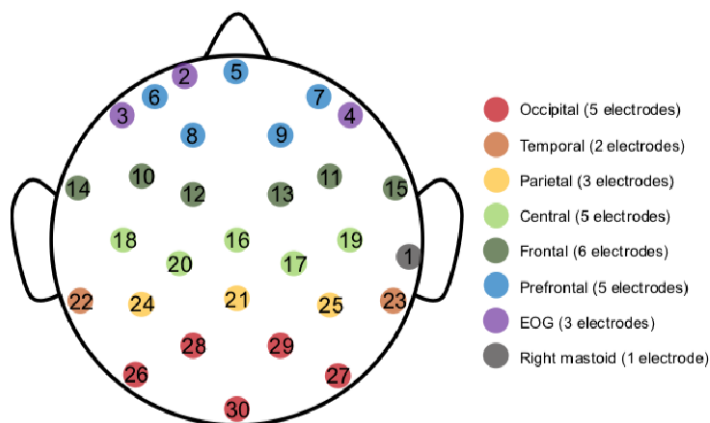


Figure 19 The Spatial Locations of 30 EEG Channels (Electrodes)

In the brainwave collection process, participants were presented with a set of image stimuli. The raw EEG signals were recorded for 1.1 seconds after each switch of the stimuli at a sampling rate of 500 Hz, resulting in a path of time series with 550 data points. Following this way, the unintentional reactions of the human subjects towards the visual stimuli were recorded when they were focusing on watching. We recorded the EEG signals over 10 adult subjects. For each subject, 30 channels of EEG were acquired simultaneously from 30 electrodes placed along the scalp and face, monitoring the electrical activities of different regions in the brain.

Following the CEREBRE protocol, the image stimuli presented to the subjects were organized into six categories: black-and-white Gabor patches (BW Gabor, a set of sinusoidal gratings), black-and-white low-frequency GRE words (BW Text), black-and-white celebrity faces (BW Celebs), black- and-white food (BW Food), color food (C Food) and color targets mixture (C Mix, which mixes up the color images of the first 5 categories). It aimed at evoking the participants' diverse intuitive responses that associated with their unique memory and knowledge. The output from the previous project was analyzed from the neuro-network perspective in this project.

3.5.5.3 Artificial Neural Network Architecture.

With regard to the neural network architecture employed, we chose a variant architecture of reservoir computing (RC). The reason was that, information in a biological brain does not propagate in a single direction; it is diffused along recurrent pathways and feedbacks. The structure

of RC networks is more like that of biological brain and enables such complex signal propagations and neural dynamics compared to feed forward networks. Among echo state network (ESN) [20] and liquid state machine (LSM) [55], we selected the ESN paradigm in this preliminary study in view of its ease of implementation.

Considering the protocol of brainwave collection, in order to fit neural networks to brain activities, EEG/ERP signals can be seen as the output potentials (activation values) of hidden units in a neural network. Based on this assumption, a single neuron was adopted for the processing of each EEG signal channel. Since the EEG/ERP data were collected/computed over 30 channels, the network structure was built with 30 neurons in total, as shown in Figure 20, representing 30 brain areas recorded by the electrodes. This small network is the most simplified model we could come up with. It may have the potential to investigate the relationships between brain regions and provide us with some initial ideas about the effects that network structures may have on the expression of memory.

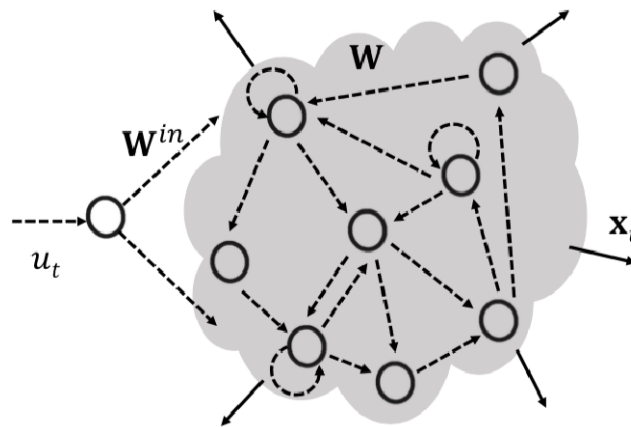


Figure 20 The ESN like Net Structure Adopted

Although an ESN-like RNN was established for brain modeling, we would not be able to build up the network normally due to some tricky issues that need to be dealt with. First of all, determining the network inputs was a tough problem. Generally, in research cases where subjects were presented with image stimuli, either the raw stimulating images or brain signals are used directly as network input. However, our intention was neither to learn the representation of the seen images, nor to identify the on-going brain activities, so neither strategy is suitable and applicable for our case. In addition, it would be unrealistic to directly use the original stimuli as the input for such a simplified model of brain simulation. Furthermore, it's not an easy job to translate the stimuli otherwise in the context of ANNs. As a result, how to properly define the network input becomes very challenging. Similarly, it is not feasible to precisely identify the network outputs. We only adopted non-volitional EEG brainwaves and did not involve or trigger any conscious behaviors [89]. Hence, the EEG signals in this situation should be seen as the internal activations rather than the output activations of the brain, and no output signal could be explicitly defined or recorded during the data acquisition process. In summary, compared with the traditional network implementations, the difference not only lies in the absence of explicit inputs, but also in the absence of clear outputs.

To tackle the issues above, we defined the network inputs as unknown but constant values to be determined in consideration of the way that image stimuli were presented (static image stimuli are presented without changing throughout the recording of each EEG clip). Random initialization of inputs would be implemented. We also got rid of the output layer in the network. The eventually measurable values were the activation values associated with each hidden neuron.

The network structure we constructed with 30 neurons is shown in Figure 20. All dashed lines including the neuronal connections and the input indicate the undetermined parameters and the solid lines illustrate the known internal states (hidden activations) composed of ERP data series. The connection weights are denoted by a 30×1 input weight matrix W^{in} and a 30×30 internal weight matrix W . Accordingly, the updating of the internal activations of the reservoir (also termed as internal states) was processed as follows:

$$\mathbf{x}_t = f(W^{in}\mathbf{u}_t + W\mathbf{x}_{t-1}) \quad (19)$$

where f is the activation function of internal neurons, \mathbf{u}_t and $\mathbf{x}_t = (x_{1,t}, \dots, x_{30,t})'$ represent the input stimulus and internal states at time t .

3.5.5.4 Modeling Method.

In this work, to evolve neural networks based on ERP data, the input and all connection weights were defined as parameters to be trained and the ERP signals were used as the training data. However, given the aforementioned restrictions, it was manifest that we are not able to train the network in the conventional manner, where the internal weights within the reservoir are fixed and only output weights are trained.

The essence of the task is a multivariate time series problem. We thus consider a vector autoregressive (VAR) model as a feasible method to estimate the parameters. Generalizing from the univariate autoregressive model, VAR allows for more than one evolving variable for the analysis of multivariate time series. It describes the evolution of a set of variables as linear functions of their past values (or lagged values) over the same period [92].

Let $\mathbf{Y}_t = (y_{1,t}, y_{2,t}, \dots, y_{n,t})'$ be an $n \times 1$ vector of time series variables. Then a p -th order (also called p -lag) VAR model, denoted by VAR(p) has the form:

$$\mathbf{Y}_t = \mathbf{c} + \mathbf{A}_1\mathbf{Y}_{t-1} + \mathbf{A}_2\mathbf{Y}_{t-2} + \dots + \mathbf{A}_p\mathbf{Y}_{t-p} + \boldsymbol{\varepsilon}_t \quad (20)$$

where $t = 1, \dots, T$, \mathbf{A}_i ($i = 1, \dots, p$) are $n \times n$ coefficient matrices, $\mathbf{c} = (c_1, \dots, c_n)'$ is a fixed ($n \times 1$) vector of constant terms and $\boldsymbol{\varepsilon}_t$ is an n -dimensional vector of error terms made up of serially uncorrelated or independent white noise. Given the observations of y over the entire sample period, \mathbf{c} , \mathbf{A}_i and $\boldsymbol{\varepsilon}_t$ can be estimated with multivariate least squares (MLS) approach.

Remember that for the network adopted, input \mathbf{u}_t is defined as unknown constant parameter and W^{in} is a vector of static parameter terms. Consequently, when multiplied together, they can be combined as a 30×1 constant vector, representing the collection of values fed into each hidden neuron. The combining of \mathbf{u}_t and W^{in} also prevents the equations from ending up with infinite solutions. We hereby define this term as

$$\mathbf{c} = W^{in}\mathbf{u}_t \quad (21)$$

It can also be noted from (19) that the internal states \mathbf{x}_t is a linear function of their first-lag values \mathbf{x}_{t-1} for all variables in the set. Hence when employing linear activation function, the state updating equation (19) fits the VAR model of order 1 (i.e., VAR(1)). We rewrite it below as:

$$\mathbf{x}_t = \mathbf{c} + \mathbf{W}\mathbf{x}_{t-1} \quad (22)$$

The activations of hidden neurons serve as 30 time series variables ($n=30$) in this model. The constant input term \mathbf{c} and internal weight matrix \mathbf{W} are the parameters that need to be estimated using vector autoregressive.

4.0 RESULTS AND DISCUSSION

4.1 Task 2: Performance Assessment and Characterization of Computational Intelligence Approaches in Autonomous Target Tracking

4.1.1 Result Comparison Among RC-based, SDAE-based and CNN-based Trackers.

10 positive and 100 negative samples were selected from the first frame to initialize the network. The training repeats for 20 epochs before the tracking starts. Experiments were developed and implemented in MATLAB. No offline pre-training was applied.

Figure 21 presents the actual and estimated trajectories of the three trackers described above. It is shown that SDAE and CNN provide similar outstanding performance. Although RC-based tracker exhibits worst accuracy in recognizing and tracking, it can still manage to seize the target through the whole process and never loss the track.

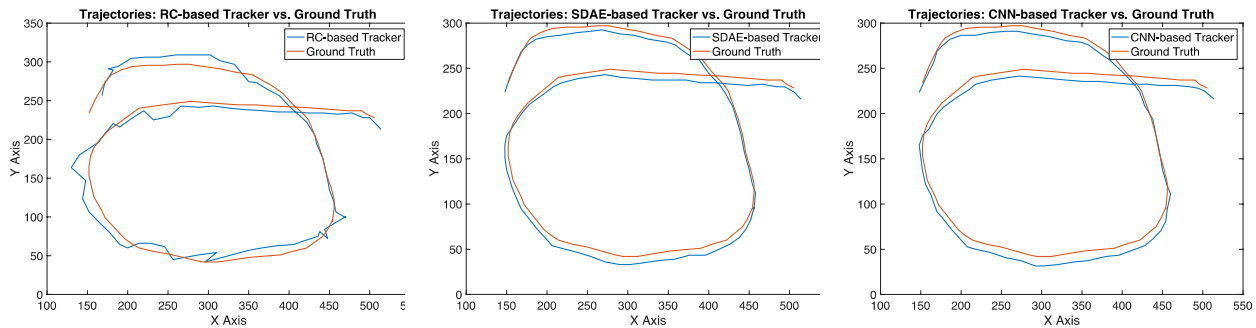


Figure 21 Ground Truth Trajectory vs. Estimated Trajectories of RC-based (left) SDAE-based (middle) and CNN-based (right) trackers

Results that are more illustrative are given in Figure 22-24, which selectively present the tracking results of all three methods in randomly selected frames. This demonstrated that despite the relatively lower accuracy in some frames, the RC network succeeds in representing the input images and achieves acceptable performance even in some complex situation such as the case indicated in frame 44 and 53 when two subjects get close to each other.

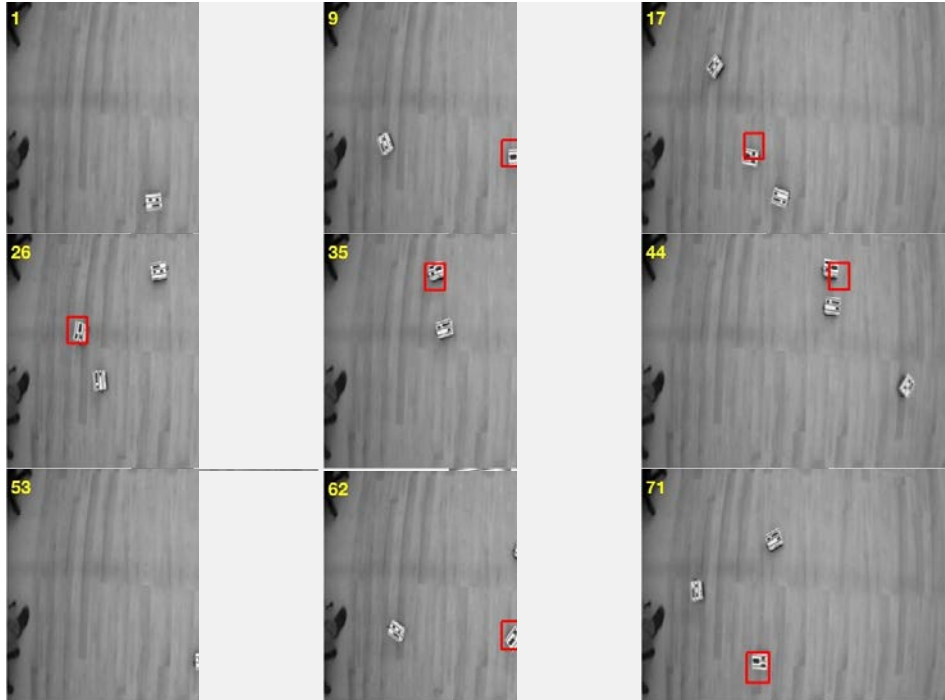


Figure 22 Tracking Results of RC-Based Tracker in 9 Selective Frames

All three tracking learners were developed and implemented in MATLAB. No off-line pre-training was applied. We only executed the trackers on a 2.7GHz 2-core Intel Core i5 processor with 8GB DDR3 memory. The computational efficiency is evaluated in Table 10.

