Adaptive Nested Neural Network (ANNN) Based on Human Gene Regulatory Network (GRN) for Gene Knowledge Discovery Engine

Zainal A. Hasibuan[1] **Romi Fadhilah Rahmat**[2] **Muhammad Fermi Pasha**[2] **Rahmat Budiarto**[2]

[1]Faculty of Computer Sciences, University of Indonesia, Jakarta, Indonesia
[2]School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang, Malaysia

Summary
For a long time neural networks have been a popular approach for intelligent machines development and knowledge discovery. However, problems still exists in neural networks, such as fixed architecture and excessive training time. One of the solutions to unravel this problem is by using neuro-genetic approach. A neuro-genetic approach is inspired by a theory in neuroscience which state that the evolution of human brain structure is significantly affected by its genome structure. Hence, the structure and performance of a neural network should be based on a gene created for it. Therefore, to overcome these existing neural network problems and with the help of new theory of neuroscience this paper we attempt to propose a neuro-genetic approach by using simple Gene Regulatory Network (GRN) as a more biologically plausible model of neural network.
Firstly, we proposed GRTE, a Gene Regulatory Training Engine to control, evaluate, mutate and train genes. Secondly, ANNN, a distributed and Adaptive Nested Neural Network that will be constructed based on the genes from GRTE. This paper focuses on ANNN for Uncorrelated Data. We conducted experiments to evaluate and validate by using the Proben1's Gene Benchmark Datasets. The results of the experiments validate the objective of this work.

Key words:
Neuro-Genetics, Neural Network, Gene Regulatory Network,

1. Introduction

Behind all success stories of neural network, there are still some major drawbacks which causes neural network being unable to continuously learn and act like the actual human brain. Though many people believe that this issue is due to the limitation of the current computer processing power, the recent discoveries in neuroscience field has raises some doubts. One of the recent theories in neuroscience state that interaction among genes, biochemical, and neural activity will affects the complexity and construction of neurons in human brain (Marcus, 2004). This theory has become obvious when [1] build NeuroGene applications to simulate process of neural development which involves gene regulation.

Gene regulation in human is controlled by Gene Regulatory Network (GRN). GRN (see Figure 1) can be represented as complex connections among gene that lead to mutation and interactions between genes [2]. GRN also holds the evolution, adaptation and interaction phase of DNA, RNA, mRNA, and protein in cell level [3].

Several attempts have been made to propose adaptive and evolvable frameworks. Among these adaptive frameworks are Evolving Connectionist System (ECOS) [4], MAB-Net as dynamic brain model [5], NeuroEvolution of Augmenting Topologies (NEAT)[6], and also Evolvable-Neural-Based Fuzzy Inference System (EFIS) [7]. All these adaptive and evolvable neural network models are generally able to solve fixed architecture problem and the catastrophic forgetting problem. But the training time and the limitation of processing power resources made it difficult to deal with vast amount of data and in the same time evolve its complex structure.

This paper proposes a Distributed and Adaptive Nested Neural Network (ANNN) as well as a Simple Gene Regulatory Network (GRN) as a Controller and Learning Engine. This research shall lead into modeling and implementing simple Gene Regulatory Training Engine with adaptive structure, less training time, and optimum CPU usage, for the purposes of interaction, mutation and training of the gene. Furthermore, with the intention of verification, we conduct experiment on gene knowledge discovery using Proben1's Gene Benchmark Datasets.

The rest of the paper is organized as follows. Section 2 discusses various works that have been done by many researchers. Section 3 discusses the proposed GRTE and ANNN architecture. Section 4 presents the experimental analysis and results of our proposed method. Finally, in Section 5, we conclude the paper.

Manuscript received June 5, 2009
Manuscript revised June 20, 2009

2. Related Works

Many works have been done to overcome conventional Artificial Neural Networks. We category the works into two: Gene Regulatory Network and Adaptive Nested Neural Network.

2.1 Gene Regulatory Network

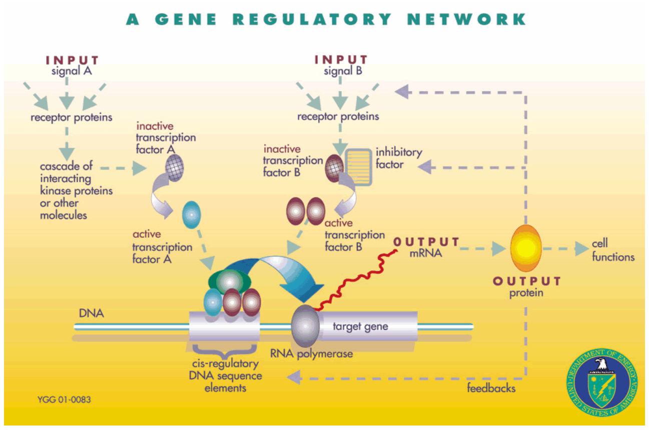

A GENE REGULATORY NETWORK
INPUT signal A
INPUT signal B
receptor proteins
receptor proteins
inactive transcription factor A
cascade of interacting kinase proteins or other molecules
inactive transcription factor B
inhibitory factor
active transcription factor A
active transcription factor B
OUTPUT mRNA
cell functions
OUTPUT protein
DNA
target gene
RNA polymerase
cis-regulatory DNA sequence elements
feedbacks
YGG 01-0083

Figure 1 Biological Gene Regulatory Networks Model [8].

Modeling gene regulatory network from biological perspective to computational perspective has been developed since 1999 by researchers from INRIA Rhône-Alpes's bioinformatics group [2]. Several models have been produced to represent GRN with many different developments method. Akutsu presented simplest and imprecise GRN modeling using Kauffman Boolean network with Boolean vectors for genes state [9]. An extensive representation was made by [10] using Bayesian and regression networks with the transitional probabilities. It is continued by [11] which uses artificial neural networks of a Hopfield type to become the core principle for GRN computational model development.
Another development applies evolving connectionist network or neural network which represents genes as neurons, and interactions between genes as path weight connections [12] and [4]. Along with that, GRN modeling has been applied in the area of adaptive fuzzy connectionist systems by [13], clustering analysis [14], genes data analytical modeling [15], gene expression correlation analysis [16] and gene evolution based on fitness function in the evolutionary computing area [17].

