mathematics

MDPI

Article

Optimizing the Neural Structure and Hyperparameters of Liquid State Machines Based on Evolutionary Membrane Algorithm

Chuang Liu *, Haojie Wang, Ning Liu and Zhonghu Yuan

School of Information Engineering, Shenyang University, Shenyang 110044, China; mayang7566@163.com (H.W.); n.liu.cn@outlook.com (N.L.); syyzh62@163.com (Z.Y.)
* Correspondence: chuang.liu@syu.edu.cn

Abstract: As one of the important artificial intelligence fields, brain-like computing attempts to give machines a higher intelligence level by studying and simulating the cognitive principles of the human brain. A spiking neural network (SNN) is one of the research directions of brain-like computing, characterized by better biogenesis and stronger computing power than the traditional neural network. A liquid state machine (LSM) is a neural computing model with a recurrent network structure based on SNN. In this paper, a learning algorithm based on an evolutionary membrane algorithm is proposed to optimize the neural structure and hyperparameters of an LSM. First, the object of the proposed algorithm is designed according to the neural structure and hyperparameters of the LSM. Second, the reaction rules of the proposed algorithm are employed to discover the best neural structure and hyperparameters of the LSM. Third, the membrane structure is that the skin membrane contains several elementary membranes to speed up the search of the proposed algorithm. In the simulation experiment, effectiveness verification is carried out on the MNIST and KTH datasets. In terms of the MNIST datasets, the best test results of the proposed algorithm with 500, 1000 and 2000 spiking neurons are 86.8%, 90.6% and 90.8%, respectively. The best test results of the proposed algorithm on KTH with 500, 1000 and 2000 spiking neurons are 82.9%, 85.3% and 86.3%, respectively. The simulation results show that the proposed algorithm has a more competitive advantage than other experimental algorithms.

Keywords: evolutionary membrane algorithm; liquid state machines; P systems; spiking neural network; supervised classification

MSC: 68T20; 68W50

Citation: Liu, C.; Wang, H.; Liu, N.; Yuan, Z. Optimizing the Neural Structure and Hyperparameters of Liquid State Machines Based on Evolutionary Membrane Algorithm. *Mathematics* **2022**, *10*, 1844. https://doi.org/10.3390/math10111844

Academic Editor: Stelios Papadakis

Received: 7 May 2022
Accepted: 24 May 2022
Published: 27 May 2022

1. Introduction

As a third-generation artificial neural network, a spiking neural network (SNN) is one of the important research fields in brain-like computing and artificial intelligence [1]. The neurons of SNN need to transmit spikes one after the other, and these spikes form a spike sequence. The spike sequence contains time information because the time interval between each spike is not fixed, which cannot be expressed by traditional artificial neural networks [2]. In addition, each neuron in the SNN only needs to perform calculations when it receives a spike, so the power consumption is lower and the calculation speed is faster, while the traditional artificial neural network needs to be calculated layer by layer, and the calculation amount is much larger [3]. The SNN has great potential to solve real-time and power consumption problems of traditional deep learning models, and it contains many advantages, such as higher biological rationality, computational efficiency, and more efficient computing models. However, the training efficiency of the existing SNN learning algorithm is low, and its effect is not as good as the deep learning model, resulting in the low practicability of the current SNN and hindering its further

development [4]. The traditional artificial neural network has been fully developed with the help of mathematical tools such as statistics and optimization, and the training effect is currently better than that of the SNN [5]. In recent years, the SNN has attracted the wide attention of many scholars because of its neuron model, which is very close to biology, as the information processing unit. Compared with the traditional artificial neural network based on analog signals for information exchange [6], the SNN uses the time coding of neuron firing to realize the representation and transmission of information, which can show the rich dynamic characteristics of real biological systems. The SNN has better biological plausibility and imitates the information processing methods of biological neural networks, and has achieved great success in the fields of pattern recognition, automatic control, machine learning, associative memory, and other aspects, which reflects their broad application prospects [7–10].

A liquid state machine is a highly biomimetic neural computational model based on the SNN [11]. The liquid layer in its network structure is formed by recursive connection of several neurons, which is called a reservoir. The process of computing the network structure is called Reservoir Computing. Since the reservoir endows the liquid state machine with real-time computing capabilities and rich dynamic characteristics, existing research results show that it performs particularly well in terms of computational speed and computational accuracy [12,13]. Therefore, the in-depth study of the dynamic properties of SNNs is of great significance for the computational power of liquid state machines.

The method of optimizing the reservoir of the liquid state machine has encountered failures from a theoretical level, so some scholars have begun to employ the stochastic method to simplify the reservoir. Rajan et al. analyzed how the state structure of the neurons affects the internal structure of the reservoir based on the input. Although the existing complex optimizations have achieved certain success, they are mostly used for theoretical analysis [14]. Wieland et al. analyzed the ring topology with an isolated neuron in the reservoir, and the neurons are not connected to each other. The result is that a forward network connected by dendrites and axons is sufficient to meet the accuracy requirements of the task [15]. Chen et al. addressed this gap in the field of artificial networks by analyzing networks trained to perform memory and pattern generation tasks and used the dynamical objects as anchors to study the effect of learning on their emergence [16]. However, the methodology for optimizing the reservoir of the liquid state machine is not successful, and new theories are urgently needed to construct some reservoir optimization methods suitable for the liquid state machine, such as spiking coding, network structure and hyperparameter learning. Existing learning algorithms are still inefficient, especially in solving practical problems. The simulation effect is not ideal, resulting in low practicability of liquid state machines. In order to solve the above problems more effectively, many scholars try to use evolutionary algorithms to optimize the liquid state machines because these methods can still find the optimal solution without prior knowledge. Currently, some evolutionary algorithms, such as genetic algorithms (GA) [17] and the covariance matrix adaptive evolution strategy (CMA-ES) [18], are utilized to adaptively learn the configuring network structure and hyperparameters of liquid state machines.

The liquid state machine has good self-learning ability, and evolutionary algorithms have adaptive abilities. An evolutionary algorithm is a robust method which can adapt to different problems in different environments and can obtain satisfactory and effective solutions in most cases [19]. These algorithms have been used for solving scheduling for program segments, social network contact tracing, strongly connected components in digraphs and many other problems [19–21]. Evolutionary algorithms give an encoding scheme to the entire parameter space of the optimization problem to be solved instead of directly dealing with the specific parameters of the problem. Instead of starting from a single initial point, they search from a set of initial points. The information of the objective function value is used in the search, and it is not necessary to use the derivative information of the objective function or the special knowledge related to the specific

problem. Evolutionary algorithms have a wide range of applications, a high degree of nonlinearity, easy modification and parallelism [20,21]. Therefore, the organic combination of these two algorithms is beneficial to the liquid state machine to automatically learn network weights, network structure and other hyperparameters according to the input data. More specifically, the hyperparameters of the liquid machine are encoded as the solutions of an evolutionary algorithm in a multidimensional solution space. These solutions are optimized using the evolutionary rules of evolutionary algorithms until optimal hyperparameters are found. It can be seen that the liquid state machine model based on evolutionary algorithm is more suitable for solving practical problems. Therefore, many scholars have conducted a lot of work based on the combination of the two algorithms mentioned above. Ju et al. proposed a genetic-algorithm-based approach to evolve liquid filters from minimal structures with no connections to optimized kernels with minimal numbers of synapses and high classification accuracy [22]. This approach helps to design optimal LSMs with reduced computational complexity. The results obtained using this genetic programming approach suggest that the evolutionary distribution of synaptic weights is similar in shape to that found in cortical circuits. Reynolds et al. employs liquid state machines and genetic algorithms to develop useful networks that can be deployed on neuromorphic hardware. Building on past work on reservoir computation and genetic algorithms, the authors demonstrated the power of combining these two techniques, and the advantages it can provide compared to manually tuning reservoirs for classification tasks [23]. Zhou et al. presented a surrogate-assisted evolutionary search method for optimizing the hyperparameters and neural architecture of LSM reservoirs using CMA-ES [18]. To reduce the search space, the architecture of the liquid state machine is encoded by connection probabilities together with hyperparameters in the spiking neuron model. To improve computational efficiency, a Gaussian process is adopted as a proxy for auxiliary CMA-ES. Tian et al. proposed a neural architecture search (NAS)-based framework to explore the architecture and parameter design space of dataset-oriented automatic liquid state machine models [24]. To handle the exponentially growing design space, the authors employ a three-step search for LSMs, including a multi-liquid architecture search with parallel and serial hierarchical topologies, variation in the number of neurons in each liquid and parameter searches such as the percent connectivity and the excitability ratio of neurons in each fluid. Li et al. proposed a multi-objective LSM/Network-on-Chip (NoC) architecture co-design framework that quickly and efficiently explores the design space of LSM/NoC to generate an optimal the architecture of liquid state machines with low latency on an NoC-based platform [25].