Table 10 Computational Efficiency of Three Trackers on CPU

Tracker	Efficiency	Elapsed Time
RC	0.175fps	452.001s
SDAE	2.839fps	27.823s
CNN	1.849fps	42.735s

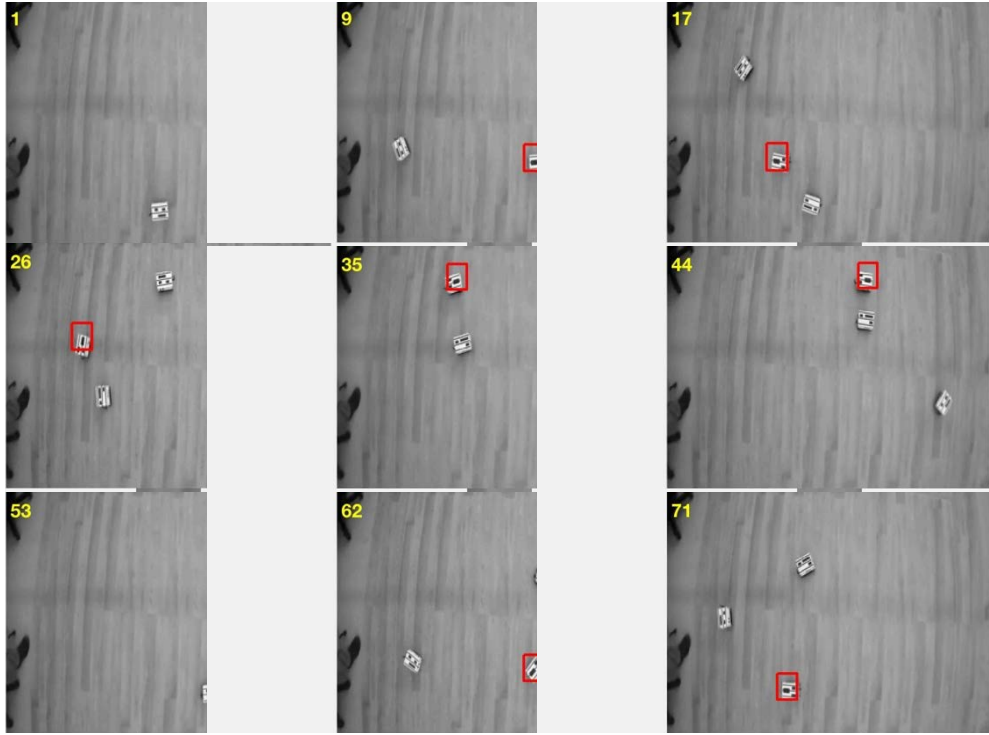


Figure 23 Tracking Results of SDAE-Based Tracker in 9 Selective Frames

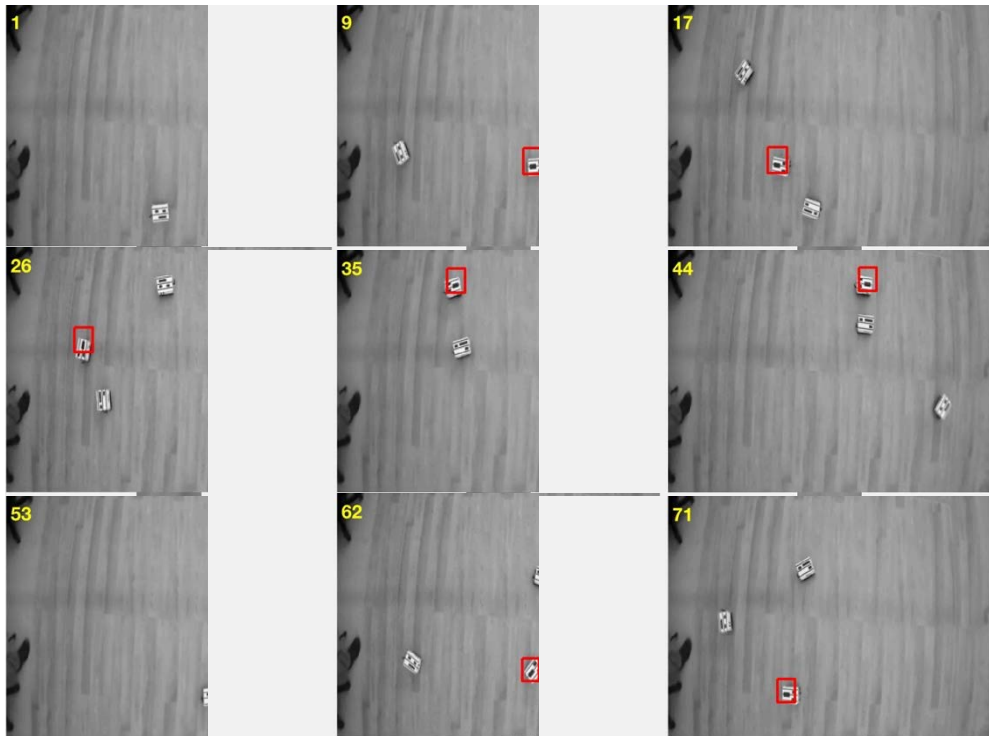


Figure 24 Tracking Results of CNN-Based Tracker in 9 Selective Frames

4.1.2 Speedup on GPU Using Reservoir Computing Based Object Tracker

In the previously proposed method, a standard RC network was adopted as the feature extractor to directly process image inputs. An additional softmax layer connected to the outputs of the RC network plays the role of binary classifier, explicitly distinguishing the object from its background. Particle filter can generate probable candidates and identify the object being tracked among them. By combining the three parts and making RC directly deal with the raw image pixels, it was observed that RCT managed to handle the tracking task with difficulty. Compared with the other two methods respectively based on stacked denoising autoencoder (SDAE) and convolutional neural network (CNN), RCT gave the worst performance on both tracking accuracy and computational efficiency, leading to the conclusion that RC is less effective and favorable on direct and accurate image representation.

Considering that the employment of a GPU platform in the massive computation may potentially promote the efficiency of the RCT, we improved the codes and activated the GPU (an NVIDIA Geforce GTX 780) by invoking the built-in CUDA functions in the MATLAB environment. The acceleration outcomes over all three methods were compared with the tracking experiments executed on CPU (two 2.40GHz 6-core Intel Xeon processors), as shown in Table 11.

Table 11 Speedup of GPU Compared with CPU

	SDAE	CNN	RC
CPU Platform	20.372s/3.878fps	53.783s/1.470fps	549.061s/0.144fps
GPU Platform	5.705s/13.846fps	188.205s/0.420fps	37.319s/2.117fps

4.1.3 Tracking Results of the Combined Method (RCAET)

As in previous experiments, 10 positive and 100 negative samples were selected from the first frame to initialize the latter part of the network. The training repeats for 20 epochs before the tracking starts. Experiments were developed and implemented in MATLAB. No offline pre-training was applied. We executed the tracking on the same CPU platform (Intel Xeon processors) as mentioned above.

Figure 25 presents the actual and estimated trajectories of two trackers, RCAET (combining RC and AE) and SDAE-based tracker. The update threshold of 0.6 was applied on both trackers, while RCAET had an additional give-up threshold 0.4. From this figure, it can be observed that RCAET managed to track the target through the whole process but provided a jagged trajectory. SDAE-based tracker showed a smoother trajectory but lost the target midway through.

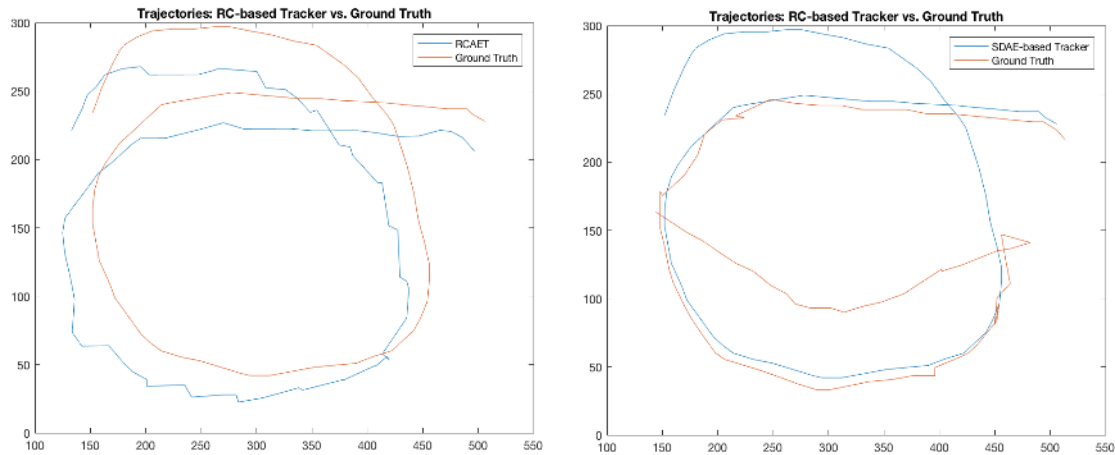


Figure 25 Ground Truth vs. Estimated Trajectories of RCAET (left) and SDAE-based Tracker (right)

In Figure 26 and Figure 27, certain frames were selected to demonstrate the tracking results of the two methods. This showed that in spite of relatively lower accuracy in some frames, RCAET succeeded in tracking the trailer and achieved acceptable performance even in some complex situations such as the case indicated in frame 44 and 53 when two subjects get close to each other. However, due to the cut down of the update threshold, (it originally adopted 0.8), the SDAE-based tracker loses some capacity of fixing the classification error, and finally lead to the failure of the task (shown in frame 62 and 71).

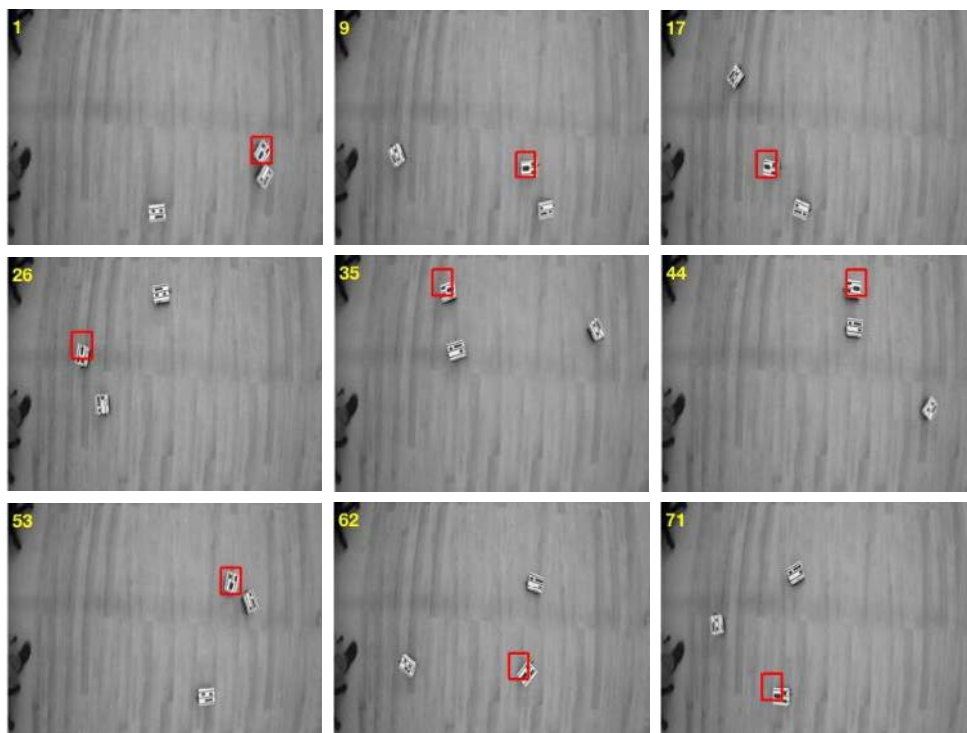


Figure 26 Tracking Results of RCAET in 9 Selective Frames

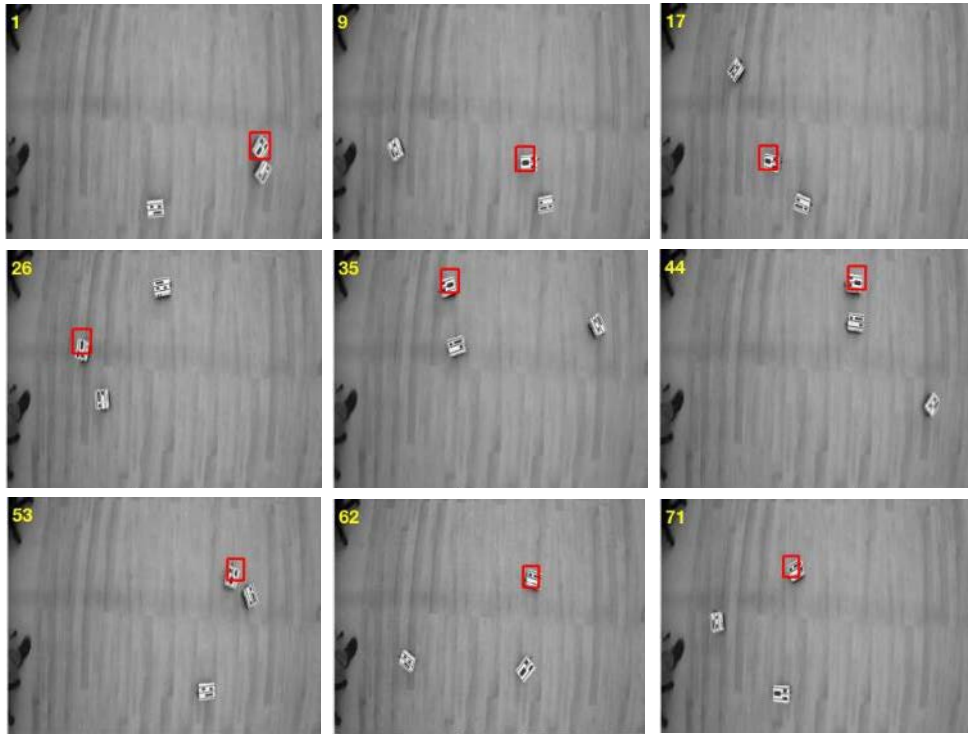


Figure 27 Tracking Results of SDAE-based in 9 Selective Frames

Timestamp data were collected for analysis from the time efficiency perspective. The summary of computational efficiency is then evaluated in Table 12.

Table 12 Computational Efficiency of Three Trackers on CPU

Tracker	Thresholds	Tracker	Elapsed Time
RCAET	$\tau_g = 0.4; \tau_u = 0.6$	19.2997s	4.093fps
RCT	$\tau_u = 0.8$	549.061s	0.144fps
SDAE-based Tracker	$\tau_u = 0.6$	15.454s (fail)	5.112fps (fail)
	$\tau_u = 0.8$	20.372s	3.878fps

This shows that by utilizing the RC prediction model, our new paradigm can improve efficiency that was comparable or even better than the original SDAE-based tracker. Although the SDAE-based Tracker with a τ_u of 0.6 consumed the shortest time, it failed to track the object throughout complete video sequences.