2.2 Adaptive Nested Neural Network

The area of brain-like computing research has been an active area of research for the past decades. It starts with the first wave of human brain's neurons connections modeling known as neural network or connectionist system (see Figure 2) introduced by McCulloch and Pitts in 1943. The wave continued with the learning theory introduced by [18] known as the Hebbian learning, the introduction of genetic algorithm by [19] and still progressing with more and more complex model such as self organizing maps [20], recurrent neural network [21], spiking neural network [22] and many more.

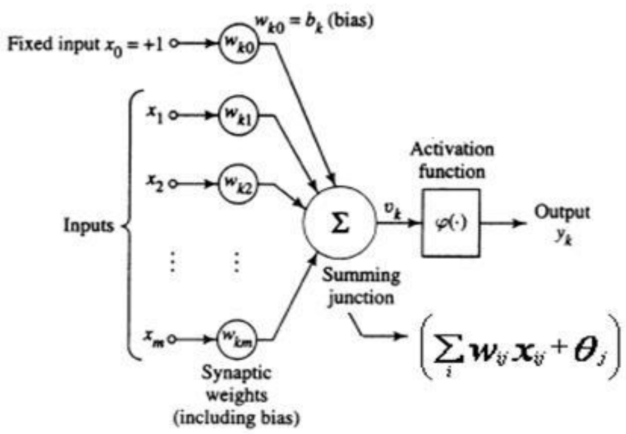


Figure 2 Artificial Neural Network [30]

Most attempts in developing intelligent machines involved neural network as a media to learn. To date the issue, neural network has been used to solve numerous complex real world problems across different fields of study such as pattern recognition (which include speech and image recognition) [23] and [24], robotics [25], visual monitoring [26], prediction problem [27], applied physics [28], etc.

Essentially, the main problem lies in the fact that neural networks have little similarity with the actual processes in the brain other than both being constructed of many interconnected units. To address this issue, a more biologically plausible model that reflects the actual human intelligence is needed. Ipso facto, we need to first understand how intelligence is developed in our brain from neuroscience perspective.

With the dynamic or adaptive architecture of neural network, it is possible to create different neural network architecture. One of the types is modular neural networks or often called nested neural networks. According to the literature means, we can conclude that nested neural network is a neural networks composed by several embedded neural networks.

Many problems occur due to the intensity and quantity of data input in neural network. Massive input data that flows to input layer in a neural network, will create long training time and sometimes create training's malfunction. Nested neural networks are concerned with this kind of problem and it can be a solution by using partition of data input distributed for several neural networks in Nested Neural Networks architecture (see Figure 3).

Figure 3 describes how neural network partitions input data. We can conclude that Nested Neural Networks is separating data into different neural networks. After training phase in A and B, the output of those two neural networks will be functionally changed as inputs to Neural Network C. (there is a chance to change the output as a bias). Then Neural Network C will perform its training after completing those previous neural networks.

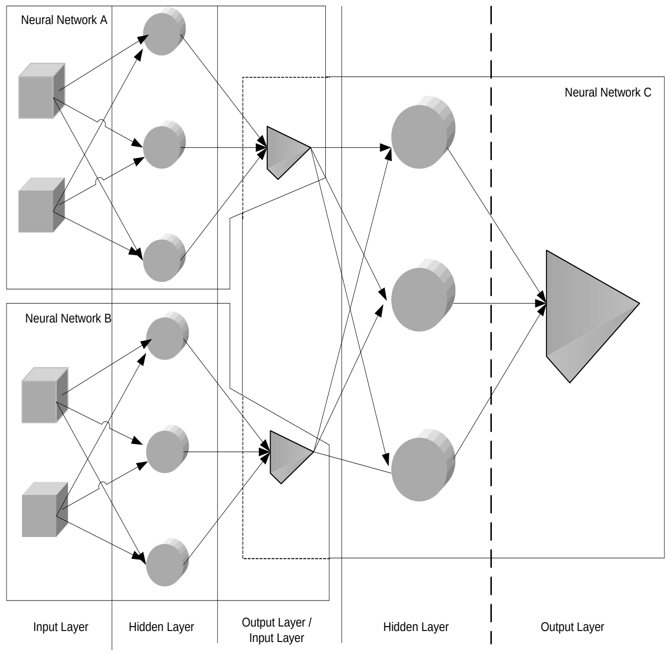


Figure 3 Nested Neural Networks

Several researches has been exposed to use Nested Neural Networks, for example in image compression (Kumar, 1998), and in Hierarchical Cluster Model that has been described as a parallel neo-cortex neural network's in brain model [29].

3. ANNN Architecture

Our proposed architecture comprises of two main methods/components as follows.

- Distributed and Nested Adaptive Neural Network using ANNN
- Gene Regulatory with Gene Regulatory Training Engine (GRTE)

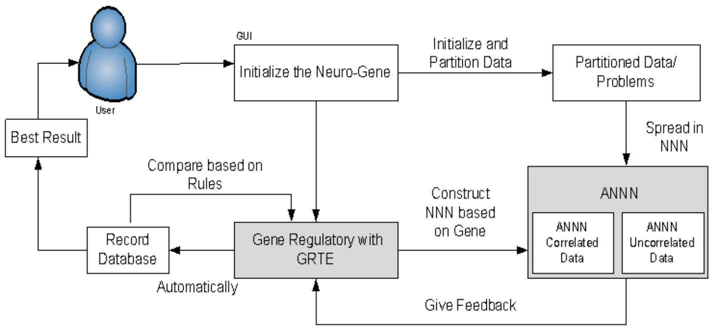


Figure 4 Proposed Neuro-Genetic Approach Architecture

The construction of ANNN depends on the output from GRTE, and the output from ANNN will be evaluated in GRTE (see Figure 4). This two ways connection shows the connection between ANNN and GRTE..