Although these models have been successfully applied to solve problems in scientific research and engineering practice and these experiences are very useful for improving the generalization performance of liquid state machines, there are still some shortcomings in the training and design phases [9,26]. To solve the above problems, this paper mainly explores an improved evolutionary membrane algorithm to optimize the network structure and the hyperparameters of the liquid state machine . Membrane systems provide a rich framework for solving optimization problems. In view of the good parallelism, distribution and non-determinism of membrane computing, traditional optimization algorithms can be combined with membrane computing, traditional optimization algorithms can be used locally and the global optimal solution can be gradually searched through the communication between membranes. Using the well-preserved solutions of the skin membrane to guide the evolution of the inner membrane population can accelerate the convergence of the population, and this strategy takes into account both the convergence and distribution of the population. More specifically, membrane systems consist of objects, reaction rules and membrane structures. During the computation of membrane systems, objects are evolved according to reaction rules, and the nested membrane structure can also be changed. Objects are stored in regions separated by membranes, and reaction rules employ local search algorithms to evolve these objects. Objects in different membranes can be exchanged by applying transport rules. The membrane algorithm based on the

mechanism of the membrane system has better advantages than other metaheuristic evolutionary algorithms, and is very suitable for solving complex optimization problems. It is these mechanisms that make membrane algorithms based on membrane systems ideal for solving complex problems. Based on our previous work [27,28], this paper aims to study the objects, rules and membrane structures of learning algorithms based on evolutionary membrane algorithms to further improve the generalization performance of liquid state machines. Then, the proposed learning algorithm is used to solve the standard image classification dataset MNIST, a handwritten character set, and Human Action Recognition using the KTH Dataset. Finally, the proposed algorithm is compared with state-of-the-art algorithms to verify its effectiveness. The experimental results show that the algorithm outperforms similar learning algorithms in classification accuracy.

The main contributions of this paper can be summarized as follows:

- There are few studies on evolutionary algorithms for optimizing the hyperparameters of liquid state machines.
- For the first time, an improved evolutionary membrane algorithm is used as a learning algorithm for a liquid state machine.
- According to the hyperparameters of the liquid state machine, three elements of an evolutionary membrane algorithm are implemented, including objects, reaction rules and membrane structures.
- In our experiments, we demonstrate that the results of the proposed algorithm are highly competitive with those of the experimental algorithms. Simulation results verify the effectiveness of the proposed algorithm as a learning algorithm for the hyperparameters of a liquid state machine.

The rest of this paper is summarized below. Section 2 discusses the concept of the liquid state machines and the evolutionary membrane algorithm. Section 3 describes the proposed algorithm in detail, especially how to design objects, reaction rules and membrane structures to optimize the neural structure and hyperparameters of liquid state machines. In Section 4, the proposed algorithm is compared with the state-of-the-art algorithms, and the simulation results are evaluated and discussed. Finally, Section 5 summarizes the conclusions of this paper.

2. Theoretical Background

This section presents the theoretical knowledge of this work, including liquid state machines and evolutionary membrane algorithms.

2.1. Liquid State Machine

Liquid state machines have been proposed to simulate the complex process by which neurons process dynamic information [13]. In the biological nervous system, the internal activity of neurons is dynamic. To some extent, a stable state can only be a "dead" state. This has aroused the research interest of scholars. Scholars will find a model that does not depend on a steady state and can perform real-time computations as a research goal in the field of neurocomputing science [17]. More specifically, when analyzing the computational power of the human brain, scholars often focus on the processing of static input signals. Signals transmitted in reality have complex structures that vary over time, which is why neurons have the ability to process these signals in real time. However, processing dynamic and complex inputs in real-time in the real world is not an easy task, which has kept researchers in computer science and other fields striving in this direction. As a result, there is growing interest in the ability of the nervous system to dynamically process information.

Liquid state machines have the advantage of handling time-varying time series and outputting results in real time, which is related to its special network structure. The liquid state machine mainly consists of three parts, as shown in Figure 1, including the spike-encoding layer (input component), the liquid layer (usually a random sparsely connected recurrent neural network) and the readout layer (output component).

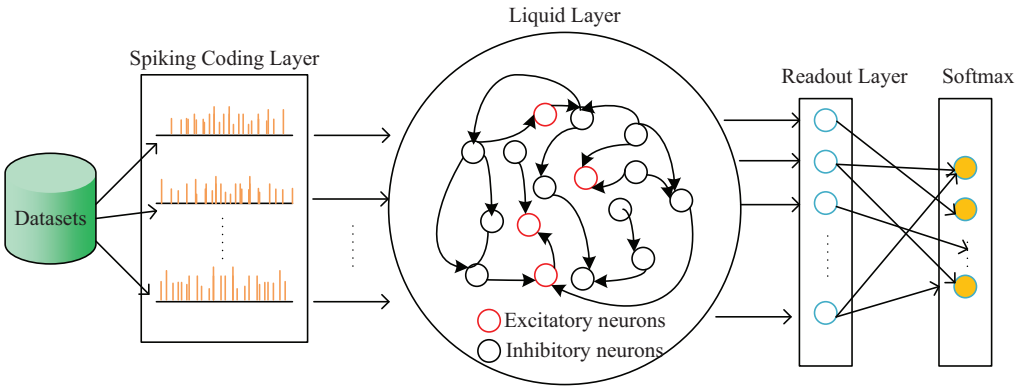

Figure 1. An example of the learning process of the spike neural network supervised learning algorithm.

Based on the liquid machine structure in Figure 1, we describe its working principle as follows:

The connection from the spiking coding layer to the liquid layer is determined randomly. Neurons in the spiking coding layer can be connected to all neurons in the liquid layer or to some neurons in the liquid layer. When neurons in the liquid layer receive input from the spiking coding layer, they respond to the input signal, and then use a filter function to convert the response of the liquid layer into a high-dimensional state, called the liquid state. It can also be said that the role of the liquid layer is to complete the mapping from the input layer to the output layer. The data are projected into a higher-dimensional space that is more distinguishable than the original space. Among them, the connection weights of the liquid layer and the output layer are trained for specific tasks, and finally, the liquid state is converted into the desired output value through the output neurons of the output layer. It is particularly worth noting that during the calculation, once the connection weights of the liquid layer are determined, they will not change; only the output weights will be trained. More specifically, suppose that the input of the liquid state machine is $u(t)$ in the spiking coding layer, which is a function of time. The output of the liquid state machine is $y(t)$, which consists of the readout layer and the softmax layer. The neural state of the liquid state machine is a vector $x^M(t)$ composed of the firing states of all neurons, and the role of the liquid layer is to map $u(t)$ to $x^M(t)$. It should be noted that assuming the current time is t, $x^M(t)$ is not only related to the current input $u(t)$ but also related to the input in $t' < t$, so the liquid layer has memory capabilities. The source of memory ability is that the neurons in the liquid layer are recursively connected. Through this recursive connection, the spiking input influence will continue to be transmitted in the liquid layer for a certain period of time before disappearing. Because the liquid layer has the ability to absorb time information, the output layer no longer needs to know the relevant spiking input before time t to be able to train the output. The output layer is f^M. Unlike the liquid layer, the output layer is memoryless. f^M maps $x^M(t)$ to $y(t)$ by a certain learning algorithm.

2.2. Evolutionary Membrane Algorithm

Inspired by the structure and dynamic activity of living cells, tissues or organs, the membrane computing as the abstracted distributed parallel computing models was first proposed by Paun. Figure 2 is a schematic diagram of a membrane computational model, which consists of objects, reaction rules and membrane structures. The specific model of membrane computing is also called the membrane system or the P system. The research on membrane computing mainly includes theoretical research and application research [29–31].