4.2 Task 3: Dynamic Ensembles of Reservoir Networks for Multi-Object Pattern Recognition

The dynamic reservoir ensemble model was tested on the temperature prediction task. The experiment ran for 69 iterations before it stopped. For all three models (standard RC, static reservoir ensemble and dynamic reservoir ensemble), testing data was used for the NRMSE calculation. NRMSEs of the standard RC and SRE were calculated as the average of 10 runs with different random weights. For DRE, the weights were first generated and then left unchanged in the evolutionary process, 5 different sets of random weights were initialized ahead of time and 5 runs were performed for each set. NRMSEs under this model were the average over 25 runs.

The optimization result over 69 generations is displayed in Figure 28, where the red dots represent the mean fitness value of each generation and the blue dots refer to the best fitness within each generation. It shows clearly that for the initial generations, the fitness values of the population are extremely large, as the genetic algorithm iteratively searches in the space and evolves the candidate solutions, better individuals are derived and the fitness function comes to a convergence. The bottom of this figure indicates the best chromosome.

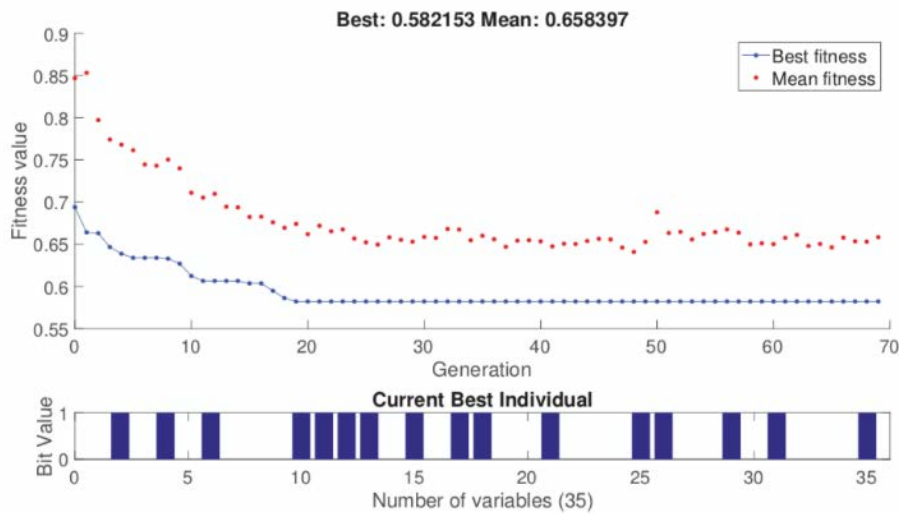


Figure 28 Optimization process and best solution of dynamic reservoir ensemble

According to this chromosome, the best reservoir ensemble structure gained can be easily reproduced as shown in Figure 29.

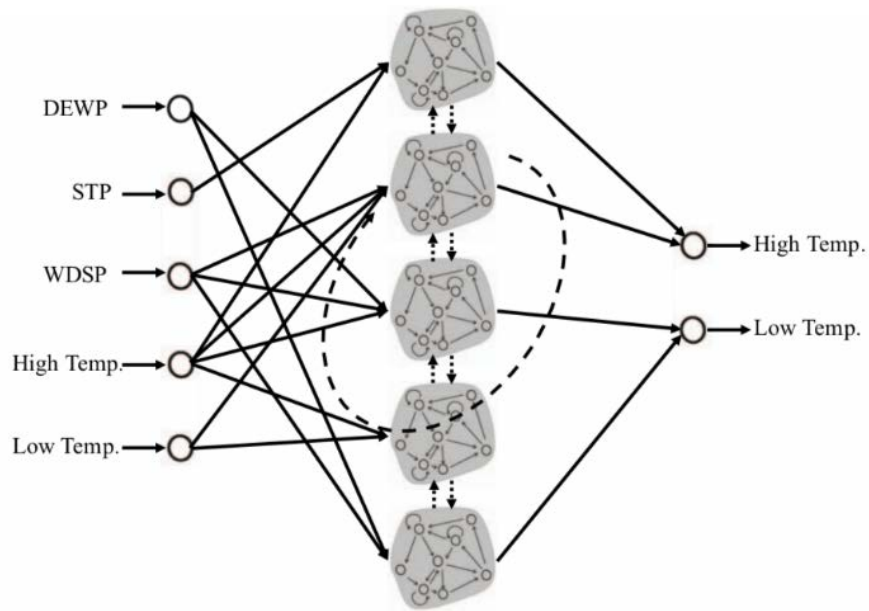


Figure 29 Optimal Reservoir Ensemble Structure Constructed by the Dynamic Reservoir Ensemble Model

Figure 30 shows the testing NRMSE of all the structures implemented. As the labels indicated, “Standard” refers to the two standard structures implemented. “1-I/O Neuron” is the one with a single input and a single output, “5I-2O Neurons” represents the model with five inputs and two outputs. “SRE” represents the static reservoir ensemble model used in this experiment, namely 5I-5R-1O. “DRE” is short for dynamic reservoir ensemble. It can be observed from those values that the structures obtained from DRE model always outperform the other structures on testing performance (with the worst case of DRE 0.62564 and 0.67456 still better than the best case of the other structures 0.70517 and 0.75923) and exhibit very stable performance.

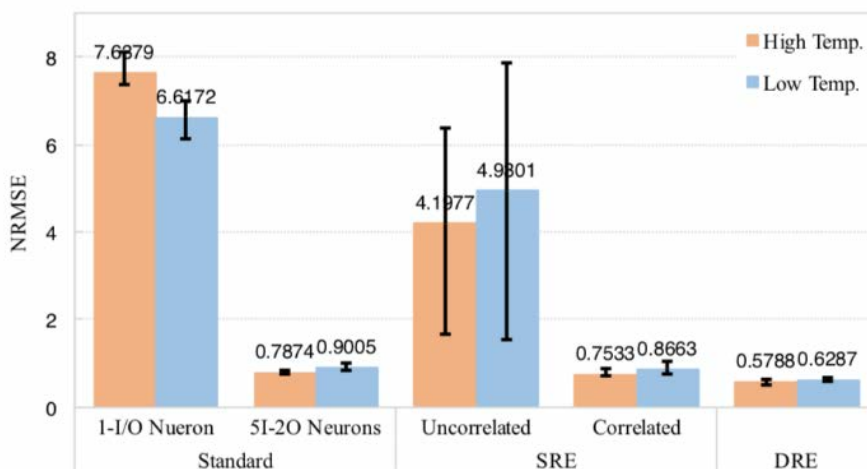


Figure 30 Testing NRMSE of All Models

Combined with some more intuitive results shown in Figure 31, Figure 32 and Figure 33, it can also be observed that the standard structure is unsolvable for single (either high or low) temperature prediction. Therefore, it is found that the static reservoir ensemble without internal correlations is also unsolvable for high and low temperature prediction, because it is similar to the standard model except for the output layer. However, the linear combination in output layer is far from capable of handling multiple sets of dynamics.

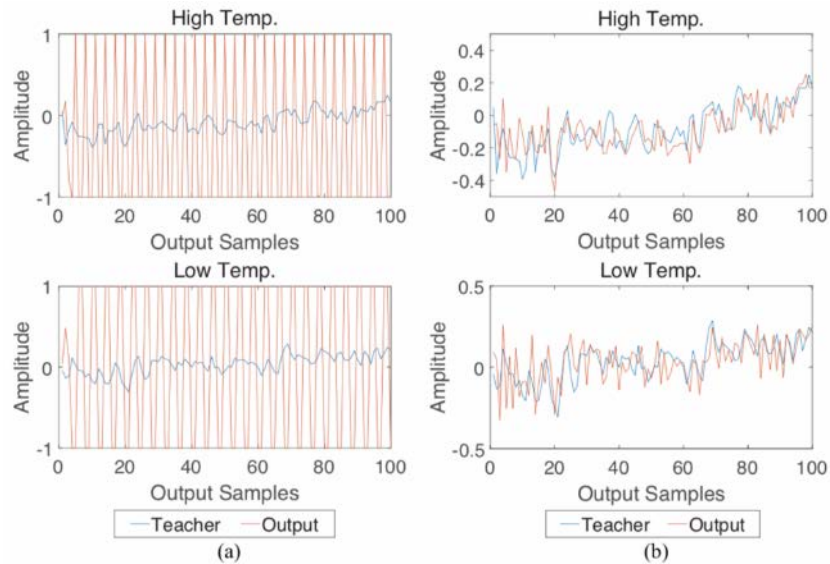


Figure 31 Performance of Standard Model (a) with 1 I/O Neuron; (b) with 5I-20 Neurons

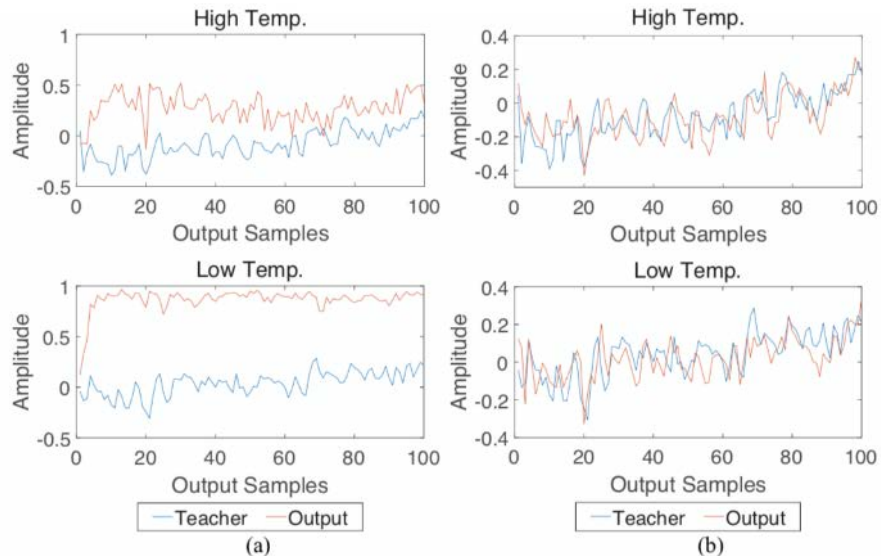


Figure 32 Performance of SRE Model without (a) and with (b) Inter-Reservoir Correlations

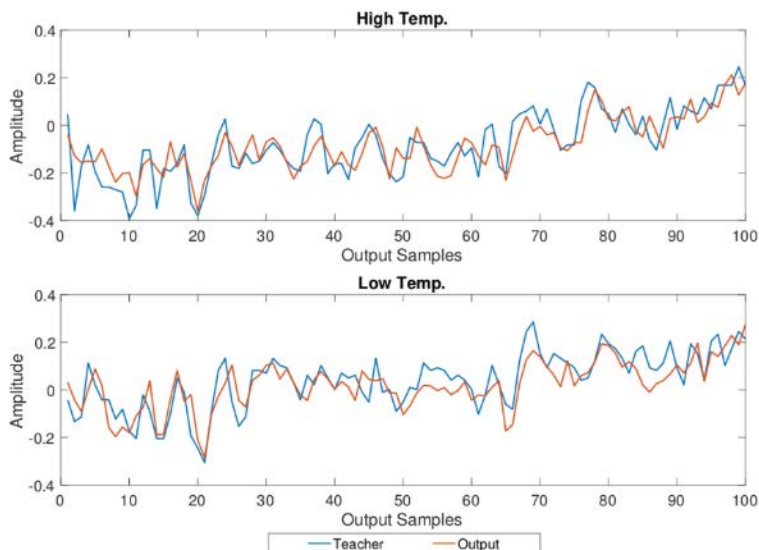


Figure 33 Performance of DRE (a) Prediction Result of High Temp.; (b) Prediction Result of Low Temp

Another interesting finding was that the static reservoir ensemble with correlations had better performance than the standard model with five input neurons and two output neurons. Note that both structures introduce some degree of dependency between the five input signals. For the standard model, due to the normal distribution of connections and weights over the entire reservoir, the dependency of the five signals can be roughly treated as equal. However, in the reservoir ensemble model, the higher sparsity between reservoirs leads to a biased dependency over different sets of dynamics. This may potentially be an explanation of this result. In Figure 37, the best performance of temperature prediction is achieved using the proposed dynamic reservoir ensemble model.

4.3 Task 4: Human Category Learning Inspired Classification Network

4.3.1 Experiment Setup.

Input and Output Sequence.

In this experiment, we used two different types of time sequence as inputs (shown as Figure 34 (a)). A random sequence (between 0 and 1) was adopted as category 1 (C1) and a sinusoidal wave with 10% noise as category 2 (C2). The input sequence of category 2 was derived using the following equation:

$$u_{C2}(i) = \sin \frac{\pi}{16} i + 0.1\epsilon \quad (23)$$

where i is the index of the current data sample, ϵ is a random error.

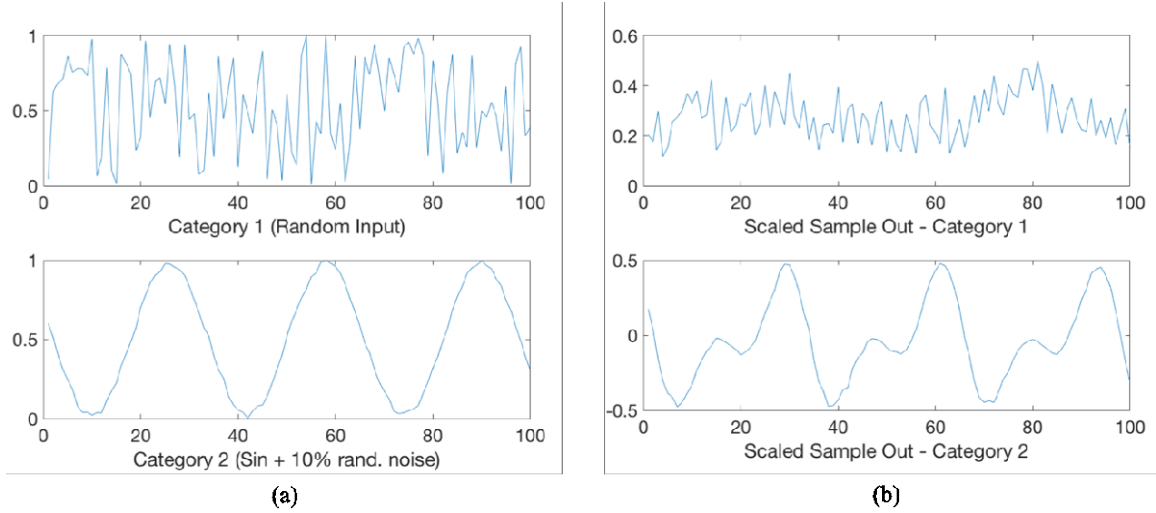


Figure 34 Model Input (a) and Output (b)

The model is supposed to distinguish between the two categories and learn a different nonlinear autoregressive moving average (NARMA) function for each of them. The expected (teacher) outputs are shown in Figure 34 (b) and the NARMA signals for C1 and C2 are described as:

$$y_{C1}(n) = 0.1u_{C1}(n-2) + 0.05u_{C1}(n-2) \sum_{i=6}^{10} (u_{C1} - i) + 0.2u_{C1}^2(n-5) + 0.5 \quad (24)$$

$$y_{C2}(n) = 0.3u_{C2}(n-1) + 0.05u_{C2}(n-1) \sum_{i=1}^5 (u_{C2} - i) + 0.5u_{C2}(n-1) \cdot u_{C2}(n-9) + 0.5 \quad (25)$$

Model Parameters.