3.1 Uncorrelated Data ANNN

Uncorrelated data ANNN means the ANNN is not suitable for n-bits parity problem. This ANNN model is modeled to receive different inputs from every different agent. It is also suitable in order to solve problem in the network, such us network traffic problems, network forensic and supervised data with huge data samples. The diagram of ANNN model for Uncorrelated Data is shown in Figure 5.

Figure 6 shows the algorithm to perform Uncorrelated Data ANNN based on Figure 5.

3.2 GRTE

Our proposed GRTE has 5 components as follows:

- Gene Representation
- Fitness Function
- Mutation Regulatory
- Training Process
- Neuro-Gene

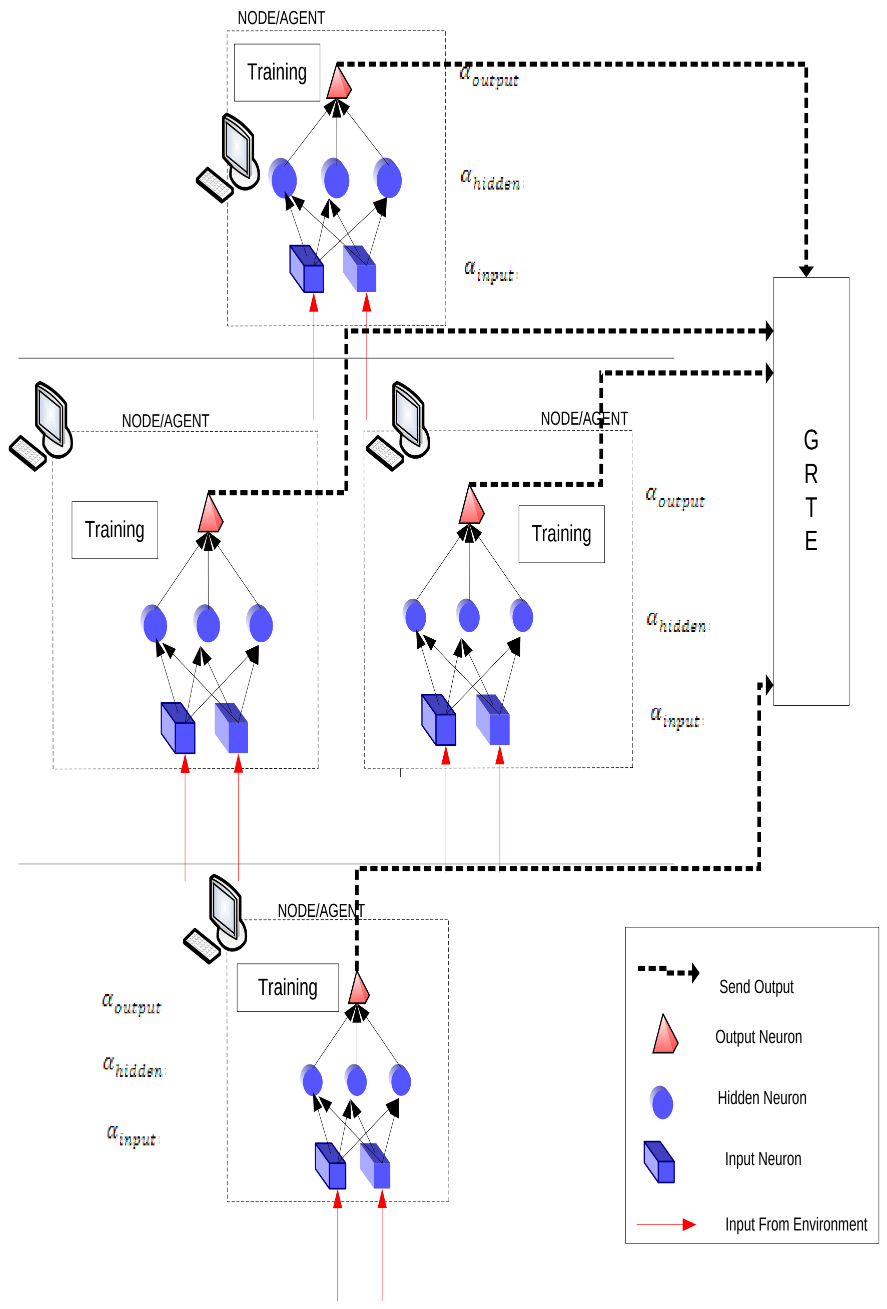


Figure 5 ANNN Model for Uncorrelated Data

```
Algorithm of ANNN Uncorrelated Data
 1:  Calculate number of Node/Agent $(N_{NG})$ based on data partition.
 2:  Initialize inner layers $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, neurons within inner neural
     networks $(\mu_j)$ and weight connections $(\omega_{ij})$.
 3:  for every Node/Agent $(1 \rightarrow N_{NG})$
 4:    for every neural network $\eta_{(layer,variable)}$
 5:      Construct neurons { $\mu_{layers,1}$, $\mu_{layers,2}$, …, $\mu_{layers,p}$ } where every $layers$
         is $\{\alpha_1, \alpha_2, \ldots, \alpha_{m+p}\}$ ,  and weight connection $\{\omega_{11}, \omega_{12}, \omega_{21}, \ldots, \omega_{ij}\}$.
 6:    end for
 7:    Train neural network
 8:    Calculate objective function $E^p$
 9:  end for
10:  Send $E^p$ to GRTE
11:  Back to Step 1 if GRTE send the new Gene
```