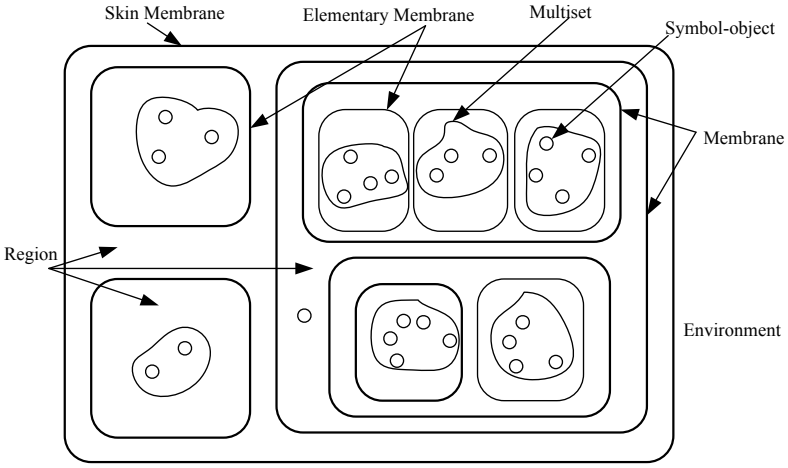

Figure 2. A membrane computational model.

Based on a membrane system, the membrane algorithm was first proposed to solve the traveling salesman problem by Nishida in 2006 [32]. Although the membrane algorithm has achieved many achievements since it was proposed, there are still many directions for improvement in its research. At present, there are many excellent membrane algorithms that can find excellent solutions within a certain period of time. Zhang et al. presented a hybrid approach based on the appropriate combination of the differential evolution algorithm and tissue P systems to solve a class of constrained manufacturing parameter optimization problems [33]. Niu et al. proposed a novel membrane algorithm based on ant colony optimization. The proposed algorithm adopts the membrane system of non-deterministic distributed parallel framework, and the sub-algorithm of the basic membrane is ant colony optimization [34]. Peng et al. proposed a multiobjective clustering framework to deal with fuzzy clustering problem by designing a tissue-like membrane system with a special cell structure [35]. Orozco-Rosas et al. proposed two path-planning algorithms based on membrane computing [36,37]. One of them is a hybrid algorithm based on a membrane pseudo-bacterial potential field, which uses a combination of the structure and rules of membrane computing. Another algorithm is a membrane evolutionary artificial potential field approach for solving the mobile robot path-planning problem. They achieve high performance, yielding competitive results for autonomous mobile robot navigation in complex and real scenarios. Song et al. proposed a new multi-membrane search algorithm (MSA) based on biological cell behavior [38]. The results show that the MSA has efficient convergence capabilities on unimodal functions and multimodal functions. Tian et al. proposed a new hybrid heuristic algorithm for solving a job shop scheduling problem inspired by the tissue-like membrane system [39]. Niu et al. proposed a membrane-inspired multi-objective algorithm (MIMOA) [40]. MIMOA is characterized by a parallel distributed framework with two operation subsystems and one control subsystem, respectively. Liu et al. proposed an improved membrane algorithm for solving multimodal multiobjective problems based on the framework of P system [41]. The simulation results show that compared with other experimental algorithms, the proposed algorithm has a competitive advantage in solving all 22 multimodal benchmark test problems in CEC2019. In addition, in the indirect application of the membrane algorithm, fruitful results have been achieved in the field of combinatorial optimization such as clustering [42,43], neural networks [28] and so on.

3. Proposed Framework

The algorithm proposed in this paper is used to optimize the network structure and hyperparameters of liquid state machines. According to the membrane structure of the membrane algorithm, it is divided into two stages, including the genetic algorithm of global exploration and the firework algorithm of local optimization. In the skin membrane, the genetic algorithm is used to search for the global optimal result, and the object is guided

to develop in the area where the optimal solution may exist but the samples are relatively sparse. In the elementary membrane, the fireworks algorithm is used as a local exploration in order to improve the robustness, approximation accuracy and construction efficiency of the candidate object. The specific steps of the proposed algorithm are discussed as follows.

Figure 3 introduces the training process of the liquid state machine based on the proposed algorithm framework.

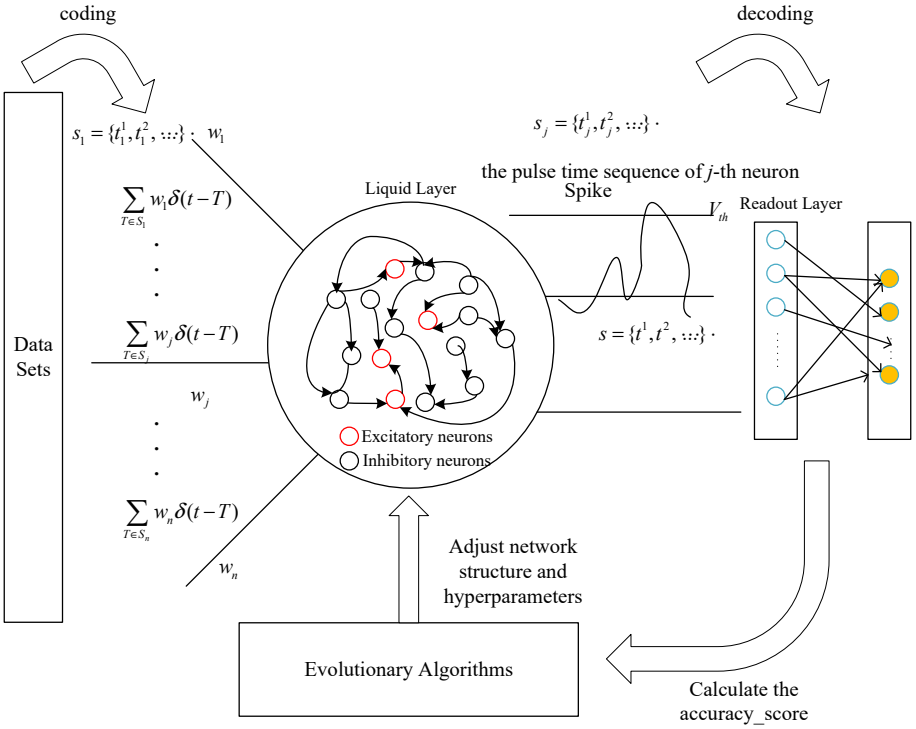

Figure 3. LSM design based on the evolutionary algorithm framework.

The procedure in Figure 3 can be explained as follows:

- The datasets need to be coded as the spiking input sequence of the liquid state machine;
- The hyperparameters and the neural structure of the liquid state machine are coded as the decision variables of the proposed algorithm;
- The optimum for the design of the liquid state machine is found by the constant searching of the proposed algorithm;
- The output spiking of the liquid state machine is loaded into the Readout layer, and the Readout layer will transfer the spiking sequence into the prediction results;
- According to the accuracy_score between the prediction results and the actual results, the proposed algorithm is called to generate new hyperparameters and the neural structure of the liquid state machine until the optimal prediction result is obtained.

Figure 4 shows the flowchart of the proposed algorithm.

The specific implementation details of the proposed algorithm will be further described and explained in the following section.