In the ESN, one reservoir with 100 neurons and 10% connectivity was constructed. Two output neurons with feedback to the reservoir were adopted, one for each category. Spectral radius was set as 0.8, hyperbolic tangent (tanh) activation function was employed for all neurons.

BCM learning was used as synaptic training of the internal weights of reservoir. $\tau = 1$, $\eta = 0.007$. 100 steps of synaptic learning were applied for each category in each iteration.

The model was trained for 5 iterations, within each iteration, the two categories were trained successively. For each category, the model was fitted to a segment of 500 samples of that category, in which a 100-sample initial run was adopted to washout the internal states. The last 100 samples were for verification. After the 5-iteration training, the trained model was tested on a new testing segment of 100 samples.

For comparison, we also implemented a control experiment, in which the model was trained only once. 1500 samples were used for this one-time training to match the data volume used for training in the iterative learning.

4.3.2 Results.

When training and testing on one category, we did not cut off the output layer of the other category, so that its output values can uninterruptedly make effect on the internal states of the reservoir as well as the output of the true category. This is a very important feature of our

implementation. For one thing, our biological cognitive system is not a simple accumulation of exclusively trained “classifiers” that are non-interfering with each other. Instead, it is a highly-sophisticated composition where different components are working with frequent interaction. Therefore, this implementation mimics the real scenario of category learning, where when learning a new concept, the previously learned knowledge or skills will still play roles in one way or another. It is the wonder of the brain that even though all the time in noise, it can still work steadily and efficiently. For another, the reservation of feedback connections guarantees the significance of the model. Without the output feedback connections, it is no different from training two individual ESNs separately. Next, the test results of one-time training and iterative training will be provided and analyzed.

The test result on both outputs after one-time training were plotted in Figure 35. Since we did not remove the other output layer when processing one category, there are some unimportant signals generated from that output (as shown in the lower left and upper right of Figure 35). Due to the interference between the categories and the biased training, the error is relatively large, especially for the sinusoidal wave.

For iterative training, it can be observed from Figure 36 that both categories have been correctly recognized and accurately predicted.

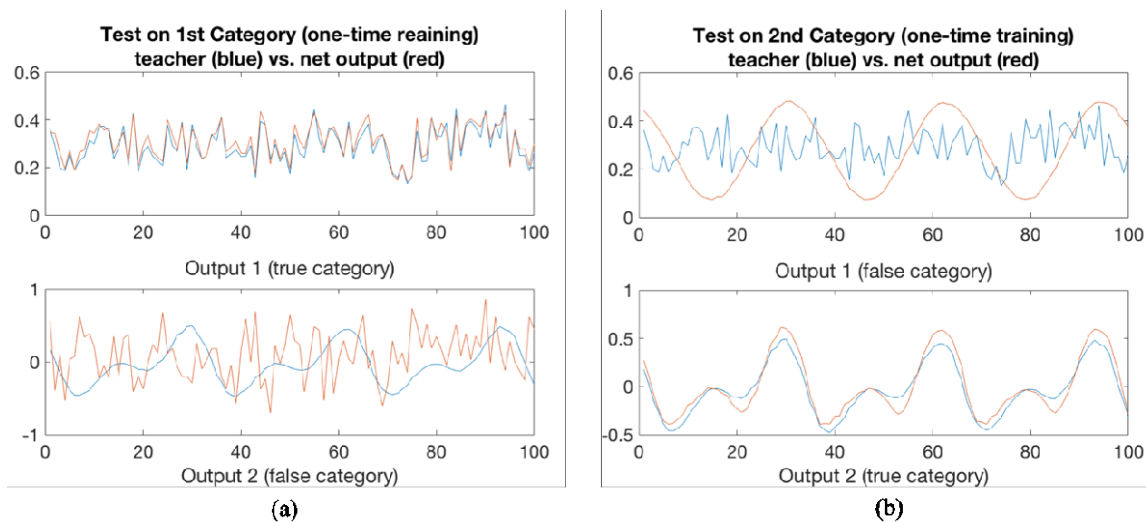


Figure 35 Test Results on Category 1 (a) and Category 2 (b) for One-time Training.



Figure 36 Test Results on Category 1 (a) and Category 2 (b) for Iterative Training.

The normalized root mean square errors (NRMSEs) on both categories for the two training schedules are listed in Table 13.

Table 13 Test Errors (NRMSE) on Both Categories for One-time Training and Iterative Training.

Test on	One-time Training		Iterative Training	
	Out 1	Out 2	Out 1	Out 2
Category 1	0.35936	(1.6398)	0.29822	(2.9154)
Category 2	(2.0142)	0.35166	(1.7771)	0.061255

The values in parentheses are the NRMSEs for the false category (the category different from the one at input) hence, they are not important. The accuracies on both categories are significantly improved, especially for category 2. It shows that the iterative training process can enhance the performance of this category learning task and balance the model over both categories.

Based on the results of this study, it can be concluded that when combined with synaptic learning and trained over multiple iterations, the ESN with one reservoir, divergent output layers and output feedback connections is capable of learning different categories and the associated computational rules from time sequences. Iterative training is helpful in the formation of a balanced model.

4.4 Task 5: Evolutionary Adaptation for Reservoir Network Optimization

4.4.1 Results of Principal Neuronal Reinforcement

The results for this study were obtained from the averaging of the data obtained over 10 repetitions of the experiment. The errors presented are that of the testing data, which is indicative of the network performance. The NRMSE of the experiments are displayed in Figures 37(a) - (c), presenting the effects of strengthening, weakening and combination respectively. In these graphs, O1 and O2 refer to the NRMSE of standard ESN outputs 1 and 2, while the corresponding dotted lines refer to the NRMSE of the PNR trained outputs. It was observed that for ESN of neuron size 100 - 300, there isn't a clear distinguishable improvement in the errors observed. This is due to the fact that the errors of these networks were quite low to begin with, and thus the effects of the

plasticity updating were not apparent. However, as the network size continues to increase, the improvement becomes obvious in all three cases.

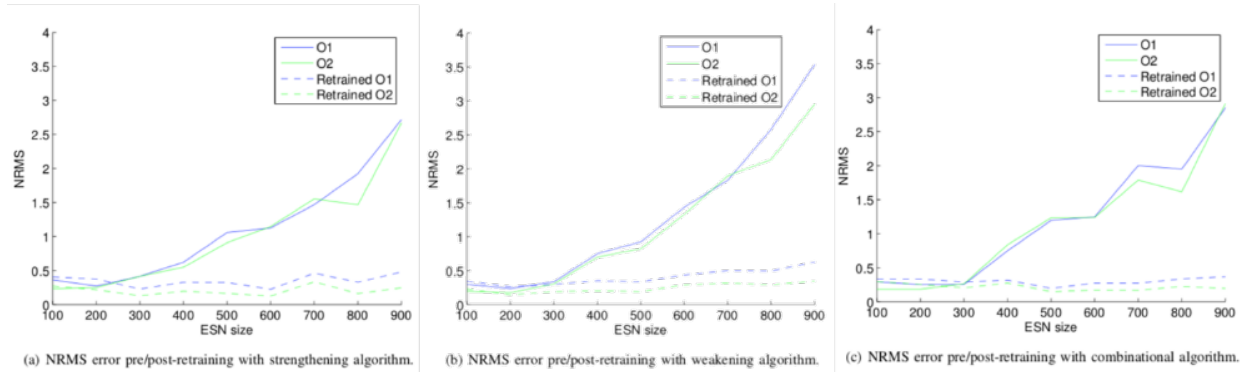


Figure 37 Comparison of NRMS Error Pre/Post-Retraining Performance

It is very clear from these figures that the pre-PNR training output NRMSE becomes increasingly large as the networks size increases, and that upon applying the PNR training, the NRMSE settle at below 0.5 for all cases.

In contrast to this, Figures 38 and Figure 39 present an example of the efficacy of the proposed PNR rule in improving performance. Figure 38 shows the ESN trained with 900 randomly initialized neurons. It is clear that a large error is present, i.e., the NRMSE errors of larger than 1.0 and a large visually verifiable mismatch between the actual (green) and target (blue) outputs. In Figure 39, the internal weights had both the strengthening and weakening schemes applied, and the error is obviously smaller (i.e., the NRMSE errors of approximately 0.4 or lower), and the output is now much more representative of the desired output.

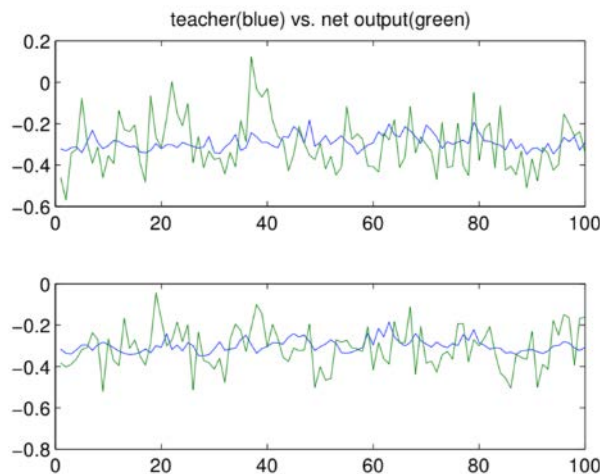


Figure 38 Trained ESN Performance from Randomized Initialization

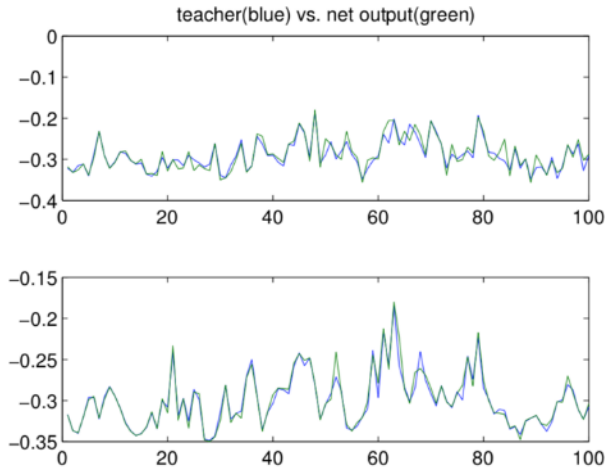


Figure 39 Retrained ESN Performance from Updated Weights

Looking at the results of for the NARMA dataset, we can see in Figure 40 that a similar trend was observed. There is a general increase in NRMSE as there is an increase in ESN size. Once again, upon the application of PNR update, there is a clear improvement in the performance of the system. Note that the error rate for 100 neurons is clearly higher than that of the other neuron sizes. This is likely due to the fact that 100 neurons are not enough to fully capture the dynamics of the system, thus the learning rule had limited effects. This was verified with an exploratory search and found the optimal starting neuron size to be around 280 neurons.

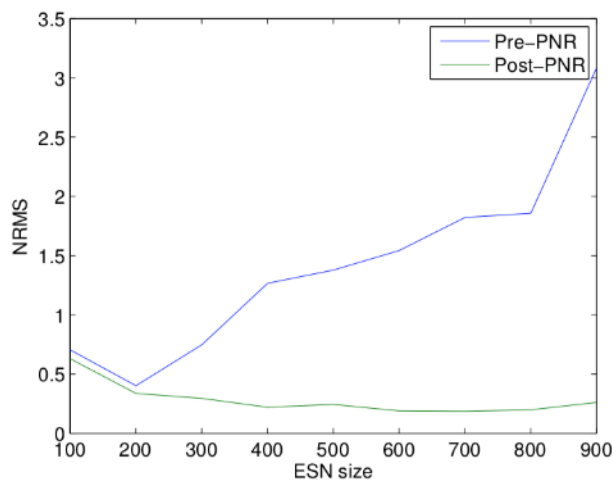


Figure 40 NRMSE Pre/Post-Retraining for NARMA Dataset

The experiments into the algorithm's effects on connectivity also showed positive results. From Figures 41(a) - (c), there is a general increase in the initialization errors present as the connectivity increases. However, it is apparent that upon the application of the PNR rule and retraining, the error for all the connectivity were reduced.

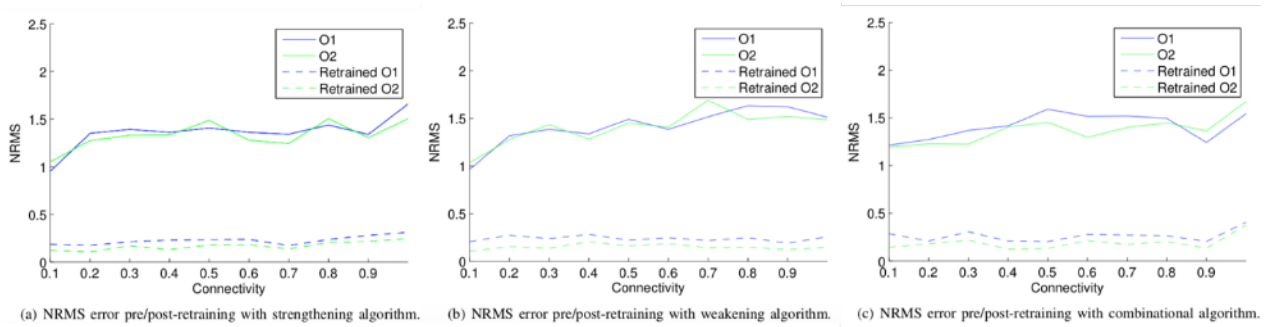


Figure 41 Comparison of NRMS Error Pre/Post-Retraining Performance

In Tables 14 and 15 we can see that the PNR outperforms the Anti-Oja rule in improving the performance of the system for both the benchmark dataset and the NARMA dataset. For the benchmark dataset, the Anti-Oja rule was not able to retrain the ESN to converge when the starting error was large, causing the retrained error to settle at around 1.1. For the NARMA dataset, it was seen that the PNR scheme also outperformed the Anti-Oja rule, with the output error settling at around 0.264 and 0.159 for outputs 1 and 2 respectively compared to the 0.497 and 0.369 of the Anti-Oja scheme.