Figure .6 Algorithm for Uncorrelated Data ANNN

```
Pseudocode of GRTE processes
 1: process GRTE()
 2: initializeGene()
 3:    while not $(F_{CD} \leq 0.030) \parallel (F_{UD} \leq 0.030)$ do
 4:        trainingGene()
 5:        send_Neural_Network()
 6:        execute_Neural_Network()
 7:        retrieve_Result()
 8:        evaluate_ Fitness_Function_for_ANNN_Correlated_Data()
 9:        evaluate_ Fitness_Function_for_ANNN_Uncorrelated_Data()
10:        begin $\mathbf{while}\ (0.01 \leq E_p \leq 0.98)\ \mathbf{do}$
11:            execute_Mutation_Regulatory()
12:            evaluate_ Fitness_Function_for_ANNN_Correlated_Data()
13:            evaluate_ Fitness_Function_for_ANNN_Uncorrelated_Data()
14:        end while
15:        crete_new_individu()
16:        send_and_compare_to_Population()
17:        choose_best_individu()
18:    end while
19: end process
```

Figure 7 Pseudocode for GRTE Process

3.2.1 Gene Representation

In general, GRTE uses 3 different set of Gene-like's variable which will interact with the environment.

- First gene is Node Behavior's gene for the node in the environment. The function of the node behavior is to differentiate the model of ANNN architecture that should be formed.

- Second gene is Inner Neural Network (INN)'s gene variable. It contains the variable of neural network parameters for every node in the layer. The parameters include number all neurons, number of input nodes, hidden nodes, output nodes, epoch learning rate, and momentum rate.

- The last gene is Outer Neural Network (ONN)'s gene variable. It contains parameters for construction of outer neural network. They are number of node, number of input, hidden and output neurons.

3.2.2 Fitness Function

Before we describe fitness function, firstly we have to introduce the objective function. The objective function basically used to verify and measure the objectiveness of such a neural network. We find that for our architecture (multi-layer feed-forward networks) is suitable if we use *root mean squared error*. The root-mean squared error can be formulated as:

$$E^p = \sqrt{\frac{1}{No}\sum_{o=1}^{No}(d_o^p - y_o^p)^2}$$

Where E^p represents root of the difference between the d_o^p desired output value for unit o with its particular pattern p and the actual network output y_o^p for every training sample, in which No is the number of output unit. Every gene has its own objective function; it means we can get the average objective function E_{AVG} for all of the genes by equation:

$$E_{AVG} = \frac{\sum_{i=0}^{N_{NG}} E_p}{N_{NG}}$$

The above quation can be calculated after one training of GRTE. When this happen, we can say, there will be a different data sample to be trained in each partition data. For example, data sample in neural network for input layer can be bigger than data in the hidden or output layer. Therefore, after GRTE sends the gene to the environment, the ANNN will produce the final output O_{CD}. Final output O_{CD} is used to measure the error with the expected output E_o, hence we can get the fitness of the output F_o, denote by:

$$F_o = |\, E_o - O_{CD} \,|$$

From the three equations above we can identify how to get fitness function in one generation of GRTE for Correlated Data ANNN. The fitness function is:

$$F_{CD} = E_{AVG} \,.\, F_o = \frac{\sum_{i=0}^{N_{NG}} E_p}{N_{NG}} \,.\, |\, E_o - O_{CD} \,|$$

$$= \frac{\sum_{i=0}^{N_{NG}} \sqrt{\frac{1}{No}\sum_{o=1}^{No}(d_o^p - y_o^p)^2}}{N_{NG}} \,.\, |\, E_o - O_{CD} \,|$$

In one generation of process in the uncorrelated data ANNN, there will be final output of uncorrelated data ANNN of every agent/node $O_{UD(i)}$ where $i = 1,2 \ldots, N_{NG}$. Since the data is uncorrelated, the Fitness Function F_{UD} can be calculated as the average of all objective function. We can denote it by the following equation:

$$F_{UD} = E_{AVG} = \frac{\sum_{i=0}^{N_{NG}} E_p}{N_{NG}}$$

$$= \frac{\sum_{i=0}^{N_{NG}} \sqrt{\frac{1}{No}\sum_{o=1}^{No}(d_o^p - y_o^p)^2}}{N_{NG}}$$

3.2.3 Mutation Regulatory

Basic requirement to give adaptive ability is to mutate several entities that we listed in Table 1.

All of GRTE's variables have different function and purpose in constructing ANNN. By having this different we can do optimization of these variables by applying Genetic Algorithm (GA). Before we proceed to GA inside GRTE, we have to describe the Mutation Regulatory. After we analyze the GRTE's gene representation, the genetic operator that can occur in this representation is only mutation operator. It means there is no cross-over occurs, no parent selection (in other words every gene are a single individual that has to be mutated with Mutation Regulatory). Mutation Regulatory consists of algorithm and equations to control a gene to be mutated. Since we want to mutate several entities (see Table 1) we provide the algorithm and equations below.

Table 1 Mutation Entity List

Mutation Entity (Inner Neural Network)	Explanation
Input Neurons	Mutation is depend on the kind of data that want to be used
Hidden Neurons	Mutation is depend on Mutation Regulatory(see Section 3.3.2)
Epoch	Can be Random and structured with relation with hidden neurons mutation (see Section 3.3.2)

The Mutation Regulatory for input neurons is related to variance of data to be used. Suppose that we want to apply the Correlated data ANNN in n- bits parity problem which means we have to have n inputs. Assume that we want to have multilayer NNN; it means n has to be $2 \leq n \leq \infty$. However, if it is uncorrelated data ANNN we can specify ($NI \geq 1$).

Mutation Regulatory for hidden neurons is different from Mutation Regulatory for input neurons. From this sequence of algorithm below, we know that this algorithm is the adaptive part to search the best gene by mutating the hidden layer and its neurons inside it. The mutation regulatory for ANNN Correlated data is described in the following algorithm.