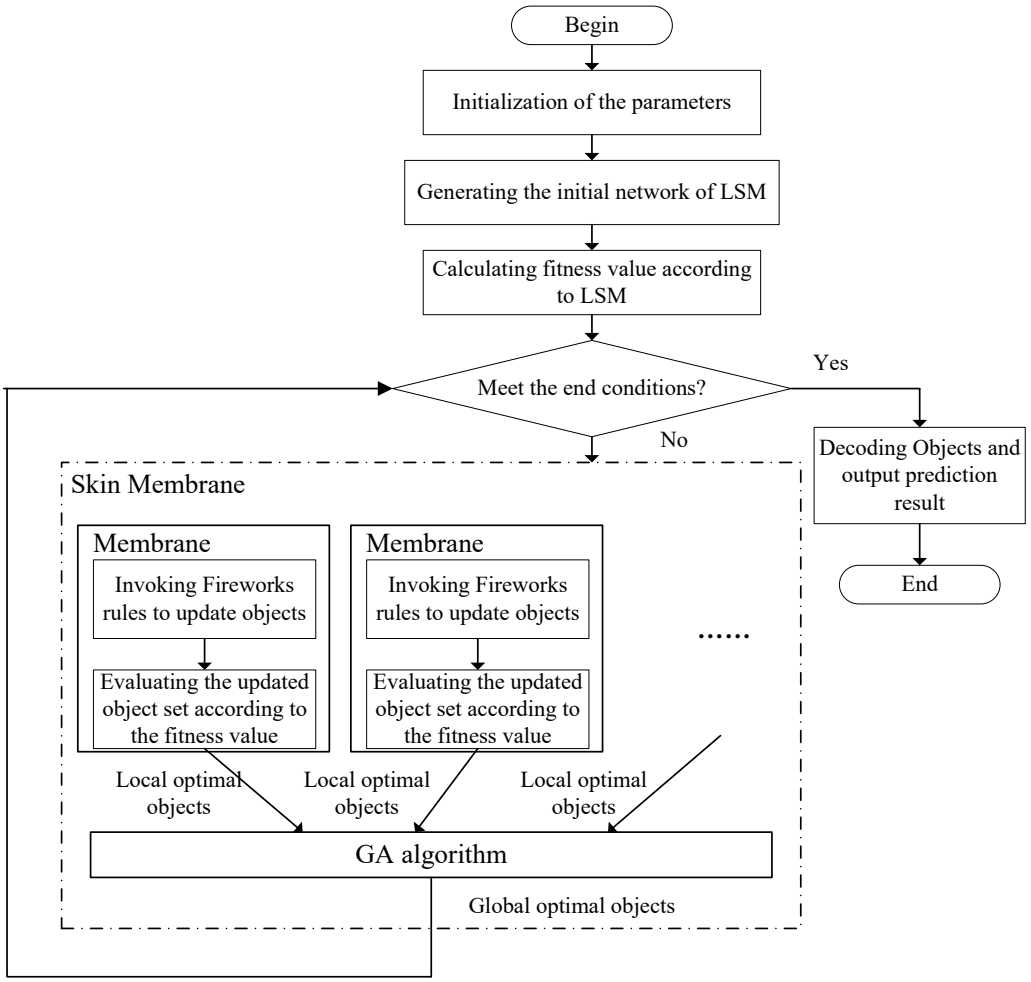

Figure 4. The flowchart of the proposed algorithm.

3.1. Membrane Structure

In the proposed algorithm, a structure is chosen in the outer membrane, named the skin membrane, that includes several elementary membranes. This structure represents a logical division of the search space, which facilitates parallel searches. However, in this paper, the structure is implemented in a serial fashion. The membrane region may have several objects, which are the decision variables of the hyperparameters of the liquid state machine. The object is similar to the chemical substance in the membrane: it can swim freely in the region of the membrane. In the proposed algorithm, the reaction rule of the membrane simulates the swimming process of the object. Therefore, the swimming process of the object is understood as finding the optimal solution. According to the literature [18], a total of nine hyperparameters are selected to encode the architecture and parameters of LSM. These parameters include λ, P_{en-R}, A_{en-R}, τ_{n-E}, τ_{n-I}, A_{EE}, A_{EI}, A_{IE} and A_{II}. The problem of setting hyperparameters is transformed into an optimization problem. More specifically, an object contains these nine hyperparameter dimensions. Before the object is used, it needs to be initialized according to Equation (1):

$$\begin{aligned} o_{i,j} &= o^l + (o^u - o^l) \times r \\ 1 &\le i \le I \\ 1 &\le j \le D \end{aligned} \tag{1}$$

where I represents the total number of objects; D is the dimension of a decision variable; $o_{i,j}$ is the value of the j-th dimension in the i-th object; o^l is the lower boundary; o^u represents the upper boundary; and r denotes a random number between 0 and 1.

3.2. Reaction Rules

There are two reaction rules in the proposed algorithm. These reaction rules include a genetic algorithm and a fireworks algorithm. In the skin membrane, the genetic algorithm [44] is employed to find the global objects when all the objects from the membrane are evolved. The fireworks algorithm [45] is then introduced to evolve the objects in the region of the membrane. The pseudocode of the fireworks algorithm is described in Algorithm 1. The specific implementation form of the fireworks algorithm is referenced in [45]. The proposed algorithm uses the explosion radius and the explosion sparks of the fireworks algorithm to improve the balance of exploitation and exploration for the candidate objects.

Algorithm 1 Pseudocode for the calling logic of fireworks algorithm.

```
Randomly select n locations, each position corresponds to a membrane region with objects
m is a constant used to adjust the amount of sparks produced by the explosion
while Meet the end condition? do
    Set off n objects at the respective n membranes
    for i = 1 : n do
        Calculate the number of sparks that the object yields:ŝ_i
        Obtain locations of ŝ_i sparks of the object in the current membrane
    end for
    for k = 1 : m do
        Randomly select an object
        Generate a specific spark for the object
    end for
    Select the best location and keep it for next explosion generation
    Randomly select n − 1 locations from the two types of sparks and the current objects
end while
return O
```

At first, a certain number of objects need to be initialized in the feasible region Ω, and the fitness value of these objects is evaluated. In order to distinguish objects from different locations, objects with better fitness values can obtain more resources and generate more sparks in a smaller area, and have a strong local search capability for the location of the object. On the contrary, objects with poor fitness values can only acquire relatively few resources, generate fewer sparks in a larger area and have a certain global search capability.

In order to achieve the purpose of object differentiation, the explosion radius of each object and the number of sparks produced by the explosion are calculated based on their fitness value relative to the other objects in the membrane region. For an object o_i, the calculation formula for the explosion radius A_i is shown in Equation (2):

$$A_i = \hat{A} \times \frac{f(o_i) - f_{min} + \varepsilon}{\sum_{i=1}^{N}(f(o_i) - f_{min}) + \varepsilon} \tag{2}$$

where $f_{min} = min(f(o_i)), i = 1, 2, \cdots, N$ is the minimum fitness in the current membrane; $\hat{A}$ is a constant, used to adjust the radius of the explosion; and ε is the minimum amount for a machine to avoid division by zero.

The number of explosion sparks S_i is calculated according to Equation (3):

$$S_i = M \times \frac{f_{max} - f(o_i) + \varepsilon}{\sum_{i=1}^{N}(f_{max} - f(o_i)) + \varepsilon} \tag{3}$$

where $f_{max} = max(f(o_i)), i = 1, 2, \cdots, N$ is the maximum fitness in the current membrane. M is a constant used to adjust the number of sparks produced by the explosion. ε is the minimum amount of a machine to avoid division by zero.

In order to increase the diversity of the object, the proposed algorithm introduces a mutation operator to generate a mutation object. The process of generating the Gaussian mutation object is described as follows: first, an object o_i is randomly selected from the membrane, and then a certain number of dimensions of the object are randomly selected to perform the Gaussian mutation operation. Performing Gaussian mutation operation on the selected dimension k of object o_i is shown in Equation (4):

$$\hat{o_{ik}} = o_{ik} \times e \tag{4}$$

where e $N(1,1)$, $N(1,1)$ represents a Gaussian distribution with a mean of 1 and a variance of 1.

4. Experimental Studies

Brian2, which is an open source, highly flexible and dynamically scalable simulation tool, is employed to simulate LSM, which is a spike neural network implemented with Python3. MNIST and KTH are selected as the benchmark datasets to analyze the performance of the proposed algorithm and provide insights. First, we describe the benchmark function and evaluation indicator used in the experiment. Second, we give a brief comparison of the latest experimental algorithms and provide the experimental setup used in this study. Finally, the comparison results of experimental algorithms under MNIST are discussed.

4.1. Benchmark Datasets and Evaluation Indicators

This section presents two benchmark datasets, an evaluation metric and experimental conditions. These benchmark datasets consist of the MNIST and KTH datasets.

4.1.1. Benchmark Datasets

The MNIST dataset comes from the National Institute of Standards and Technology (NIST). The training set is composed of 250 handwritten numbers from different people, of which 50% are high school students and 50% are from the population Census Bureau (the Census Bureau) staff. The test set is also the same proportion of handwritten digital data. The MNIST handwritten character set is a standard image classification dataset, containing 70,000 gray-scale images of handwritten digits, of which 60,000 are the training dataset and 10,000 are the test set; the size of each image is 28 × 28. As shown in Figure 5, the image classification dataset is composed of 0 to 9. We obtained the MNIST dataset from http://yann.lecun.com/exdb/mnist/ (accessed on 17 April 2022). The dataset contains four files, namelym training set images, training set labels, test set images and test set labels. The details of these files are described below:

1. The compressed file name of the training set images is train-images-idx3-ubyte.gz, the file size is 9.9 MB and the file contains 60,000 samples.
2. The compressed file name of the training set labels is train-labels-idx1-ubyte.gz, the file size is 29 KB and the file contains 60,000 labels.
3. The compressed file name of the test set images is t10k-images-idx3-ubyte.gz, the file size is 1.6 MB and the file contains 10,000 samples.
4. The compressed file name of the test set labels is t10k-labels-idx1-ubyte.gz, the file size is 5 KB and the file contains 10,000 labels.