Table 14 Average NRMSE Over 100-900 Neurons After Plasticity Rule (Toolbox Dataset)

	PNR	Anti-Oja
Average NRMSE	0.242	1.146

Table 15 Average NRMSE Over 100-900 Neurons After Plasticity Rule (NARMA Dataset)

	PNR		Anti-Oja	
	Output 1	Output 2	Output 1	Output 2
NRMSE	0.264	0.159	0.497	0.369

4.4.2 Neuronal Plasticity Inspired Reservoir Ensemble Optimization

Enhanced Dynamic Reservoir Ensemble with Principal Neuron Reinforcement

The experiments were conducted with a reservoir ensemble consisting of 5 reservoirs, under two settings: with 40/100 neurons for each reservoir.

Four configurations were investigated in this study. Upon training the system with the GA method, the system was further trained under the following scenarios:

- **Applying PNR with respect to entire ensemble:** This configuration was chosen as it is the most straightforward and simple combination of the two methods. The PNR method is applied directly to the structure determined by the GA.
- **Applying PNR with respect to each reservoir in the ensemble:** This configuration was chosen to ensure an equal influence on each reservoir by the PNR. The first configuration chooses principal neurons from all the output weights, which may result in bias over different

reservoirs. For example, more neurons could be chosen from one reservoir than another. By performing the PNR individually and separately on each reservoir in the ensemble, the same number of principal neurons will be chosen in each reservoir.

- **Applying PNR to principle neurons in dominant reservoirs in the ensemble:** As the connections to the outputs vary among the reservoirs in the ensemble, it is intuitive that certain reservoirs may be more dominant than others. This configuration attempts to strengthen only the principal neurons identified in the dominant reservoirs.
- **Applying PNR to all neurons in dominant reservoirs in the ensemble:** Similar to the previous configuration, the dominant reservoir is determined. For this case, all the neurons in the dominant reservoir is identified as principal neurons. This in effect strengthens all connections in and out of the dominant reservoir.

For the PNR training process, the output weights are ranked from high to low, where the neurons connected to the highest 10% of the weights are considered the principal neurons. All weights that are connected to these principal neurons are increased by 1% and the system is retrained. This is repeatedly performed until no further improvement in the output error is seen.

All the experiments were also repeated with the reservoir size of 100 for each reservoir to investigate the effect of having a larger dynamic system on the performance of the output.

4.4.2.1 Strengthening Principal Neurons with Respect to Ensemble.

Under the first scenario, the DRE method was first executed to optimize the reservoir ensemble with GA, then PNR was directly applied to the optimized reservoir ensemble structure. The weights to both output neurons are ranked from high to low, and the reservoir neurons connected to the top tenth of the weights were labeled as principal neurons. PNR was applied separately, first to the principal neurons identified from the weights to only output 1, then to the principal neurons identified from the weights to only output 2, finally to both simultaneously. The results are displayed respectively in Table 16 and Table 17.

Table 16 Strengthening Principal Neurons with Respect to Ensemble (40 neurons per reservoir) - Instance 1

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/40 neuron)	0.6431	0.5231
Strengthened by output 1	0.6635	0.5175
Strengthened by output 2	0.6750	0.5186
Strengthened by both	06833	05116

Table 17 Strengthening Principal Neurons with Respect to Ensemble (40 neurons per reservoir) - Instance 2

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/40 neuron)	0.6788	0.5628
Strengthened by output 1	0.6792	0.5629
Strengthened by output 2	0.6708	0.5890
Strengthened by both	0.6712	0.5889

It can be seen that for output 2, the NRMSE is reduced when PNR is applied in all three cases where the opposite is seen with output 1. However, this effect is not consistent across multiple tests. As can be seen in Table 17, the error of output 1 has been reduced while the error for output 2 has increased. It was discovered through repeated testing that each GA determined structure usually yielded one output that could be improved using the PNR method while the other output's performance could be weakened.

The experiment was also repeated with 100 neurons in each reservoir. Result is shown in Table 18. In this configuration, the same trend was observed where one output was likely to gain an improvement after PNR being performed while the error for the other output increases. It is perhaps important to note that though there is a change in value for the errors, visually it is not noticeable. As it can be seen in Figure 42 and Figure 43, the two are virtually indistinguishable.

Table 18 Strengthening Principal Neurons with Respect to Ensemble (100 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/100 neurons)	0.6689	0.6295
Strengthened by output 1	0.6691	0.6298
Strengthened by output 2	0.6331	0.7661
Strengthened by both	0.6234	0.7669

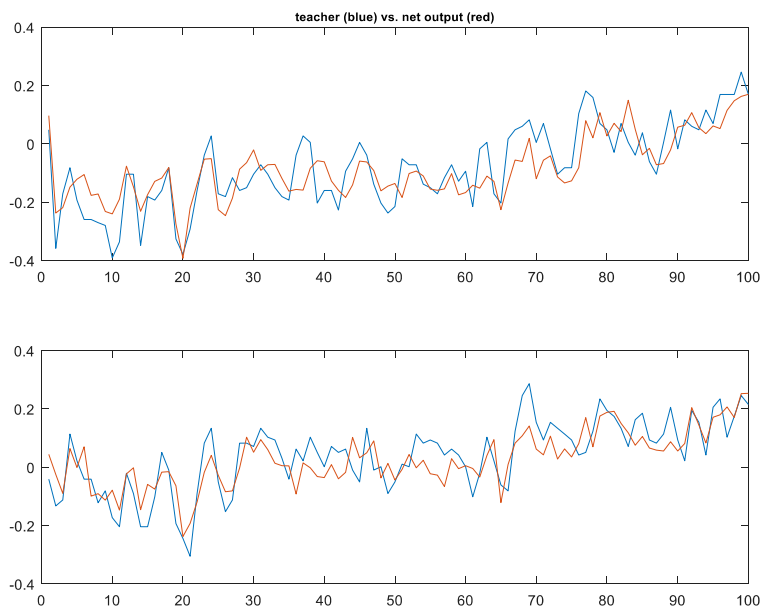


Figure 42 Output 1 and 2 Optimized with GA (40 neuron)

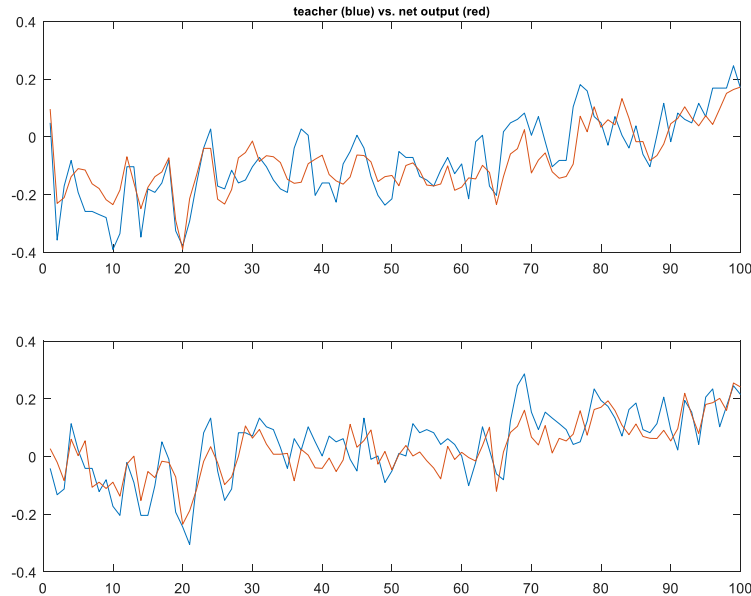


Figure 43 Output 1 and 2 Optimized with GA and PNR (40 neuron)

4.4.2.2 Strengthening Principal Neuron with Respect to Each Reservoir.

Under the second scenario, PNR was applied to each reservoir separately. This case did not yield a clear improvement on the error for either outputs. Table 19 shows typical results for this scenario, where there may be a small improvement on one of the outputs and a small increase in error for the other. This was also tested for the configuration 100 neurons per reservoir, a similar type of improvement is seen in Table 20.

Table 19 Strengthening Principal Neuron with Respect to Each Reservoir (40 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/40 neurons)	0.6431	0.5231
After Strengthening with PNR	0.6583	0.5220

Table 20 Strengthening Principal Neuron with Respect to Each Reservoir (100 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/100 neurons)	0.6689	0.6295
After Strengthening with PNR	0.6745	0.6226

4.4.2.3 Strengthening Principal Neurons in Dominant Reservoirs.

The third configuration carried out PNR on one reservoir that was determined to be dominant. To find this, the output weights of each reservoir was summed and the one with the largest magnitude was deemed to be the dominant reservoir. It was noted that having a larger summed weight suggests a higher importance of a specific reservoir. This configuration often resulted with

the error being decreased for both as seen in Table 21. However, for the 100-neuron configuration, we can see from Table 22 that this is not true, where the NRMSE for output 1 barely changed and increased for output 2.

Table 21 Strengthening Principal Neurons within Dominant Reservoirs (40 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/40 neurons)	0.6431	0.5231
After Strengthening with PNR	0.6425	0.5176

Table 22 Strengthening Principal Neurons within Dominant Reservoirs (100 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/100 neurons)	06689	06295
After Strengthening with PNR	06688	06329

4.4.2.4 Strengthening All Neurons in Dominant Reservoirs.

The final scenario tested was to select a principal reservoir, and strengthen all the connections to the neurons within it. From the results in Table 23 and Table 24, the changes to the errors of this configuration were usually very small for both the 40/100 neuron configurations.

Table 23 Strengthen Connections to all Neurons within Dominant Reservoir (40 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/40 neurons)	0.6431	0.5231
After Strengthening with PNR	0.6425	0.5240

Table 24 Strengthen Connections to all Neurons within Dominant Reservoir (100 neurons per reservoir)

Steps	NRMSE of Output 1	NRMSE of Output 2
DRE (5 reservoir/100 neurons)	06689	06295
After Strengthening with PNR	06688	06329

The results of the four scenarios illustrate that when being directly applied on the DRE model, PNR can neither gain stable improvements nor well handle the initialization issue.

4.4.2.5 Enhanced Dynamic Reservoir Ensemble with Classic Synaptic Plasticity

To test the capabilities of all synaptic plasticity learning rules mentioned and explore their potentials to be combined with DRE model, a set of experiments were performed based on the weather prediction application implemented in our previous work. Using the activation function and scaling settings described in the BCM part of section 3.5.3, we tested all the synaptic rules to provide unified standard and benchmark their learning capabilities.

Based on the new activation function, the reservoir ensemble structure obtained through the GA evolution is given in Figure 44. The NRMSEs of such structure are 0.5642008 for output 1 (High Temp.) and 0.6090616 for output 2 (Low Temp.).

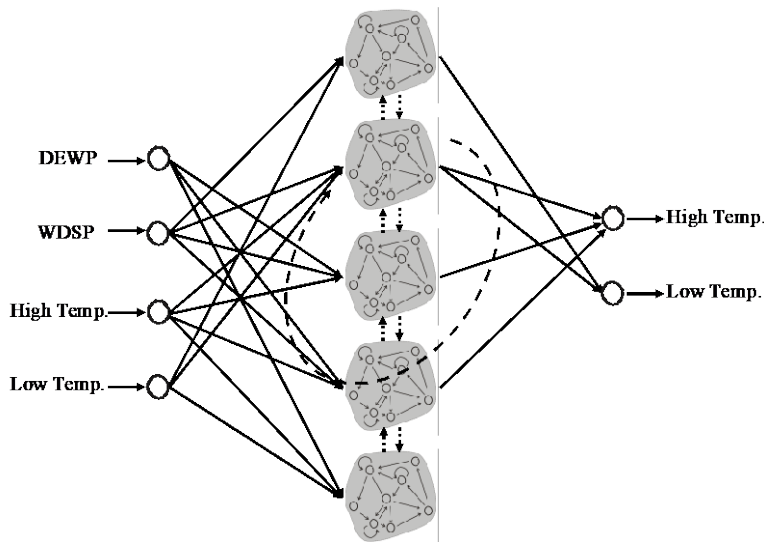


Figure 44 Reservoir Ensemble Structure Constructed by DRE Process

During evaluation, we observed that the two outputs are not necessarily improved together. In fact, their optimal improvements are usually achieved under different settings. Considering this, we mainly focus on the average over two outputs, and once the lowest error on the average value is found, we go back to check out the separate improvement of each output. Without applying neural plasticity, the NRMSEs are 0.56420, 0.60906 for output1 and output2 respectively and 0.58663 on average. The evaluation of the learning rules is provided below.

For Anti-Oja learning, the lowest error rates are 0.56348 and 0.60299 for individual outputs and 0.58324 on average. The learning performance over the entire setting space is exhibited in the form of color map (see Figure 45), where the darker the blue, the lower the error, and therefore the better the learning performance. Similarly, the darker the red, the worse the learning rule worked. The learning trend of Anti-Oja is shown in Figure 46.