Pseudocode of Hidden Neurons Mutation Regulatory

```
1:  initialize (NH=2) and (Epoch=10)
2:  mutate_hidden_neuron (NH=NH+(2 .(Random(4)))
3:  mutate_epoch (Epoch=Epoch+(10 .(Random(4)))
4:  begin  IF (|E_p-E_(p-1) |<0.1)
5:    stop_hidden_neuron_mutation;
6:    mutate_epoch (Epoch=Epoch+(100.(Random(30)))
7:  end IF
```

Another factor to optimize the objective/fitness value is epoch mutation. We can combine both of hidden neurons *NH* and epoch as one equation for Uncorrelated Data ANNN; it is because of the nature of portioned data which allow the equation possible. The equation is:

$$Epoch = NH \, . \, 10 \text{ and } NH = \frac{Epoch}{10}$$

Since we have different data partitioning, the appropriate number of output neurons NO for Correlated Data ANNN depends on the data and should be $NO > 1$. For Uncorrelated Data ANNN, number of output neurons is $NO = 1$, and. For rate and momentum, it can be mutated also. However, for simplicity we make it as a fixed number with $rate \geq 0.5$ and $momentum \geq 0.7$ for both modes of ANNN.

3.2.4 Training Process

Training process is a process in Neural Network to get appropriate weight connection to give a correct result. We have two different places to conduct training process in our system. The first place is inside the GRTE, for Correlated Data ANNN. The second place is for Uncorrelated Data ANNN, the training process will be held inside the Agents.

3.2.5 Neuro-Gene

Neuro-gene transformation is a process to create an execution or pre-execution file. Neuro-gene has the ability to execute, or train the neural network inside the Agents.

3.3 Main Algorithm

The GRTE Algorithm is an adaptive algorithm with Fitness Function F_{CD} or F_{UD} as termination criteria. GRTE also involves in training, sending and retrieving the result from ANNN

GRTE adopts the GA concept in its algorithm; in fact it is a simple GA with only mutation operator inside. The algorithm in Figure 7 explains how GRTE actually adapting GA algorithm in line 8 till line 17. What makes it different from common GA is our GRTE does not use crossover and parent selection (every gene act as a single parent), and the randomize initialization of population does not occur in the algorithm.

Fitness Function and Mutation Regulatory has become an important element for GRTE to do its algorithm. The GRTE will become more adaptive and can be used to support Correlated and Uncorrelated Data ANNNs. The first thing to do is translating the ONN to construct the INN. The genes produced by INN have to be trained, before the genes send to the environment. Results from environment will be analyzed using Fitness Function and placed in population. Since Fitness Function does not affect the mutation in ONN level, than mutation process will be held in INN. The rule of mutation will be conducted by Mutation Regulatory for controlling the objectiveness of the gene. From this point, GRTE will create new gene to be prepared for training and to be sent to the environment. This iteration will be stopped after Fitness Function reaches the termination point.

4. Experiment

Proben1 is a collection of 15 datasets from 12 different problems and domain in neural network for the purpose of neural network learning in the term of pattern classification. Each of the datasets contains realistic problems or so called diagnosis tasks. It can be retrieved from the Neural Bench archive7 at Carnegie Mellon University and from UCI machine learning databases repository and the energy predictor shootout archive (Precheltz, 1994).
Besides of its variety and real data, Proben1 also suited for supervised learning. Therefore we design an experiment to test our proposed Uncorrelated Data ANNN model using appropriate data in Proben1 benchmark datasets. The purpose is to identify performance and adaptive ability of our model to be adapted to the situation that given from datasets. After we analyze all the data from Proben1 dataset, gene dataset is the most suitable to be test data of our Uncorrelated Data ANNN. Gene dataset consists of three different dataset, Gene 1, Gene 2, and Gene3. Each dataset have 3175 samples data, which consist of 1558 training samples, 794 validation samples, and 793 test samples respectively. The data also have 120 input values and 3 output values. In shake of simplicity and decreasing training time, our experiment will be held with 1558 data sample from training sample. This particular 1558 training sample has to be divided, our way to conduct this data partitioning is by using the data representation.
Data partitioning in Uncorrelated Data ANNN is slightly different with Correlated Data ANNN. Here, data is divided on the perspective of sample's number and variance of the data. Gene's Proben1 dataset, represents one alphabet in the gene as couple numbers of input. It represents C as (-1, 1), A as (-1, 1), G as (1, -1), and T as (1, 1). With these four representations we can create number of neuro-gene corresponds with the representation. If the first two input of every samples is (-1,-1) then it is appointed to neuro-gene 1, if it is (-1, 1) appointed it to neuro-gene 2, (1,-1) appointed to neuro-gene 3, and (1, 1) appointed to neuro-gene 4. It means the number of neuro-gene that will be created in GRTE is four. The partition of the data is shown in Table 2.

Table 2 Sample Data Partitioning for ANNN Uncorrelated Data

Data Prefix	Gene 1	Gene 2	Gene 3
(1, 1)	351	350	356
(1, -1)	454	432	409
(-1, 1)	337	352	375
(-1, -1)	416	424	418

4.1 Experimental Set up

Our implementation application is divided into two different parts; they are Server Application and Client/Agent Application. Figure 8 shows the diagram of Client/Agent-Server Application.
From application perspective we can divide our methodology in Client/Agent Application and Server Application. Inside the Agent Application, there are three different modules. Firstly, there is data collector, which is purposed to collect data from benchmark datasets. This data should be partitioned. Secondly is Input Analyzer, it is used to analyze and classify the data's, or it can be describes as pre-processing data module. Lastly is Agent's Neural Network Engine, used to run the neuro-gene engine that has been injected to the node by using the dataset as the input resources.