From MINIST, we randomly select 3000 and 500 as the number of training data and test data, respectively. Each datum here is a 28 × 28 matrix. In addition, we randomly select another 500 samples from the training data as validation data. To find the best network prediction model, the training dataset is employed to train the hyperparameters of liquid state machines by the experimental algorithms. The validation data are used to evaluate the performance of the liquid state machine during the optimization process. To validate the prediction model obtained by the experimental algorithm, the test dataset is used to demonstrate the performance of the experimental algorithms in providing the best solution,

which serves as the final accuracy score of the model. Here, the 28 × 28 matrix is converted into a 784 × 1 vector as the input of the liquid state machine.

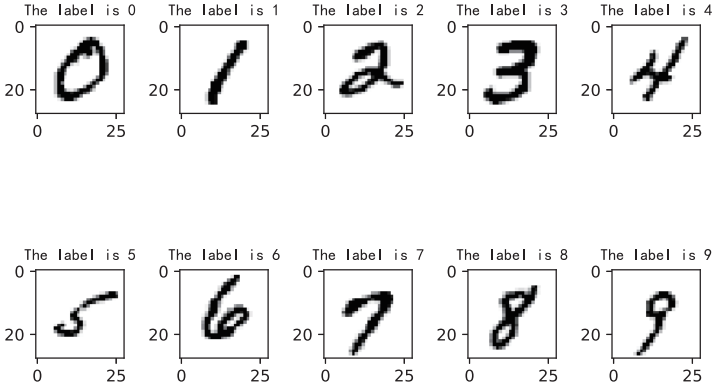

Figure 5. An example of some 0–9 numbers of MNIST parsed by Python.

The release of the KTH dataset in 2004 was a milestone in the field of computer vision. The KTH dataset includes a total of 2391 video samples of 6 types of actions (walking, jogging, running, boxing, hand waving and hand clapping) completed by 25 people in 4 different scenarios, which was the largest human action dataset recorded at that time. It is possible to systematically evaluate the performance of different algorithms on the input data. The video samples in the dataset include scale changes, clothing changes and lighting changes, but the background is relatively simple and the camera is fixed. All sequences are stored in AVI file format and are available online (https://www.csc.kth.se/cvap/actions/ (accessed on accessed on 17 April 2022)). Each combination has $25 \times 6 \times 4 = 600$ video files with 25 themes, 6 actions and 4 scenes. Each file contains approximately four subsequences that were used as sequences in our experiments. Each file is subdivided into sequences by start_frame and end_frame, and a list of all sequences is in "00sequences.txt".

The KTH dataset is divided into a training set (8 people), a validation set (8 people) and a test set (9 people) with respect to subjects. LSM is trained on the training set, while the validation set is used to evaluate the performance of LSM. The recognition results of LSM are obtained on the test set.

4.1.2. Evaluation Indicators

The accuracy_score is employed to evaluate the performance of all experimental algorithms. In multi-label classification, this indicator is used as the proportion of the predicted results that exactly match the corresponding actual results. Its expression is shown in Equation (5):

$$accuracy\_score = \frac{m}{L} \tag{5}$$

where m is the number of predicted results that exactly match the corresponding actual results. L is the number of real target classification results.

4.1.3. Experimental Conditions

Four experimental algorithms, including BO [46], GA [44], CMA-ES [47] and GP-assisted CMA-ES [18], are employed to evaluate the performance of the proposed algorithm. The solving performances of all experimental algorithms are calculated by the accuracy_score.

All experiments were run on Windows 10 Pro with a Dual Intel Xeon Platinum 8160 processor (33 M cache, 2.10 GHz), 160 GB of physical RAM and two Xeon CPUs containing 48 parallel cores and 96 threads. All experimental algorithms were implemented in Pycharm community with Python 3.10.

All comparison algorithms were initialized according to the parameters given in the literature. The proposed algorithm was initialized according to Equation (6):

$$\prod = \begin{cases} \{o_{i,j} | 1 \le i \le 10, 1 \le j \le 5\}, \\ [_0[_1]_1, [_2]_2, \cdots, [_5]_5]_0, \\ \{O \to \bigcup_i^5 o_j, o_{i,j} \to o_{i,j}^{'}, \\ o_{i,j} \to [o_i]_j, [o_i]_j \to o_{i,j}, \\ []_0 \to [_0[_1]_1, [_2]_2, \cdots, [_5]_5]_0, \} \end{cases} \tag{6}$$

where $\prod$ denotes a membrane system; $[_0[_1]_1, [_2]_2, \cdots, [_5]_5]_0$ represents that the skin membrane contains five elementary membranes; $o_{i,j}$ indicates the i-th object in the region of the j-th membrane; and $O \to \bigcup_i^5 o_j$ consists of all objects from the different membranes. The rule of $o_{i,j} \to o_{i,j}^{'}$ is the reaction rules by executing the fireworks algorithm and the genetic algorithms. $[]_0 \to [_0[_1]_1, [_2]_2, \cdots, [_5]_5]_0$ represents a change in the membrane structure from a skin membrane to a structure comprising multiple elementary membranes.

4.2. Comparing the Results of All Experimental Algorithms on MINIST

The performance of the proposed algorithm is evaluated by the accuracy_score value on MINIST and compared with four experimental algorithms.

4.2.1. Comparing Results with 500 Spiking Neurons on MINIST

These neurons include 400 excitatory neurons and 100 inhibitory neurons. Each experimental algorithm can achieve good results in the training phase. In the validation and testing stages, some experimental algorithms perform well, but the proposed algorithm achieves better results on the training datasets, the validation datasets, and the test datasets. In Table 1, the training results of all experimental algorithms have a correct rate of more than 90%. The proposed algorithm obtains the best training and testing results. However, the GA achieves the best results in the validation dataset.

Table 1. Simulated results of all experimental algorithms with 500 spiking neurons on MINIST.

Accuracy_Score	BO [46]	GA [44]	CMA-ES [47]	GP-Assisted CMA-ES [18]	Proposed Algorithm
Train	91.2%	90.2%	90.8%	91.5%	92.2%
Validation	85.1%	84.4%	84.6%	85.2%	
Test	86.2%	81.2%	85.2%	85.4%	86.8%

4.2.2. Comparing Results with 1000 Spiking Neurons on MINIST

The above results may be related to the use of a small number of spiking neurons, and the following will increase the number of spiking neurons in the liquid state machine to 1000, of which the number of excitatory neurons is 800 and the number of inhibitory neurons is 200.

Table 2 shows the simulated results of four state-of-the-art algorithms and the proposed algorithm. We can see from the results in Table 2 that the simulation results of all experimental algorithms can be improved when the number of spiking neurons in the liquid state machines is increased. On the training datasets and testing datasets, the proposed algorithm can still achieve better results than other experimental algorithms. However, on the validation datasets, the GP-assisted CMA-ES attained the best accuracy_score.

Table 2. Simulated results of all experimental algorithms with 1000 spiking neurons on MINIST.

Accuracy_Score	BO [46]	GA [44]	CMA-ES [47]	GP-Assisted CMA-ES [18]	Proposed Algorithm
Train	96.5%	97.3%	91.6%	96.5%	98.6%
Validation	87.1%	87.6%	82.1%	89.4%	89.6%
Test	89.6%	89.8%	88.2%	90.4%	90.6%

4.2.3. Comparing Results with 2000 Spiking Neurons on MINIST

To verify the advantages of the proposed algorithm, we increase the number of spiking neurons of liquid state machines to 2000, and these neurons consist of 1600 excitatory neurons and 400 inhibitory neurons. We conduct experiments and compare the quantitative results with other experimental algorithms.

Table 3 gives the comparison results of all experimental algorithms on the MINIST datasets with 2000 spiking neurons. It can be seen that the proposed algorithm outperforms other algorithms on the training datasets. We can easily see that the GP-assisted CMA-ES algorithm outperforms BO, GA and CMA-ES. CMA-ES also achieves good results on the test datasets. Compared to other algorithms, BO has the worst results on all datasets.

Table 3. Simulated results of all experimental algorithms with 2000 spiking neurons on MINIST.