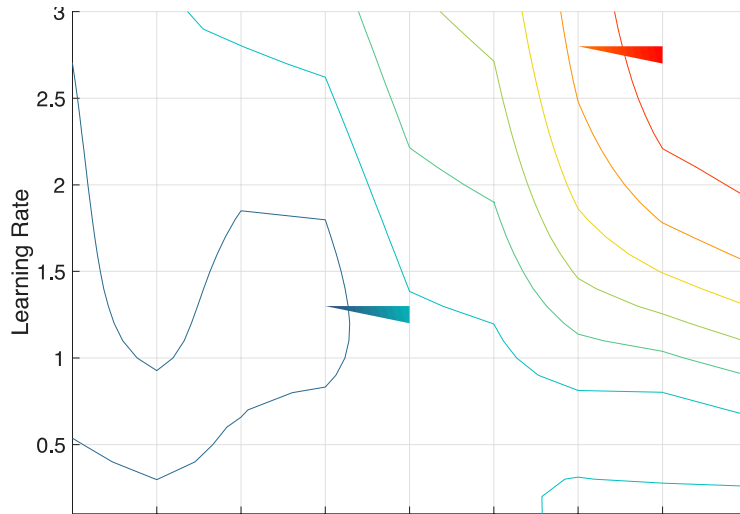


Figure 45 Color Map of Anti-Oja

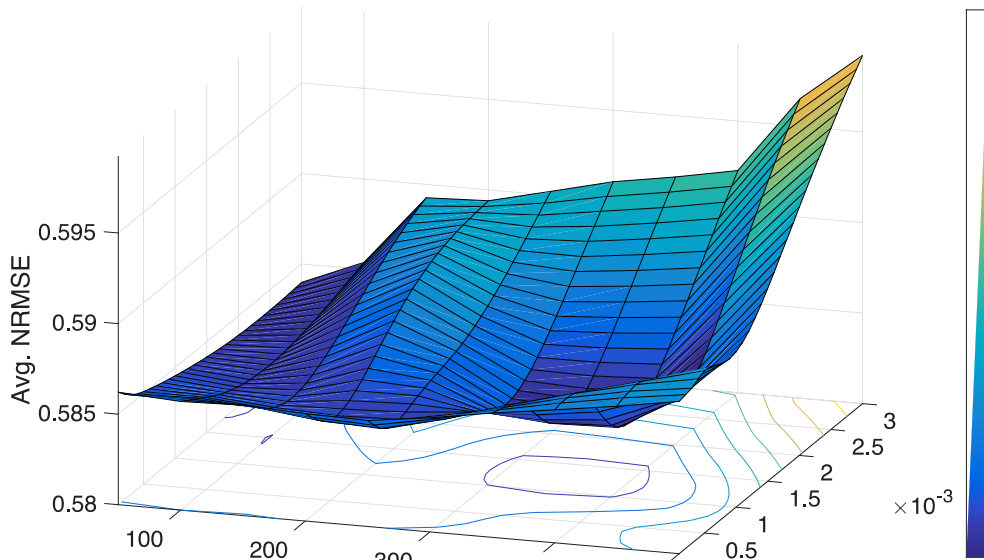


Figure 46 Learning Trend of Anti-Oja

For Hebbian learning and Hebbian learning with decay, we put the results together. Since there are 3 tunable parameters, learning rate, decay rate and training iterations, we cut the 4-dimensional result chunk based on different values of decay rate. Specifically, when decay rate equals 0, it represents the classic Hebbian learning rule. The best case under each decay factor is shown in Figure 47. We notice that the classic Hebb's rule with no decay gets the optimal improvement.



Figure 47 Hebbian Learning's Best Case for Each Decay Rate

We hence visualize the color map and learning trend of classic Hebbian learning in Figure 48 and Figure 49. When the learning rate is 0.0012, training for 350 iterations without decay, Hebbian learning achieves the lowest errors 0.57478, 0.55433 and 0.59523 for average output, output1 and output2 respectively.

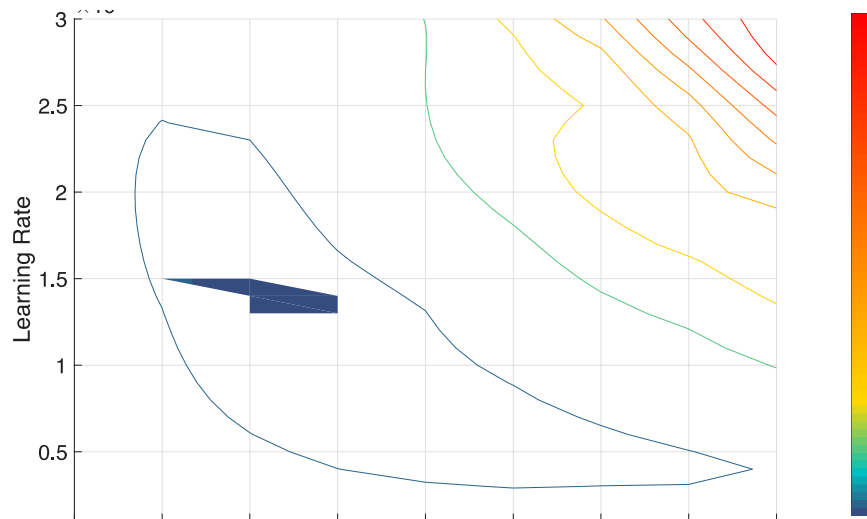


Figure 48 Color Map of Hebbian Learning (decay rate = 0)

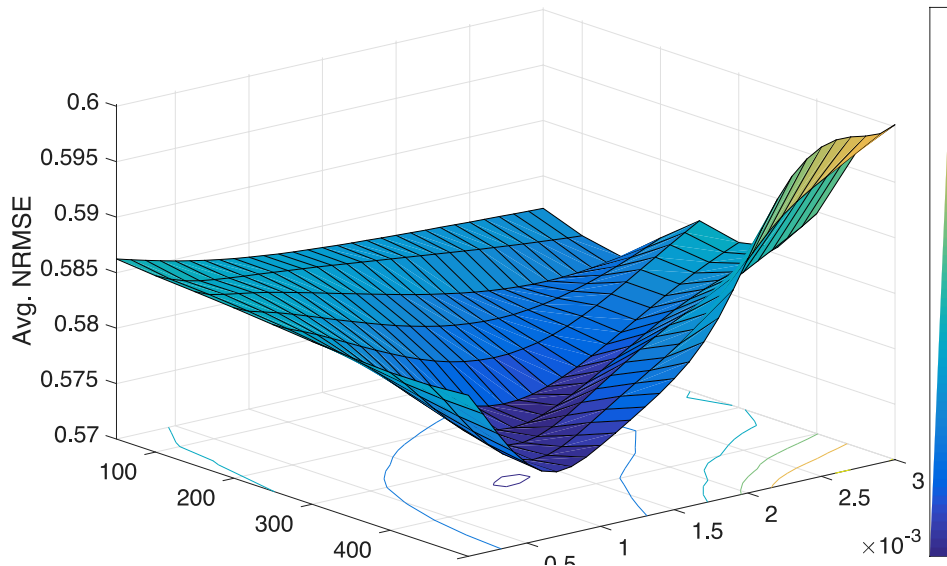


Figure 49 Learning Trend of Hebbian Learning (decay rate = 0)

BCM learning also has 3 tunable parameters, learning rate, τ and learning iterations. Based on different values of τ , we cut the result matrix into 15 parts. It can be seen from Figure 50 that changing the value of τ has a significant effect on the learning performance of BCM. We point out that the optimal point was reached when τ took on value 260. Consequently, the detailed results for $\tau = 260$ are organized and shown in Figure 51 and Figure 52.

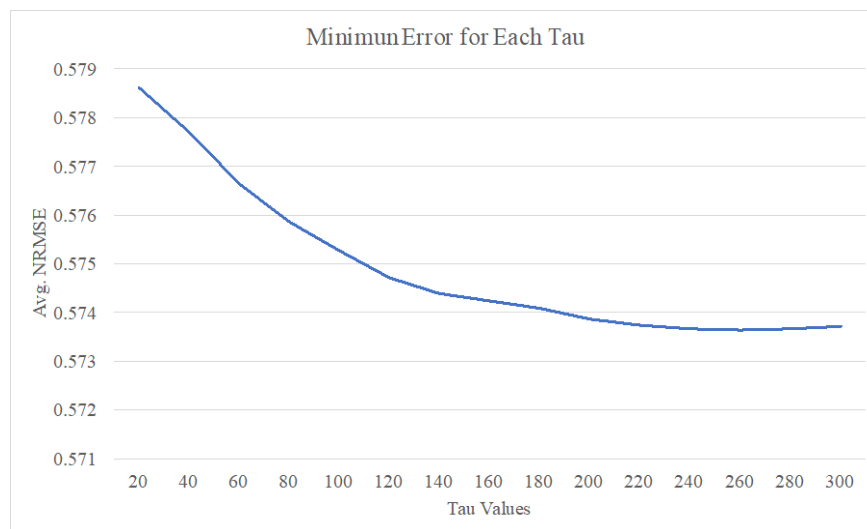


Figure 50 BCM's Best Case for Each Tau

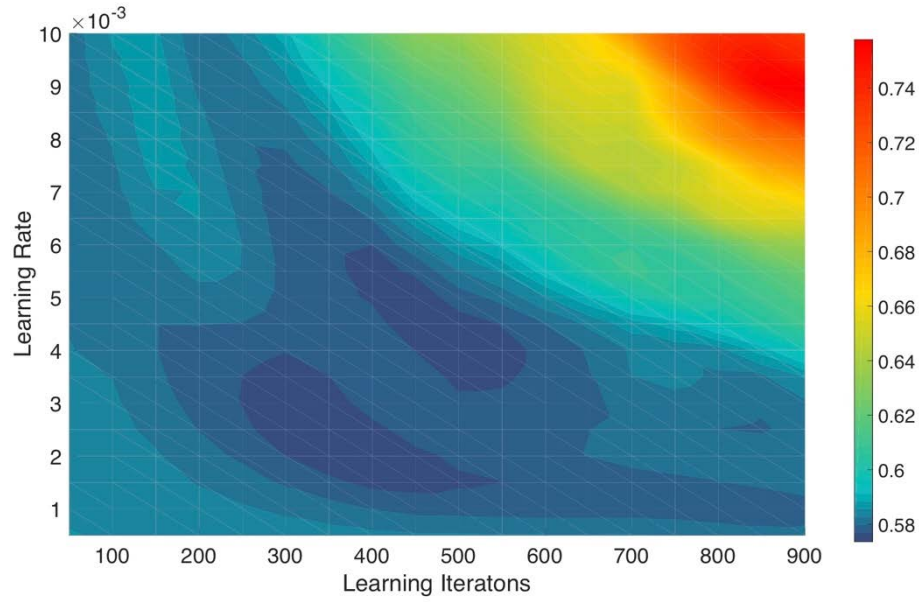


Figure 51 Color Map of BCM (tau = 260)

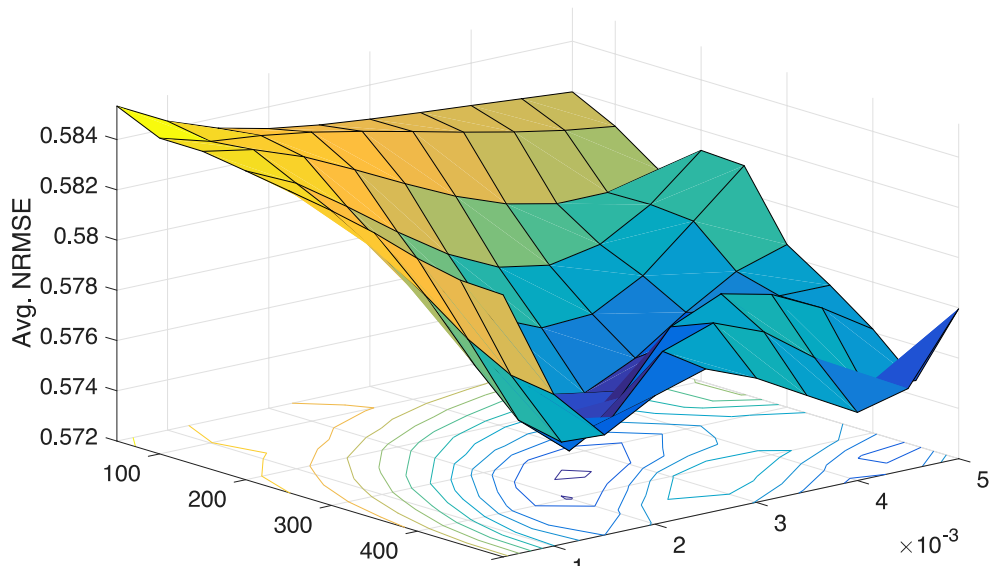


Figure 52 Learning Trend of BCM (tau = 260)

The best average output when applying BCM learning is 0.57364, the corresponding output errors of output1 and output2 are 0.55700 and 0.59028. This optimum is obtained at a learning rate of 0.0025 and a training iteration of 350.

To compare the capabilities of all synaptic learning rules above, the lowest output errors for each rule are concluded in Table 25. Note that for Hebbian learning with decay, the best performance was obtained at the smallest decay rate implemented (i.e. 0.00002).

Table 25 The Best Performance for Each Rule

Learning Rules	Output 1	Output 2	Output Avg.
No Plasticity	0.564200751	0.609061584	0.586631168
Hebb's Rule	0.554328315	0.595228256	0.574778285
Hebb's Rule with Decay	0.554194701	0.595378040	0.574786371
Oja's Rule	0.563482968	0.602993386	0.583238177
BCM	0.556995932	0.590276516	0.573636224

The best improvements are calculated by subtracting the lowest NRMSE achieved from the value before synaptic learning. Figure 53 shows the comparison on improvements in histograms, from which we can clearly see that except for the improvement on output1, BCM outperforms the other two learning rules and Anti-Oja gives the worst results.

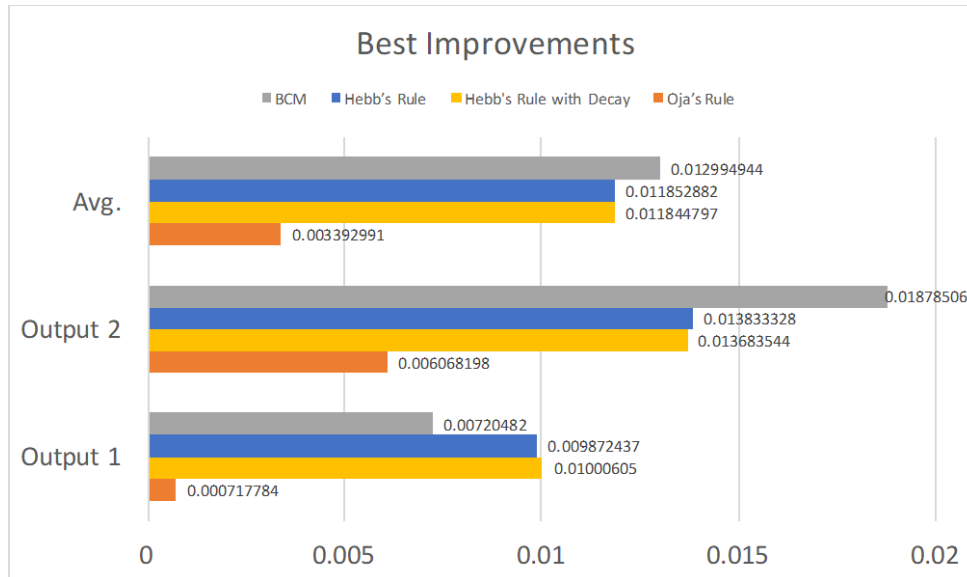


Figure 53 The Best Improvements of All Learning Rules

4.4.3 Interpretive Reservoir: A Preliminary Study on The Association Between Artificial Neural Network and Biological Neural Network

4.4.3.1 Experiment Setup

In this section, we used data from another project to simulate the brain activities in the form of ERP brainwaves based on a customized ESN and estimate the parameters using VAR model. ESN is usually created with fixed sparse internal connections. However, in our case, we don't want to limit the network structure to some specific topology. For this reason, we initialized the network with a fully connected reservoir.