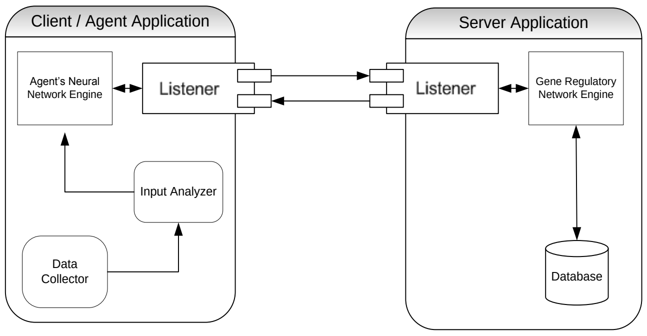

Client / Agent Application
Agent's Neural Network Engine
Listener
Input Analyzer
Data Collector
Server Application
Listener
Gene Regulatory Network Engine
Database

Figure 8 Client/Agent-Server Applications

For server side there will be one module and one database. The module is Gene Regulatory Network, which creates the neuro-gene and as a home-base of neural network training and Genetic Algorithm. GRTE is connected to database to record gene activity and population.

Figure 9 explains the process-flows of Server Implementation. The first step describes the user interaction with the servers. User needs to give the initialization for the neuro-gene in server, before it runs adaptively. What will happen next is the entire nodes in the environment will request neuro-gene file through opened port to the GRTE server, then GRTE server will send its particular file to the particular node with an ID to prevent the unstructured nested neural network architecture.

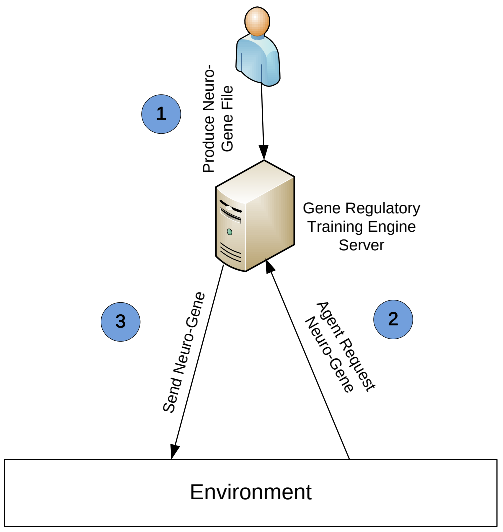


Figure 9 Process-flow of Server Implementation

Figure 10 gives an example of process flow in the environment. First step should be the process of activating the agents. After the agents activated, agent will request neuro-genes to the server, then server gives the neuro-gene. From here every agent should start the neural-network activity inside the node and the architecture will be built automatically inside the environment based on given neuro-genes. Every agent will record the information of their activities to be used later in the GRTE as the parameter for performance measurements.

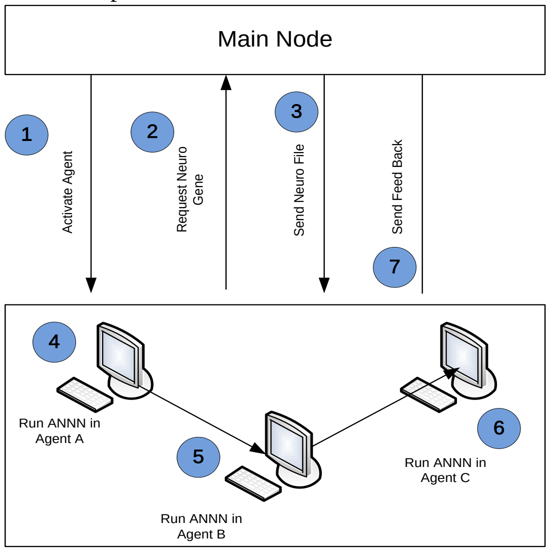


Figure 10 Process-flow of Environment Agent Implementation

4.2 Result Analysis

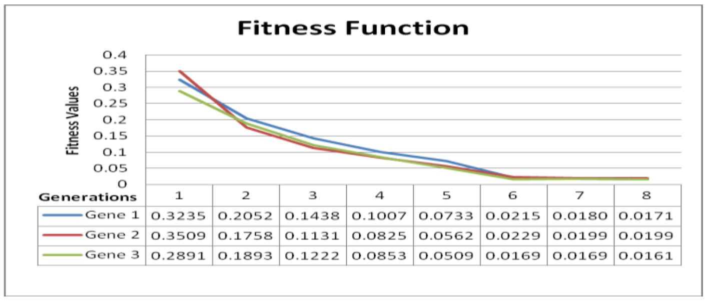
The first results come from Objective Function of every gene dataset. We attempt to test our ANNN model with 3 different gene dataset. Every dataset had been partitioned into 4 (see Table 2). E^P is expected to be lower in every generation, to prove that our neural network evolves properly. .

After eight generations it becomes $E^P = 0.0088$. It shows that our neural network in our proposed approach reaches its objective and works properly. Objective Function also shows the accuracy of our neural network. The accuracy measured by percentage of correctness based on average of all objective values in gene dataset. We obtain the accuracy as shown in Table 3.

Table 3 Accuracy of ANNN Uncorrelated Data

Gene 1	Gene 2	Gene 3
98.3%	98%	98.15%

The result described here is a Fitness Function results from our GRTE method. We follow the standard procedure of fitness testing. Figure 11 shows that the Fitness Function of Gene 1, Gene 2 and Gene 3 converge at the same best result in generation 6 (F_{UD} reaches ≤ 0.030). It shows that our GRTE works properly for this kind of data.



Generations	1	2	3	4	5	6	7	8
Gene 1	0.3235	0.2052	0.1438	0.1007	0.0733	0.0215	0.0180	0.0171
Gene 2	0.3509	0.1758	0.1131	0.0825	0.0562	0.0229	0.0199	0.0199
Gene 3	0.2891	0.1893	0.1222	0.0853	0.0509	0.0169	0.0169	0.0161

Figure 11 Fitness Function for the three genes.

4.3 Statistical Analysis of the Experiment

After the one-full run of the GRTE, we try to experiment with 5 runs starts from generation 5 till generations 8. It means, in our GRTE we have 3 dataset, with each data contains 4 partitions; each partition runs 5 times from generation 5 till generation 8. In other word, our GRTE produce 3*4*5*3 = 180 neuro-genes training.
The results are shown in Table 4 and Table 5. The mean of Objective Function for each gene in Figure 12 shows that after 6 generation, the result is convergent.