Accuracy_Score	BO [46]	GA [44]	CMA-ES [47]	GP-Assisted CMA-ES [18]	Proposed Algorithm
Train	98.9%	99.8%	99.5%	99.2%	99.9%
Validation	88.1%	88.8%	88.2%	89.1%	88.6%
Test	89.6%	90.6%	91.4%	89.6%	90.8%

4.3. Comparing the Results of All Experimental Algorithms on KTH

In order to further verify the effectiveness of the proposed algorithm, the following experiments select the prediction results of all experimental algorithms on the KTH dataset.

4.3.1. Comparing Results with 500 Spiking Neurons on KTH

The simulated results of all experimental algorithms are attained on the KTH datasets with 500 neurons. These neurons consist of 400 excitatory neurons and 100 inhibitory neurons. As can be seen from Table 4, the results of all experimental algorithms except CMA-ES on the training set are similar. The proposed algorithm outperforms these experimental algorithms on datasets of different stages.

Table 4. Simulated results of all experimental algorithms with 500 spiking neurons on KTH.

Accuracy_Score	BO [46]	GA [44]	CMA-ES [47]	GP-Assisted CMA-ES [18]	Proposed Algorithm
Train	89.8%	90.1%	86.3%	91.1%	91.7%
Validation	81.9%	82.1%	78.9%	84.4%	84.7%
Test	80.1%	80.4%	77.4%	82.5%	82.9%

4.3.2. Comparing Results with 1000 Spiking Neurons on KTH

In order to further compare the differences between these experimental algorithms in optimizing the structure and hyperparameters of the liquid state machine, the number of spiking neurons in the liquid state machine is increased to 1000. The number of excitatory neurons is 800, and the number of inhibitory neurons is 200. Table 5 shows the experimental results of four state-of-the-art algorithms and the proposed algorithm. In Table 5, we can see from the results that the experimental results of these algorithms has been improved. On the different datasets, the proposed algorithm can still achieve the best results in comparison with the other experimental algorithms.

Table 5. Simulated results of all experimental algorithms with 1000 spiking neurons on KTH.

Accuracy_SCORE	BO [46]	GA [44]	CMA-ES [47]	GP-Assisted CMA-ES [18]	Proposed Algorithm
Train	93.8%	94.1%	91.8%	96.3%	97.1%
Validation	84.0%	84.3%	82.4%	86.3%	86.6%
Test	83.3%	83.5%	81.7%	84.9%	85.3%

4.3.3. Comparing Results with 2000 Spiking Neurons on KTH

In this experimental section, the number of spiking neurons in liquid state machines include 1600 excitatory neurons and 400 inhibitory neurons.

Table 6 gives the quantitative results with other experimental algorithms. It can be seen that the proposed algorithm still outperforms other experimental algorithms on the different datasets.

Table 6. Simulated results of all experimental algorithms with 2000 spiking neurons on KTH.

Accuracy_Score	BO [46]	GA [44]	CMA-ES [47]	GP-Assisted CMA-ES [18]	Proposed Algorithm
Train	96.7%	96.9%	95.3%	98.1%	98.8%
Validation	85.8%	86.0%	84.2%	87.2%	87.5%
Test	85.4%	85.6%	83.6%	85.9%	86.3%

4.4. Discussion

In order to verify the advantages of the proposed algorithm in optimizing the liquid state machine, we designed two experiments on the MINIST and KTH datasets. The neurons of the liquid state machine have three values of 500, 1000 and 2000 in these two experiments. In the above experimental scenarios, four experimental algorithms are compared with the proposed algorithm. Based on the above experimental results, we can think that the proposed algorithm outperforms these experimental algorithms on datasets of different stages, and the membrane structure and reaction rules can help the proposed algorithm to improve the accuracy_score results. We can easily find that the proposed algorithm has a better accuracy_score. On the MNIST dataset, it can be seen from the results in Tables 1–3 that the proposed algorithm has a high performance in the training dataset and validation. Part of the results achieved good results in the test dataset. However, on the liquid state machine model with a large number of spiking neurons, the prediction accuracy of the proposed algorithm is not significantly better than that of the comparison algorithm. In other words, the advantages of the proposed algorithm are more pronounced on liquid state machines with a small number of spiking neurons. To show the convergence speed of the proposed algorithm, Figure 6 presents the convergence curves of the proposed algorithm with 500, 1000 and 2000 neurons for solving the MINIST dataset. The proposed algorithm with different numbers of neurons can converge in around 100 generations. On the training dataset, the proposed algorithm with a larger number of neurons has a better value for the accuracy_score. On the validation and test datasets, the proposed algorithm achieves similar results with different numbers of neurons, especially 1000 and 2000 neurons. The proposed algorithm with 500 neurons is worse than the algorithms with 1000 and 2000 neurons.

On the KTH dataset, from the results in Tables 4–6, the proposed algorithm can achieve the best results in the different stages. Figure 7 presents the convergence curves of the proposed algorithm with 500, 1000 and 2000 neurons for solving the KTH dataset. The proposed algorithm with different numbers of neurons can converge in around 80 generations. On the training dataset, the proposed algorithm with 500 neurons has a bad accuracy_score. This experimental result shows that the number of neurons is too small, and the liquid state machine is prone to overfitting. On validation and test datasets, the proposed algorithm achieves similar results with different numbers of neurons, especially 1000 and 2000 neurons. The proposed algorithm with 500 neurons yields slightly worse results compared to 1000 and 2000 neurons. Overall, liquid state machines with more neurons obtain better results.

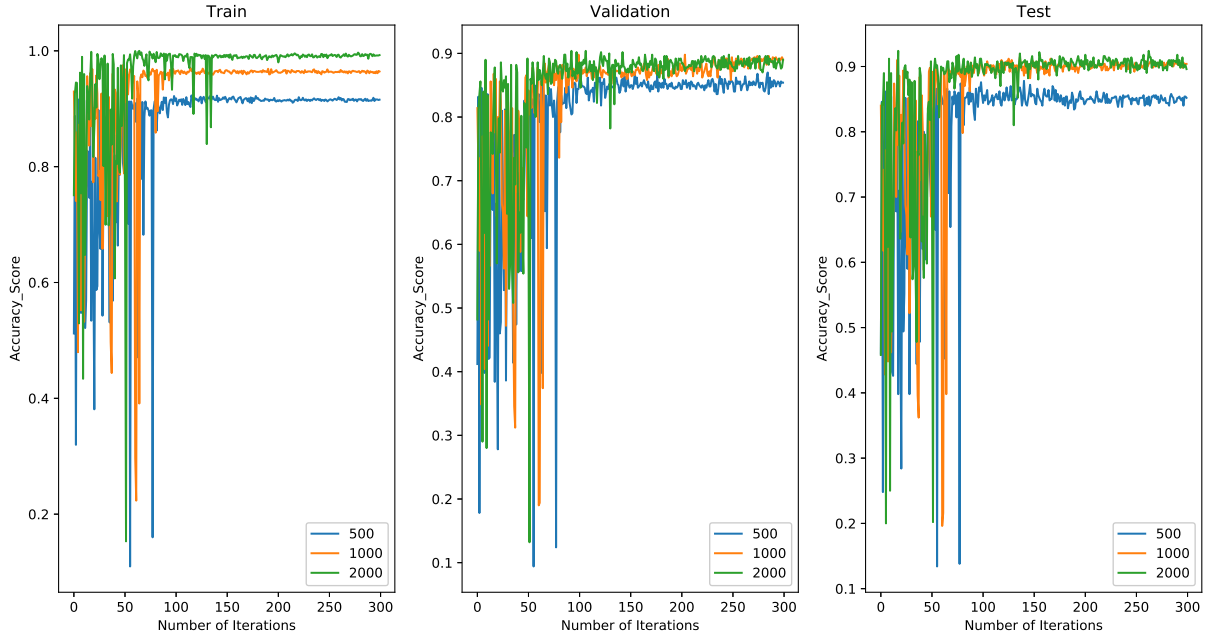

Figure 6. Convergence curve of the proposed algorithm for solving MINIST in training, validation and testing.