In the past study, the subjects were shown 5 categories of image stimuli. Based on the EEG brain waves collected during the stimulation process, ERPs were calculated as the average of 20

EEG trials to provide a more reliable measurement. For each type of stimuli presented to each subject, 80 EEG trials were collected over all 30 channels. Among them, different combinations were randomly chosen and 10 ERPs (on 30 channels) were generated per person per stimulus. Based on the ERP clips calculated, we tried to model the brain activities in three scenarios. 1) Scenario I: we focused on the same subject stimulated by the same type of images. Consequently, 10 networks can be evolved based on 10 ERP episodes in each case. 2) Scenario II, distinct neural networks were established and compared based on the ERP data from different subjects with the same stimuli. 3) Scenario III: models were built upon a single subject under diverse stimuli. By involving the three scenarios, we expected that the different memories of people, rooted in their biological brain and captured by EEG or ERP signals, could be measured and reflected on the topology of artificial neural networks. Ultimately, correlation of networks' connections is going to be analyzed to compare between different models generated and to interpret the characteristics of memory representation in artificial neural networks.

4.4.3.2 Results

In consideration of all three scenarios, overall 10 network models were estimated for each subject with each type of stimulus. The correlation coefficients of input term \mathbf{c} and internal weights \mathbf{W} pairwise between different networks were then calculated to compare the similarities according to the needs of each scenario.

For Scenario I, where the models of a single individual who was given the same categories of stimuli are investigated, the results are carried out for each subject independently based on “C Food” stimuli. Figure 54 and Figure 55 present the average values of correlation coefficients computed over 10 networks on the input term and internal weight matrix. When concerning the same person, all models hold high correlations on the estimations of both the input term and the internal weights. This meets our expectation, because for a single individual, his/her brain structure is unlikely to be changed dramatically in the short term. Therefore, its simplified model, the network connections, should also be relatively stable. Moreover, because of being shown with images of the same kind, a subject should have certain areas or certain signal conduction pathways activated in the brain, so the input term which combines both the input signal and the input pathways should be steady in this case.

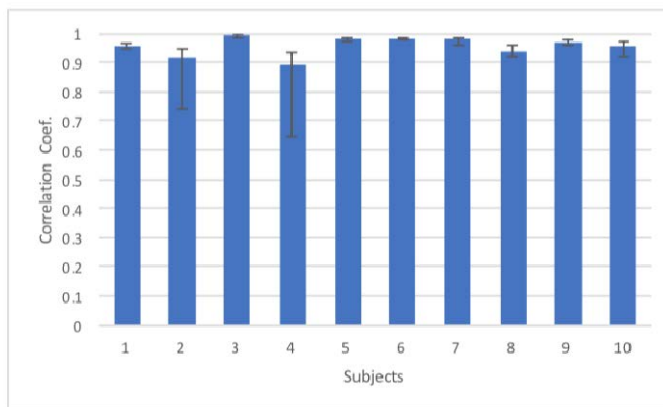


Figure 54 The Avg. Correlation Coefficients on Input Term (Scenario I)

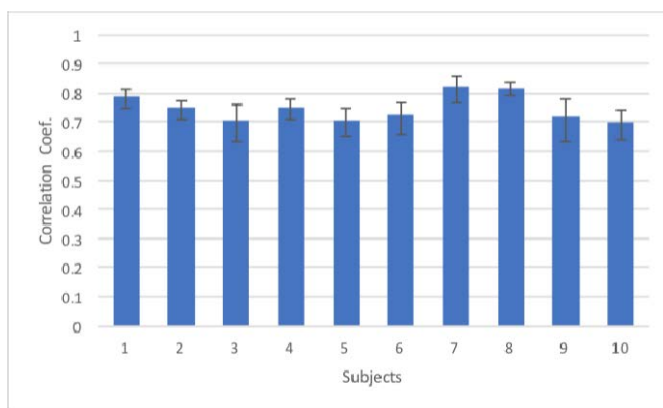


Figure 55 The Avg. Correlation Coefficients on Internal Weights (Scenario I)

For Scenario II, the average coefficients calculated pairwise between different subjects with stimuli “C Food” are provided in Figure 56 and Figure 57. Note that in this scenario, the value of each subject in the figures is provided based on the correlation coefficients between this subject and all others. It is noted that the models generated for different people are diverse from each other, no matter for the input part or for the internal configurations, which is consistent with our expectations. Despite broad similarities, everyone’s brain is unique. Influenced by our past experiences, the biological connections inside the brain (synapses) evolve slowly, shaping our unique memories. As a result, when given the same stimulus (for example, seeing the same picture), despite of some general common feelings we may have on it (for some specific food or people), people are excited with diverse activations and pathways.

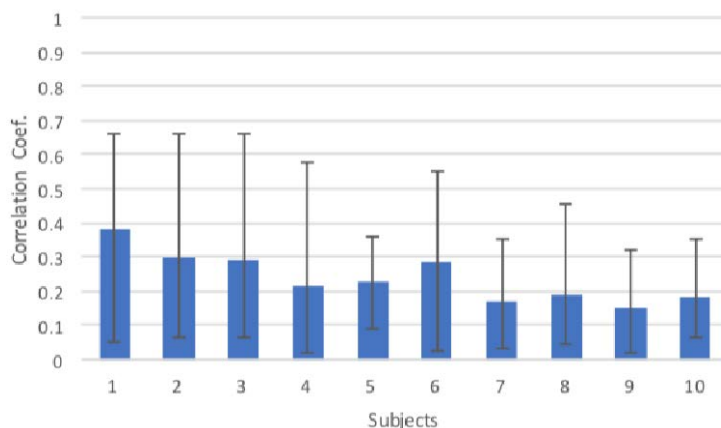


Figure 56 The Avg. Correlation Coefficients on Input Term (Scenario II)

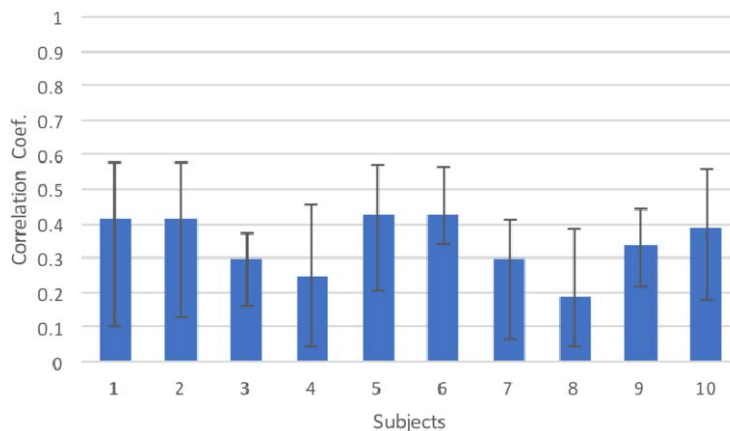


Figure 57 The Avg. Correlation Coefficients on Internal Weights (Scenario II)

We further drew the probability density functions (PDFs) of the correlation coefficients in Scenario I and II together. The left plot in Figure 58 shows the coefficients for the input term in both scenarios and the right in Figure 58 indicates the coefficients for the similarity of the internal weights. It is clearly shown that the correlation coefficient distribution of the same person's models differs significantly from the one across different subjects.

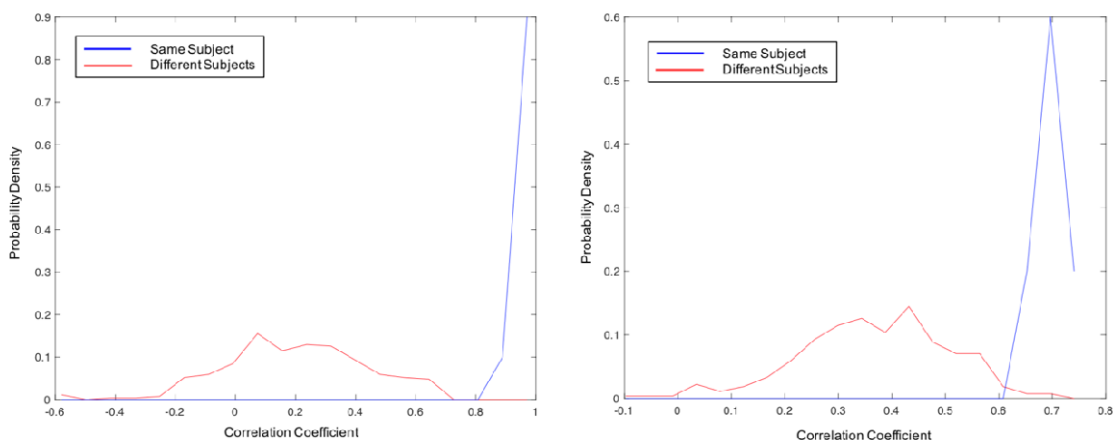


Figure 58 PDF of Correlation Coefficients for Input Term (left) and Internal Weights (right) (Scenario I & II)

As for Scenarios III, as shown in Figure 59 and Figure 60, the network models were compared between different categories of stimuli on the same subject, utilizing the data from all participants. We are pleased to find that even presented with different stimuli, the similarity of internal network connections maintains at the same level as in Scenario I. The variation of input stimuli doesn't affect the network models' internal structure much, while resulting in the slight drop of input correlations. This interesting finding verifies that neural network has similar characteristics with biological brain regarding the memory and topology: for one person, diverse stimuli only arouse his/her different reactions, but do not lead to great changes of internal connections within a short period of time.

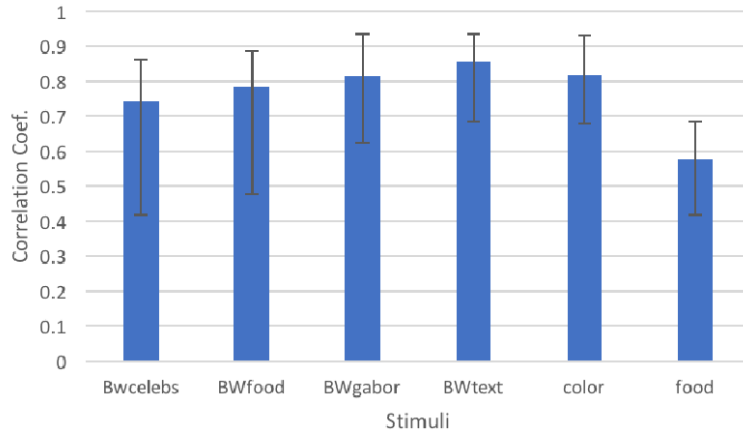


Figure 59 Avg. Correlation Coefficients for the Input Term (Scenario III)

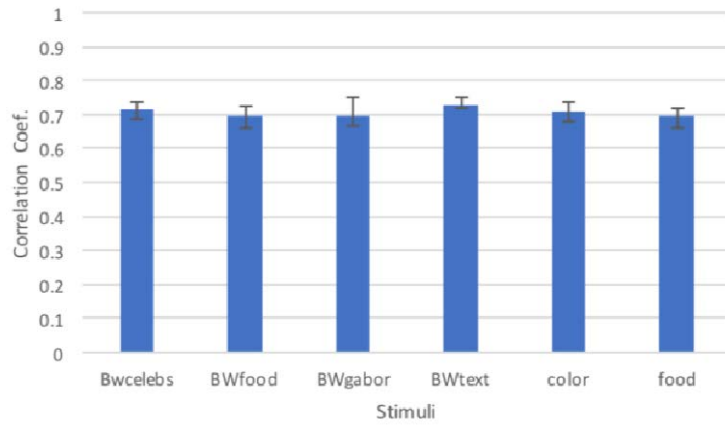


Figure 60 Avg. Correlation Coefficients for the Internal Weights (Scenario III)

5.0 CONCLUSIONS

In this study, we investigated several emerging, neuromorphic computing paradigms that may play key roles in supporting next-generation autonomy. In accordance with the planned tasks, several objectives have been achieved.

We first systematically created a general picture of cognitive architecture and processing flow of autonomous systems for real-time and adaptive problem solving, through an extensive literature survey. Then, we focused on the autonomous target tracking problem utilizing multiple computational intelligence approaches including ANNs, deep learning and reservoir computing. Accordingly, the performance metrics of those solutions were assessed, in terms of accuracy, speed, and energy consumption. Reservoir computing was further investigated and optimized with the bio-inspired concepts of dynamic ensembles of reservoir networks and neuronal plasticity. Based on this, evolutionary and adaption methods for reservoir networks are proposed from structural and synaptic level. Finally, some frontal issues related to biological cognition and intelligence were discussed through the exploration of human category learning inspired classification network and the association between ANNs and biological neural networks.

6.0 REFERENCES

- [1] J. Hawkins, "What intelligent machines need to learn from the neocortex," *IEEE Spectrum*, **54**(6), 34–71, 2017.
- [2] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, **2**, 1019-1025, 1999.
- [3] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. Int'l Symp. Circuits and Systems (ISCAS)*, 253-256, 2010.
- [4] T. Serre and T. Poggio, "A neuromorphic approach to computer vision," *Comm. ACM*, **53**(10), 54-61, 2010.
- [5] J. Yin and Y. Meng, "Reservoir computing ensembles for multi-object behavior recognition," in *Proc. IEEE Int'l Joint Conference on Neural Networks (IJCNN)*, 1-8, 2012.
- [6] S. Haeusler and W. Maass, "A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models," *Cerebral Cortex*, **17**(1), 149-162, 2007.
- [7] A. M. Thomson, D. C. West, Y. Wang, and A. P. Bannister, "Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2-5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro," *Cerebral Cortex*, **12**(9), 936-953, 2002.
- [8] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communications," *Science*, **304**(5667), 78-80, 2004.
- [9] Seger, Carol A., and Earl K. Miller, "Category learning in the brain," *Annual review of neuroscience*, **33**, 203-219, 2010.
- [10] Ashby, F. Gregory, and W. Todd Maddox, "Human category learning," *Annu. Rev. Psychol*, **56**, 149-178, 2005.
- [11] Ashby, F. Gregory, and Shawn W. Ell, "The neurobiology of human category learning," *Trends in cognitive sciences*, **5.5**, 204-210, 2001
- [12] Kéri, Szabolcs. "The cognitive neuroscience of category learning." *Brain Research Reviews*, **43.1**, 85-109, 2003.
- [13] R. Kicingier, T. Arciszewski, and K. De Jong, "Evolutionary computation and structural design: a survey of the state-of-the-art," *Computers and Structures*, **83**(23-24): 1943-1978, 2005.
- [14] J. Zhang, Z.-H. Zhan, Y. Lin, N. Chen, Y.-J. Gong, J.-H. Zhong, H. S. H. Chung, Y. Li, and Y.-H. Shi, "Evolutionary computation meets machine learning: a survey," *IEEE Computational Intelligence Magazine*, **6**(4): 68-75, 2011.
- [15] Francisco J. Varela, and Paul Bourgine, "Toward a practice of autonomous systems," *Proceedings of the First European Conference on Artificial Life*, MIT Press, 1992.
- [16] Esther Thelen, and Linda B. Smith, "A dynamic systems approach to the development of cognition and action," MIT press, 1996.
- [17] JA Scott Kelso and Armin Fuchs, "Self-organizing dynamics of the human brain: Critical instabilities and Šil'nikov chaos," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **5.1**, 64-69, 1995.
- [18] JA Scott Kelso, "Dynamic patterns: The self-organization of brain and behavior." MIT press, 1997.
- [19] David Vernon, Giorgio Metta, and Giulio Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE Transactions on Evolutionary Computation*, **11.2**, 151, 2007.