5. Conclusion

The experiment shows the ability of our methods to adapt the node-only mode based on ruled partitioning data. We have shown that our GRTE could evolve successfully using Proben1's Gene Benchmark Datasets. The Fitness

Function also reaches its best using the equation and algorithm of our Mutation Regulatory of GRTE.

In terms of adaptive ability, the mutation regulatory obtained a good result. What we pointed out here is that our mutation regulatory method still can be better, if we involve another mutation entity, such as rate, momentum and hidden neurons of outer neural network mutation. Since our training process is held in the Agents, the performance of our approach significantly improved. The result from fitness function also reaches the best level, and because of that, we can conclude our experiment's objective achieves a good result.

Acknowledgments

The authors would like to thank the Universiti Sains Malaysia (USM) for the research grant.

References

[1] Storjohann, R., Marcus, G., NeuroGene: Integrated Simulation of Gene Relation, Neural Activity and Neurodevelopment. Proc. Of International Joint of Conference on Neural Networks, Montreal,Canada, 2005.

[2] De Jong, H., Modeling and simulation of genetic regulatory systems: a literature review, Journal of Computational Biology, vol.9, No.1, 67-102, 2002

[3] Pevzner, P., A., Computational Molecular Biology: An Algorithmic Approach, MIT Press, 2000.

[4] Kasabov, N., Evolving Connectionist System: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines, Springer-Verlag, London, 2003.

[5] Ohtani, S., Ishido, M. and Shimohara, K., "Multi-Agent Based Neural Network as a Dynamical Brain Model", Proceedings of the Fifth International Symposium on Artificial Life and Robotics (AROB5'00), Oita, Japan, 2000.

[6] Stanley, K. O. and Miikkulainen, R., Evolving Neural Networks Through Augmenting Topologies, Evolutionary Computation, 10(2), pp. 99–127, 2002.

[7] Pasha, M. F., Budiarto, R., Syukur, M. and Yamada, M., "EFIS: Evolvable-Neural-Based Fuzzy Inference System and Its Application for Adaptive Network Anomaly Detection" In D.S.Yeung et al. (Eds.) Advances in Machine Learning and Cybernetics - ICMLC2005, Lecturer Notes in Artificial Intelligence (LNAI), vol. 3930, Springer-Verlag, pp. 662-671, 2006.

[8] DEO Genomics: GTL Systems Biology for Energy and Environment. (2006) U.S. Department of Energy Genome Programs, Retrieved from: http://genomics.energy.gov/gallery/. 05/12/07.

[9] Akutsu, T., Miyano, S., and Kuhara, S., "Identification of genetic networks from a small number of gene expression patterns under the boolean network model," Pacific Symposium on Biocomputing, vol. 4, pp.17-28, 1999.

[10] Kato, M., Tsunoda, T., Takagi, T., Inferring genetic networks from DNA microarray data by multiple regression analysis, Genome Informatics, 11, 118-128, 2000.

[11] Marnellos, G., and Mjolsness, E., D., Modeling Neural Development, MIT Press, Cambridge, MA, pp. 27-28, 2003.

[12] Vohradsky, J., Neural network model of gene expression, The FASEB Journal, vol. 15, March, pp.846-854, 2001.

[13] Sehgal, M., Gondal, I., Dooley, L., AFEGRN: Adaptive Fuzzy Evolutionary Gene Regulatory Network Reconstruction Framework, IEEE International Conference on Fuzzy Systems, Canada, 2006.

[14] D'Haeseleer, P., Liang, S., and Somogyi, R., "Genetic network inference: from co-expression clustering to reverse engineering", Bioinformatics, vol. 16, no. 8, pp.707-726, 2000.

[15] Kohn, W., K., and Dimitrov, D., S., Mathematical Models of Cell Cycles, Computer Modeling and Simulation of Complex Biological Systems, 1999.

[16] Lindlof, A., and Olsson, B., Could Correlation-based Methods be used to Derive Genetic Association Networks?, Proceedings of the 6thJoint Conference on Information Sciences, March 8-12, pp.1237-1242, 2002.

[17] Mimura, and Iba, H., "Inference of a Gene Regulatory Network by Means of Interactive Evolutionary Computing," Proceedings of the 6th Joint Conference on Information Sciences, March 8-12, pp.1243-1248,2002.

[18] Hebb, D., *The Organization of Behavior, John Wiley and Sons, New York, 1949.*

[19] Goldberg, D. E., *Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, Reading, 1989.*

[20] Kohonen, T., Self Organizing Maps, Springer, Berlin, Heidelberg, 1995.

[21] Mandic, D. and Chambers J., Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability, Wiley, 2001.

[22] Christodoulou C., Bugmann, G., and Clarkson, T. G., "A spiking neuron model: applications and learning", Neural Networks, vol. 15, pp. 891–908, 2002.

[23] Ghobakhlou, A., Watts, M. and Kasabov, N., "Adaptive speech recognition with evolving connectionist systems" Information Sciences, vol. 156, 2003, pp. 71-83.

[24] Gou, S. and Jiao, L., "Image Recognition Using Synergetic Neural Network" Advances in Neural Network - ISNN2005, Lecturer Notes in Computer Sciences (LNAI), vol. 3497, Springer-Verlag, pp. 286-291, 2005.

[25] Sun W. and Wang, Y. N., "A Robust Robotic Tracking Controller Based on Neural Network", International Journal of Robotics and Automation, 20(3), 2005.

[26] Yongtae D., "Tracking People in Video Camera Images Using Neural Networks" Advances in Intelligent Computing, Lecturer Notes in Computer Sciences (LNAI), vol. 3644, Springer-Verlag, pp. 301-309, 2005.

[27] Liu, F., Findlay, R. D. and Song, Q., "A Neural Network Based Electrical Loss Prediction of Bare Overhead ACSR Conductors" Proceeding of the First International Conference on Innovative Computing, Information and Control - Volume II (ICICIC'06), IEEE Press, pp. 392-395, 2006.