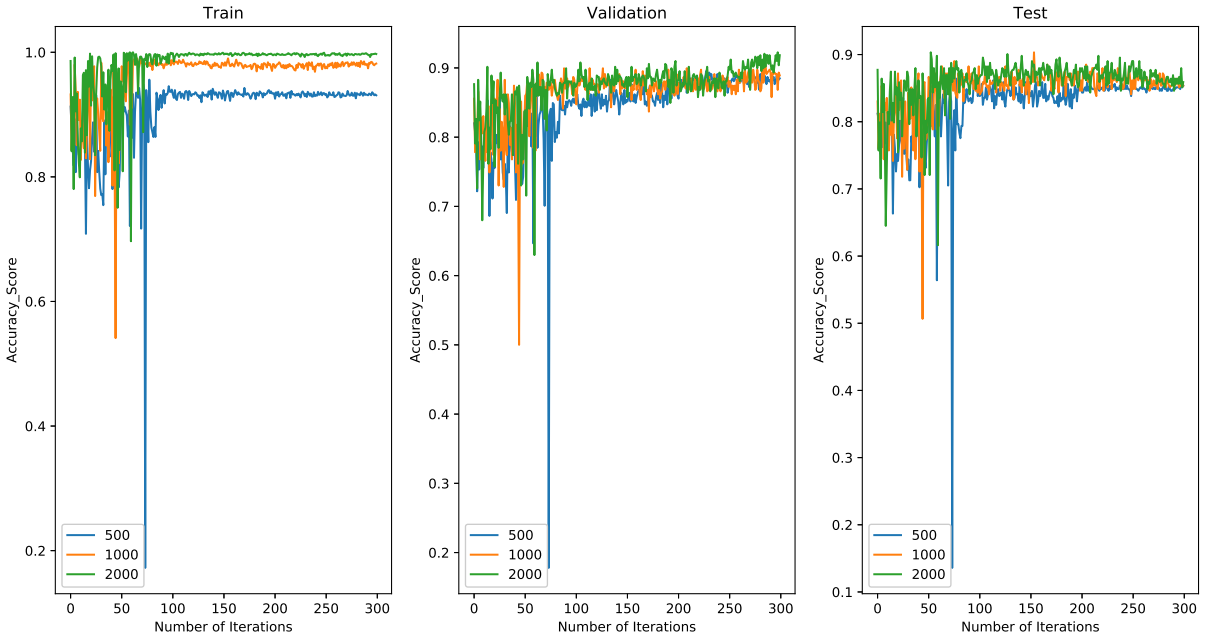

Figure 7. Convergence curve of the proposed algorithm for solving KTH in training, validation and testing.

It is precisely because these mechanisms can balance the relationship between exploration and exploitation well that they can help the proposed algorithm to approach the global optimum and avoid local optimum and finally find the optimal hyperparameters of the liquid state machine. Therefore, the proposed algorithm effectively utilizes objects, reaction rules and membrane structures and is effective for optimizing the network structure and hyperparameters of liquid state machines.

5. Conclusions and Future Work

This paper demonstrates that using the evolutionary membrane algorithm to optimize the network structure and hyperparameters of the liquid state machine model improves

its prediction accuracy more effectively than other experimental algorithms setting these network hyperparameters. When these hyperparameters of the liquid state machine model are changed using the proposed algorithm, the classification ability of the liquid state machine is improved. More specifically, on the MNIST datasets, the best test results of the proposed algorithm with 500, 1000 and 2000 spiking neurons are 86.8%, 90.6% and 90.8%, respectively. The best test results of the proposed algorithm on the KTH dataset with 500, 1000 and 2000 spiking neurons are 82.9%, 85.3% and 86.3%, respectively.

From the simulation results, it can be seen that the improved evolutionary membrane algorithm is effective in optimizing the network structure and hyperparameters of liquid state machines. The proposed algorithm outperforms four state-of-the-art experimental algorithms in varying numbers of spiking neurons of liquid state machines. The results obtained by the proposed algorithm are related to the object, reaction rule and membrane structure of the evolutionary membrane algorithm. These mechanisms of the evolutionary membrane algorithm can achieve a balance of exploration and exploitation, which helps the proposed algorithm to jump out of the local optimal hyperparameters of the liquid state machine and then find the best hyperparameters for the liquid state machine. From this, we can conclude that the application of the evolutionary membrane algorithm can improve the prediction accuracy of the liquid state machine. The proposed algorithm is not sensitive to the changes in various parameters and is suitable for scientific research and engineering practice.

Before the proposed model can be successfully applied to various tasks, there are still many problems to be solved, including the improvement in the evolutionary membrane algorithm, the encoding mechanism of the parameters of the liquid machine, and the design of its output layer, etc. The mentioned problem can be solved using various metaheuristic approaches such as chemical reaction optimization [20,48], harmony search [49], grey wolf optimization [21] and others.

Author Contributions: Conceptualization, C.L. and Z.Y.; methodology, C.L. and H.W.; software, H.W.; validation, C.L., H.W. and Z.Y.; writing—original draft preparation, C.L.; writing—review and editing, N.L. and Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by 69 batches of general funding projects from the China Postdoctoral Science Foundation, China (Grant No. 2021M693858), the Technological Innovation Program for Young People of Shenyang City, China (Grant No. RC210400), and the Scientific Research Funding Project of the Education Department of Liaoning Province, China (Grant No. 2020JYT05).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The MNIST dataset used for the evaluation is available at http://yann.lecun.com/exdb/mnist accessed on 17 April 2022, and the KTH dataset is available at http://www.nada.kth.se/cvap/actions accessed on 17 April 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, Y.; Mai, Y.; Feng, R.; Xiao, J. An adaptive threshold mechanism for accurate and efficient deep spiking convolutional neural networks. *Neurocomputing* **2022**, *469*, 189–197. [CrossRef]
2. Doborjeh, M.; Doborjeh, Z.; Merkin, A.; Bahrami, H.; Sumich, A.; Krishnamurthi, R.; Medvedev, O.N.; Crook-Rumsey, M.; Morgan, C.; Kirk, I.; et al. Personalised predictive modelling with brain-inspired spiking neural networks of longitudinal MRI neuroimaging data and the case study of dementia. *Neural Netw.* **2021**, *144*, 522–539. [CrossRef] [PubMed]
3. Jamshidi, M.B.; Lalbakhsh, A.; Talla, J.; Peroutka, Z.; Roshani, S.; Matousek, V.; Roshani, S.; Mirmozafari, M.; Malek, Z.; Spada, L.L.; et al. Deep learning techniques and covid-19 drug discovery: Fundamentals, state-of-the-art and future directions. In *Emerging Technologies during the Era of COVID-19 Pandemic*; Springer: Cham, Switzerland, 2021; pp. 9–31.
4. Petro, B.; Kasabov, N.; Kiss, R.M. Selection and Optimization of Temporal Spike Encoding Methods for Spiking Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 358–370. [CrossRef] [PubMed]