- [20] Gregor Schöner and Esther Thelen, "Using dynamic field theory to rethink infant habituation," *Psychological review*, **113.2**, 273, 2006.
- [21] Allen Newell, "The knowledge level," *Artificial intelligence*, **18.1**, 87-127, 1982.
- [22] Pat Langley, John E. Laird, and Seth Rogers. "Cognitive architectures: Research issues and challenges." *Cognitive Systems Research* **10.2** (2009): 141-160.
- [23] Allen Newell, "*Unified theories of cognition*," Harvard University Press, 1994.
- [24] Laird, John E., Allen Newell, and Paul S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial intelligence*, **33.1**,1-64, 1987.
- [25] Lehman, Jill Fain, John E. Laird, and P. S. Rosenbloom, "A gentle introduction to Soar, an architecture for human cognition," *Invitation to cognitive science*, **4**,212-249, 1996.
- [26] Laird, John E., "*The Soar cognitive architecture*," MIT Press, 2012.
- [27] Hui-Qing Chong, Ah-Hwee Tan, and Gee-Wah Ng, "Integrated cognitive architectures: a survey," *Artificial Intelligence Review*, **28.2**,103-130, 2007.
- [28] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin, "An integrated theory of the mind," *Psychological review*, **111.4**,1036, 2004.
- [29] Giacomo Rizzolatti, Leonardo Fogassi, and Vittorio Gallese, "Parietal cortex: from sight to action," *Current opinion in neurobiology*, **7.4**, 562-567, 1997.
- [30] Giacomo Rizzolatti, et al. "The space around us." *Science* **277**.5323 (1997): 190-191.
- [31] Rodney A. Brooks, "Visual map making for a mobile robot," *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, **2**, 1985.
- [32] Rodney A. Brooks, "Cambrian intelligence: the early history of the new AI", Vol. 1. Cambridge, MA: MIT press, 1999.
- [33] Rodney A. Brooks, "Intelligence without representation," *Artificial intelligence*, **47.1**, 139-159, 1991.
- [34] Ron Sun, Paul Slusarz, and Chris Terry, "The interaction of the explicit and the implicit in skill learning: a dual-process approach," *Psychological review*, **112.1**, 159, 2005.
- [35] Mantas Lukoševičius and Herbert Jaeger, *Overview of reservoir recipes*, No. 11. Technical report, 2007.
- [36] Mantas Lukoševičius and Herbert Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, **3.3**,127-149, 2009.
- [37] Hrushikesh N. Mhaskar, and Charles A. Micchelli, "Dimension-independent bounds on the degree of approximation by neural networks," *IBM Journal of Research and Development*, **38.3**, 277-284, 1994.
- [38] Herbert Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*, **5**. Bonn: GMD-Forschungszentrum Informationstechnik, 2002.
- [39] Mantas Lukoševičius, "A practical guide to applying echo state networks," *Neural networks: Tricks of the trade*, Springer, Berlin, Heidelberg, 659-686, 2012.
- [40] Büsing, Lars, Benjamin Schrauwen, and Robert Legenstein., "Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons," *Neural computation*, **22.5**, 1272-1311, 2010.
- [41] Ali Rodan, and Peter Tino, "Minimum complexity echo state network," *IEEE transactions on neural networks*, **22.1**, 131-144, 2001.
- [42] Danil Prokhorov, "Echo state networks: appeal and challenges," *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, **3**, 2005.

- [43] Benjamin Schrauwen, et al., "Improving reservoirs using intrinsic plasticity," *Neurocomputing*, **71.7-9**, 1159-1171, 2008.
- [44] Huttenlocher, Peter R., "Neural plasticity: The effects of environment on the development of the cerebral cortex", Harvard University Press, 2009.
- [45] Hughes, John R., "Post-tetanic potentiation," *Physiological reviews*, **38.1**, 91-113, 1958.
- [46] Hebb, Donald Olding, "The first stage of perception: growth of the assembly," *The Organization of Behavior*, Psychology Press, 102-120, 2005.
- [47] Elie L. Bienenstock, Leon N. Cooper, and Paul W. Munro, "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex," *Journal of Neuroscience*, **2.1**, 32-48, 1982.
- [48] Donald Olding Hebb, "The organization of behavior: A neuropsychological approach," John Wiley & Sons, 1949.
- [49] Erkki Oja, "Simplified neuron model as a principal component analyzer," *Journal of mathematical biology*, **15.3**, 267-273, 1982.
- [50] Steil, Jochen J., "Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning," *Neural Networks*, **20.3** 353-364, 2007.
- [51] Hiroaki Kitano, "Empirical Studies on the Speed of Convergence of Neural Network Training Using Genetic Algorithms," AAAI. 1990.
- [52] Hiroaki Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex systems*, **4.4**, 461-476, 1990.
- [53] A. Souahlia, A. Belatreche, A. Benyettou, and K. Curra, "An experimental evaluation of echo state network for colour image segmentation," *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016.
- [54] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks-with an erratum note," Bonn, "Germany: German National Research Center for Information Technology", *GMD Technical Report*, **148(34)**, 13, 2001.
- [55] W. Maass, T. Natschlager, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, **14(11)**, 2531–2560, 2002.
- [56] Boudjelal Meftah, Olivier Lézoray, and Abdelkader Benyettou, "Novel approach using echo state networks for microscopic cellular image segmentation," *Cognitive Computation*, **8.2**, 237-245, 2016.
- [57] Guihua Wen, Huihui Li, and Danyang Li, "An ensemble convolutional echo state networks for facial expression recognition," *International Conference on Affective Computing and Intelligent Interaction (ACII)*, IEEE, 2015.
- [58] Fabian Triefenbach, et al., "Acoustic modeling with hierarchical reservoirs," *IEEE Transactions on Audio, Speech, and Language Processing*, **21.11**, 2439-2450, 2013.
- [59] Plöger, Paul G., et al., "Echo state networks for mobile robot modeling and control," *Robot Soccer World Cup*, Springer, Berlin, Heidelberg, 2003.
- [60] Yanbo Xue, Le Yang, and Simon Haykin, "Decoupled echo state networks with lateral inhibition," *Neural Networks*, **20.3**, 365-376, 2007.
- [61] Jürgen Schmidhuber, et al., "Training recurrent networks by evolino.," *Neural computation* **19.3**, 757-779, 2007.
- [62] Daan Wierstra, Faustino J. Gomez, and Jürgen Schmidhuber, "Modeling systems with internal state using evolino," *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, ACM, 2005.

- [63] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. Hudspeth, "Principles of neural science," **4**, McGraw-hill New York, 2000.
- [64] Fabian Triefenbach, et al. "Phoneme recognition with large hierarchical reservoirs." *Advances in neural information processing systems*. 2010.
- [65] J. H. Holland, "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence," MIT press, 1992.
- [66] NNDC Climate Data Online, <https://www7.ncdc.noaa.gov/CDO/cdo>, Accessed Dec. 1, 2016.
- [67] H. Jaeger, "Simple and very simple matlab toolbox for echo state networks," <http://reservoir-computing.org/software>, 2009. Accessed Dec. 1, 2016.
- [68] Reilly, Douglas L., Leon N. Cooper, and Charles Elbaum, "A neural model for category learning," *Biological cybernetics*, **45.1**, 35-41, 1982.
- [69] Kruschke, John K. "ALCOVE: an exemplar-based connectionist model of category learning," *Psychological review*, **99.1**, 22, 1992.
- [70] Love, Bradley C., "Comparing supervised and unsupervised category learning," *Psychonomic bulletin & review*, **9.4**, 829-835, 2002.
- [71] Kurtz, Kenneth J., "The divergent autoencoder (DIVA) model of category learning." *Psychonomic Bulletin & Review*, **14.4**, 2007, 560-576.
- [72] Kurtz, Kenneth J., "The divergent autoencoder (DIVA) account of human category learning," *Proceedings of the Annual Meeting of the Cognitive Science Society*, **27(27)**, 2005.
- [73] Sebastián Basterrech and Gerardo Rubino, "Echo State Queueing Network: a new reservoir computing learning tool," *IEEE Consumer Communications and Networking Conference (CCNC)*, IEEE, 2013.
- [74] Robert S. Zucker and Wade G. Regehr, "Short-term synaptic plasticity," *Annual review of physiology*, **64.1**, 355-405, 2002.
- [75] Mohd-Hanif Yusoff, Joseph Chrol-Cannon, and Yaochu Jin, "Modeling neural plasticity in echo state networks for classification and regression," *Information Sciences*, **364**, 184-196, 2016.
- [76] Wei Wang, Hsiao-Tien Fan, and Zhanpeng Jin, "Structure optimization of dynamic reservoir ensemble using genetic algorithm," *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017.
- [77] Hsiao-Tien Fan, Wei Wang, and Zhanpeng Jin, "Performance optimization of echo state networks through principal neuron reinforcement," *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017.
- [78] Intrator N, Cooper L N, "Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions," *Neural Networks*, **5(1)**, 3-17, 1992.
- [79] Nathan Intrator, and Leon N. Cooper, "Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions," *Neural Networks*, **5.1**, 3-17, 1992.
- [80] R. C. Paolicelli, G. Bolasco, F. Pagani, L. Maggi, M. Scianni, P. Panzanelli, M. Giustetto, T. A. Ferreira, E. Guiducci, L. Dumas, et al., "Synaptic pruning by microglia is necessary for normal brain development," *Science*, **333(6048)**, 1456–1458, 2011.
- [81] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *Computing Research Repository (CoRR)*, arXiv: 1702.08608, 2017.
- [82] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & explorable approximations of black box models," *Computing Research Repository (CoRR)*, arXiv: 1707.01154, 2017.

- [83] Z. C. Lipton, “The mythos of model interpretability,” *Computing Research Repository (CoRR)*, arXiv: 1606.03490, 2017.
- [84] C. Molnar, “Interpretable Machine Learning: A Guide for Making Black Box Models Explainable,” <https://christophm.github.io/interpretable-ml-book/>, Accessed 2018.
- [85] M. F. Glasser, T. S. Coalson, E. C. Robinson, C. D. Hacker, J. Harwell, E. Yacoub, K. Ugurbil, J. Andersson, C. F. Beckmann, M. Jenkinson, S. M. Smith, and D. C. V. Essen, “A multi-modal parcellation of human cerebral cortex,” *Nature*, **536**, 171–178, 2016.
- [86] T. Horikawa, M. Tamaki, Y. Miyawaki, and Y. Kamitani, “Neural decoding of visual imagery during sleep,” *Science*, **340**(6132), 639–642, 2013.
- [87] M. Lynch, “Long-term potentiation and memory,” *Physiological reviews*, **84**(1), 87–136, 2004.
- [88] D. Purves and J. W. Lichtman, “Elimination of synapses in the developing nervous system,” *Science*, **210**(4466), 153–157, 1980.
- [89] M. V. R. Blondet, S. Laszlo, and Z. Jin, “Assessment of permanence of non-volitional eeg brainwaves as a biometric,” In *IEEE Int’l Conf. Identity, Security and Behavior Analysis (ISBA)*, pages 1–6, 2015.
- [90] M. V. Ruiz-Blondet, Z. Jin, and S. Laszlo, “Cerebre: A novel method for very high accuracy event-related potential biometric identification,” *IEEE Trans. Information Forensics and Security*, **11**(7), 1618–1629, 2016.
- [91] M. V. Ruiz-Blondet, Z. Jin, and S. Laszlo, “Permanence of the cerebre brain biometric protocol,” *Pattern Recognition Letters*, 2017.
- [92] H. Lutkepohl, “New introduction to multiple time series analysis,” *Springer Science & Business Media*, 2005.

7.0 LIST OF ACRONYMS

Acronym	Expansion
ACS	Action-Centered Subsystem
ACT-R	Adaptive Control of Thought – Rational
AE	Auto Encoder
ANN	Artificial Neural Network
BCM	Bienenstock, Cooper and Munro Rules
CEREBRE	Cognitive Event RELATED Biometric REcognition
BOLD	Blood Oxygen Level Dependent
CNN	Convolutional Neural Network
DESN	Decoupled Echo State Networks
DEWP	Daily Mean Dew Point
DIVA	Divergent Auto-Encoder
DL	Deep Learning
DLPFC	Dorsolateral Prefrontal Cortex
DNN	Deep Neural Network
DoD	Department of Defense
DRE	Dynamic Reservoir Ensemble
EEG	Electroencephalography
EOG	Electrooculogram
ERP	Event Related Potential
ESN	Echo State Network
fMRI	functional Magnetic Resonance Imaging
GA	Genetic Algorithm
GRN	Gene Regulatory Network
GRN-SO-RC	GRN-Self-Organizing Reservoir Computing
LSM	Liquid State Machine
LTM	Log Term Memory
LTP-IE	Long Term Potential of Intrinsic Excitability
MCS	Metacognitive Subsystem
MS	Motivational Subsystem
MSO	Multiple Superimposed Oscillator
NACS	Non Action-Centered Subsystem
NARMA	Non-linear Auto Regressive Moving Average
NN	Neural Network
NRMSE	Normalized Root Mean Square Error
PNR	Principle Neuron Reinforcement
PSA	Particle Swarm Algorithm
RC	Reservoir Computing
RCT	Reservoir Computing Tracker
SAR	Search and Rescue
SDAE	Stacked Denoising Autoencoder
SRE	Static Reservoir Ensemble
STP	Daily Mean Station Pressure
UAV	Unmanned Aerial Vehicle
VAR	Vector Autoregressive

VLPFC
WDSP
WU

Ventrolateral Prefrontal Cortex
Daily Mean Wind Speed
Working Memory