[28] Syukur, M., Pasha, M. F., and Budiarto, R., "A Neural Network-Based Application to Identify Cubic Structures in Multi Component Crystalline Materials using X-Ray Diffraction Data" International Journal of Computer

Science and Network Security (IJCSNS), 7(2), pp. 49-54, 2007.
[29] Yap, K., H., Guan., L., Liu., W., A Recursive Soft-Decision Approach to Blind Image Deconvolution, IEEE Transactions on Signal Processing, Vol. 51, Issue 2, pp. 515 – 526, Feb. 2003.
[30] Gan, J., Problem Solving using Neural Network: A Tutorial, Retrieved From www.sx.ac.uk/ccfea/NewsEvents/seminarsetc/seminars/archive/past/John%20Gan.pdf , 2003

Zainal Hasibuan received the BSc. from Bogor Institute of Agriculture, Indonesia in year 1986, MSc., and PhD in Information Sciences from Indiana University in year 1989 and 1995, respectively., He is currently as a researcher and Academic Staff at Faculty of Computer Science, University of Indonesia (UI), Jakarta, Indonesia, His research interests include Information Storage and Retrieval System, E-Learning Systems as well as Grid Systems.

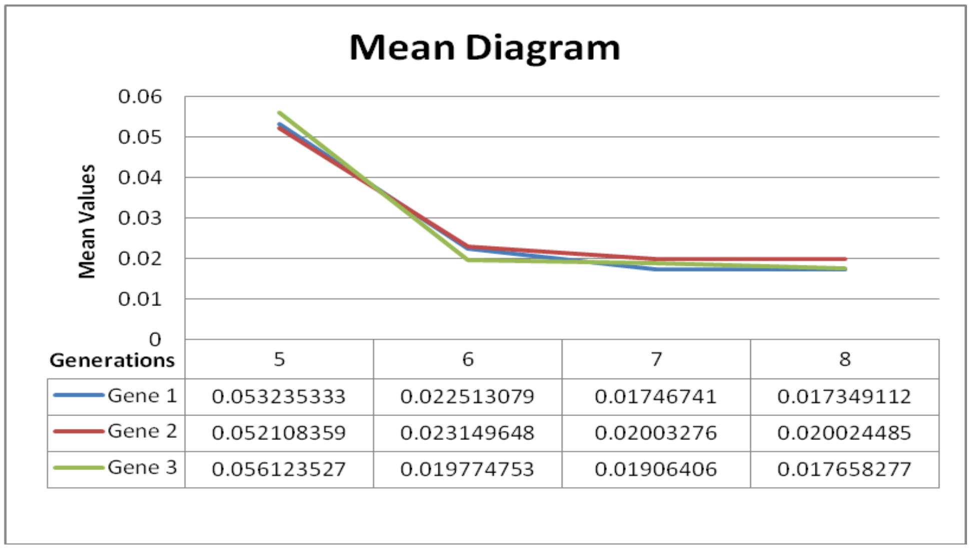
Romi Fadhilah Rahmat received his BSc. and MSc in Computer Sciences from Universiti Sains Malaysia (USM). He is currently a PhD Candidate at the School of Computer Sciences, USM. His research interests include brain modeling and quantum computing.

Muhammad Fermi Pasha received his BSc. and MSc in Computer Sciences from Universiti Sains Malaysia (USM). He is currently a PhD Candidate at the School of Computer Sciences, USM. His research interests include brain modeling, evolving systems and network monitoring.

Rahmat Budiarto received B.Sc. degree from Bandung Institute of Technology in 1986, M.Eng, and Dr.Eng in Computer Science from Nagoya Institute of Technology in 1995 and 1998 respectively. Currently, he is an associate professor at School of Computer Sciences as well as the deputy director of National Advanced IPv6 (NAv6) Center, USM. His research interest includes IPv6, Network Security, Intelligent Systems. He was chairman of APAN Security Working Group.

Table 4 GRTE's Training Set Result in 5 Runs from Generation 5 until Generation 8

	Training Set (Generation 5)			
	Mean	St. Deviation	Best	Worst
Gene 1	0.053235333	0.009017695	0.038196	0.060059
Gene 2	0.052108359	0.006574782	0.046294	0.059739
Gene 3	0.056123527	0.004567437	0.50759	0.061481
	Training Set (Generation 6)			
	Mean	St. Deviation	Best	Worst
Gene 1	0.022513079	0.00386124	0.018911	0.028879
Gene 2	0.023149648	0.00142499	0.0217787	0.025314
Gene 3	0.019774753	0.00250884	0.0170197	0.023295
	Training Set (Generation 7)			
	Mean	St. Deviation	Best	Worst
Gene 1	0.01746741	0.0006743	0.016411	0.0181105
Gene 2	0.02003276	0.00024025	0.0196672	0.0202456
Gene 3	0.01906406	0.00016075	0.01884906	0.01920871
	Training Set (Generation 8)			
	Mean	St. Deviation	Best	Worst
Gene 1	0.017349112	0.000891341	0.0159336	0.01806952
Gene 2	0.020024485	0.000229594	0.019689435	0.02026467
Gene 3	0.017658277	0.001729001	0.015649829	0.018966511



Generations	5	6	7	8
Gene 1	0.053235333	0.022513079	0.01746741	0.017349112
Gene 2	0.052108359	0.023149648	0.02003276	0.020024485
Gene 3	0.056123527	0.019774753	0.01906406	0.017658277

Figure 12 Mean of the Objective Functions for each gene

Table 5 GRTE's Validation Set Result in Generation 8

	Validation Set			
	Mean	St. Deviation	Best	Worst
Gene 1	0.00758	0.000332	0.00814	0.00708
Gene 2	0.00720	0.000214	0.00693	0.00739
Gene 3	0.00676	0.000332	0.00709	0.00633