5. Khalaj, O.; Jamshidi, M.B.; Saebnoori, E.; Mašek, B.; Štadler, C.; Svoboda, J. Hybrid Machine Learning Techniques and Computational Mechanics: Estimating the Dynamic Behavior of Oxide Precipitation Hardened Steel. *IEEE Access* **2021**, *9*, 156930–156946. [CrossRef]
6. Jamshidi, M.B.; Talla, J.; Peroutka, Z. Deep learning techniques for model reference adaptive control and identification of complex systems. In Proceedings of the 2020 19th International Conference on Mechatronics-Mechatronika (ME), Prague, Czech Republic, 2–4 December 2020; pp. 1–7. doi: doi: 10.1109/ME49197.2020.9286698. [CrossRef]
7. Zhang, A.; Niu, Y.; Gao, Y.; Wu, J.; Gao, Z. Second-order information bottleneck based spiking neural networks for sEMG recognition. *Inf. Sci.* **2022**, *585*, 543–558. [CrossRef]
8. Liu, J.; Lu, H.; Luo, Y.; Yang, S. Spiking neural network-based multi-task autonomous learning for mobile robots. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104362. [CrossRef]
9. Lobo, J.L.; Del Ser, J.; Bifet, A.; Kasabov, N. Spiking neural networks and online learning: An overview and perspectives. *Neural Netw.* **2020**, *121*, 88–100. [CrossRef]
10. Wang, J.; Hafidh, B.; Dong, H.; Saddik, A.E. Sitting Posture Recognition Using a Spiking Neural Network. *IEEE Sens. J.* **2021**, *21*, 1779–1786. [CrossRef]
11. Norton, D.; Ventura, D. Improving liquid state machines through iterative refinement of the reservoir. *Neurocomputing* **2010**, *73*, 2893–2904. [CrossRef]
12. Zhang, Y.; Li, P.; Jin, Y.; Choe, Y. A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2635–2649. [CrossRef]
13. Florescu, D.; Coca, D. Learning with Precise Spike Times: A New Decoding Algorithm for Liquid State Machines. *Neural Comput.* **2019**, *31*, 1825–1852. [CrossRef] [PubMed]
14. Rajan, K.; Abbott, L.F.; Sompolinsky, H. Stimulus-dependent suppression of chaos in recurrent neural networks. *Phys. Rev. E* **2010**, *82*, 011903. [CrossRef] [PubMed]
15. Wieland, S.; Bernardi, D.; Schwalger, T.; Lindner, B. Slow fluctuations in recurrent networks of spiking neurons. *Phys. Rev. E* **2015**, *92*, 040901. [CrossRef] [PubMed]
16. Beer, C.; Barak, O. One Step Back, Two Steps Forward: Interference and Learning in Recurrent Neural Networks. *Neural Comput.* **2019**, *31*, 1985–2003. [CrossRef]
17. Iranmehr, E.; Shouraki, S.B.; Faraji, M.M.; Bagheri, N.; Linares-Barranco, B. Bio-Inspired Evolutionary Model of Spiking Neural Networks in Ionic Liquid Space. *Front. Neurosci.* **2019**, *13*, 1085. [CrossRef]
18. Zhou, Y.; Jin, Y.; Ding, J. Surrogate-Assisted Evolutionary Search of Spiking Neural Architectures in Liquid State Machines. *Neurocomputing* **2020**, *406*, 12–23. [CrossRef]
19. Goel, L. An extensive review of computational intelligence-based optimization algorithms: Trends and applications. *Soft Comput.* **2020**, *24*, 16519–16549. [CrossRef]
20. Mahafzah, B.A.; Jabri, R.; Murad, O. Multithreaded scheduling for program segments based on chemical reaction optimizer. *Soft Comput.* **2021**, *25*, 2741–2766. [CrossRef]
21. Al-Shaikh, A.; Mahafzah, B.; Alshraideh, M. Metaheuristic approach using grey wolf optimizer for finding strongly connected components in digraphs. *J. Theor. Appl. Inf. Technol.* **2019**, *97*, 4439–4452.
22. Ju, H.; Xu, J.X.; Chong, E.; VanDongen, A.M. Effects of synaptic connectivity on liquid state machine performance. *Neural Netw.* **2013**, *38*, 39–51. [CrossRef]
23. Reynolds, J.J.M.; Plank, J.S.; Schuman, C.D. Intelligent Reservoir Generation for Liquid State Machines using Evolutionary Optimization. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
24. Tian, S.; Qu, L.; Wang, L.; Hu, K.; Li, N.; Xu, W. A neural architecture search based framework for liquid state machine design. *Neurocomputing* **2021**, *443*, 174–182. [CrossRef]
25. Li, S.; Tian, S.; Kang, Z.; Qu, L.; Wang, S.; Wang, L.; Xu, W. A multi-objective LSM/NoC architecture co-design framework. *J. Syst. Archit.* **2021**, *116*, 102154. [CrossRef]
26. Wang, X.; Lin, X.; Dang, X. Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Netw.* **2020**, *125*, 258 – 280. [CrossRef] [PubMed]
27. Liu, C.; Du, Y. A membrane algorithm based on chemical reaction optimization for many-objective optimization problems. *Knowl.-Based Syst.* **2019**, *165*, 306–320. [CrossRef]
28. Liu, C.; Shen, W.; Zhang, L.; Du, Y.; Yuan, Z. Spike Neural Network Learning Algorithm Based on an Evolutionary Membrane Algorithm. *IEEE Access* **2021**, *9*, 17071–17082. [CrossRef]
29. García-Victoria, P.; Cavaliere, M.; Gutiérrez-Naranjo, M.A.; Cárdenas-Montes, M. Evolutionary game theory in a cell: A membrane computing approach. *Inf. Sci.* **2022**, *589*, 580–594. [CrossRef]
30. Dong, J.; Stachowicz, M.; Zhang, G.; Cavaliere, M.; Rong, H.; Paul, P. Automatic Design of Spiking Neural P Systems Based on Genetic Algorithms. *Int. J. Unconv. Comput.* **2021**, *16*, 201–216.
31. Casauay, L.J.; Macababayao, I.C.H.; Cabarle, F.G.C.; Cruz, R.T.D.L.; Adorna, H.N.; Zeng, X.; Martínez del Amor, M.Á. A Framework for Evolving Spiking Neural P Systems. In Proceedings of the ACMC 2019: The 8th Asian Conference on Membrane Computing, Xiamen, China, 14–17 November 2019; pp. 271–298.

32. Nishida, T. Membrane algorithms: Approximate algorithms for NP-complete optimization problems. In *Applications of Membrane Computing*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 303–314.
33. Zhang, G.; Cheng, J.; Gheorghe, M.; Meng, Q. A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. *Appl. Soft Comput.* **2013**, *13*, 1528–1542. [CrossRef]
34. Niu, Y.; Wang, S.; He, J.; Xiao, J. A novel membrane algorithm for capacitated vehicle routing problem. *Soft Comput.* **2015**, *19*, 471–482. [CrossRef]
35. Peng, H.; Shi, P.; Wang, J.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. Multiobjective fuzzy clustering approach based on tissue-like membrane systems. *Knowl.-Based Syst.* **2017**, *125*, 74–82. [CrossRef]
36. Orozco-Rosas, U.; Picos, K.; Montiel, O. Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots. *IEEE Access* **2019**, *7*, 156787–156803. [CrossRef]
37. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. Mobile robot path planning using membrane evolutionary artificial potential field. *Appl. Soft Comput.* **2019**, *77*, 236–251. [CrossRef]
38. Song, Q.; Huang, Y.; Lai, W.; Han, T.; Xu, S.; Rong, X. Multi-membrane search algorithm. *PLoS ONE* **2021**, *16*, e0260512. [CrossRef] [PubMed]
39. Tian, X.; Liu, X. Improved Hybrid Heuristic Algorithm Inspired by Tissue-Like Membrane System to Solve Job Shop Scheduling Problem. *Processes* **2021**, *9*, 219. [CrossRef]
40. Niu, Y.; Zhang, Y.; Cao, Z.; Gao, K.; Xiao, J.; Song, W.; Zhang, F. MIMOA: A membrane-inspired multi-objective algorithm for green vehicle routing problem with stochastic demands. *Swarm Evol. Comput.* **2021**, *60*, 100767. [CrossRef]
41. Liu, C.; Shen, W.; Zhang, L.; Yang, H.; Du, Y.; Yuan, Z.; Zhao, H. Improved Membrane Algorithm Under the Framework of P Systems to Solve Multimodal Multiobjective Problems. *Int. J. Pattern Recognit. Artif. Intell.* **2021**, *35*, 2159024. [CrossRef]
42. Peng, H.; Wang, J.; Shi, P.; Riscos-Nunez, A.; Perez-Jimenez, M.J. An automatic clustering algorithm inspired by membrane computing. *Pattern Recognit. Lett.* **2015**, *68*, 34–40. [CrossRef]
43. Zhao, Y.; Zhang, W.; Sun, M.; Liu, X. An Improved Consensus Clustering Algorithm Based on Cell-Like P Systems With Multi-Catalysts. *IEEE Access* **2020**, *8*, 154502–154517. [CrossRef]
44. Bergstra, J.; Yamins, D.; Cox, D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; Dasgupta, S., McAllester, D., Eds.; PMLR: Atlanta, GA, USA, 2013; Volume 28, pp. 115–123.
45. Tan, Y.; Zhu, Y. Fireworks Algorithm for Optimization. In Proceedings of the International Conference in Swarm Intelligence, Beijing, China, 12–15 June 2010; Tan, Y., Shi, Y., Tan, K.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 355–364.
46. Eriksson, D.; Pearce, M.; Gardner, J.; Turner, R.D.; Poloczek, M. Scalable Global Optimization via Local Bayesian Optimization. In *Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
47. Hansen, N.; Müller, S.D.; Koumoutsakos, P. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evol. Comput.* **2003**, *11*, 1–18. [CrossRef]
48. Khattab, H.; Mahafzah, B.A.; Sharieh, A. A hybrid algorithm based on modified chemical reaction optimization and best-first search algorithm for solving minimum vertex cover problem. *Neural Comput. Appl.* **2022**, 1–29. doi: 10.1007/s00521-022-07262-w. [CrossRef]
49. Al-Shaikh, A.; Mahafzah, B.A.; Alshraideh, M. Hybrid harmony search algorithm for social network contact tracing of COVID-19. *Soft Comput.* **2021**, 1–23. doi: 10.1007/s00500-021-05948-2. [CrossRef] [PubMed]