# Evolving Boolean Networks Through Genetic Learning Algorithms

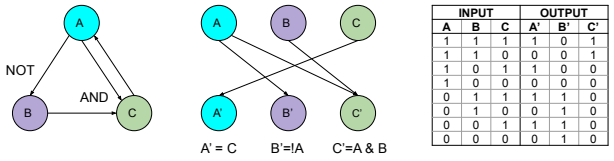

## Abstract

Decision making scenarios for simple organisms are simulated using Boolean networks and adapted to fit the demands of a given scenario. The way these networks are modified is meant to mimic the biological phenomenon of evolution. The more successful networks are kept and modified to see if a better performing network arises. These networks are meant to model an organism's specific reactions to stimuli. The simulation's variables can be altered so that changes in the resulting graphs can be studied for future development of these systems.


## Introduction

Various organisms have to rely on the given stimuli from their environment (visual, tactile, chemical, etc.) to develop and understanding about the area around them. These stimuli are processed by the organism and directly or indirectly attribute to a response given by that organism. For organisms with simple nervous systems or none at all, these responses still require some amount of processing, but not to the same extent of those complex nervous systems. In order to simulate some of these stimuli-response processes of simple organisms, we can look to Boolean networks to represent and study them in an artificial space (Figure 1). Each network consists of a set of nodes representing either the values '0' or '1'. Each node has a varying number of connections that link itself to other nodes and vice versa. Each node then uses those nodes along with a specified rule to calculate what the new value stored in that node will be for the next time step. This parallels the way neurons connect to each other within a nervous system. The neurons store and send chemical signals to one another across the synapses of the connections between them. The nodes in a Boolean network can be set up with designated "input" and "output" nodes to represent an organisms sensory receptors and reaction to the given stimuli.

In nature, organisms are required to learn and adapt in their environment if they want to survive and pass their genes on to the next generation. In this process, organisms that have genetic traits which will make them better fitted




| INPUT | | | OUTPUT | | |
|---|---|---|---|---|---|
| A | B | C | A' | B' | C' |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |

Figure 1: Example of a Boolean network with multiple representations.

for survival, can more easily pass those traits on to their offspring. In the process of evolution, whether these genes are a result from a mutation or from reproduction, favorable traits tend to stay in the population more often than less favorable traits. This biological phenomena can be used as an inspiration for computer simulations. An algorithm can be used to decide which members of a population perform better and improve those members for future generations. This algorithm can be applied to work on Boolean networks by observing how they would react when given the same set of inputs and initial conditions (representative of an organisms environment in nature). These networks can then be modified based on traits observed in better performing networks in hopes of creating a better suited network to fit the given problem.

## Genetic Learning Algorithm Application

To start working with a genetic learning algorithm, a population of artificial organisms needs to be generated with a way to distinguish different members of the population. Given that a Boolean network relies on binary values and Boolean functions, a simple string of binary digits could work to describe any number of networks and act as a stand in "genotype" for this simulation. In biology, the genetic information in DNA/RNA strands consist of 4 different nucleotides. These can be used to represent $4^n$ different combinations given a string of *n* nucleotides. The simulation can use these generated binary strings of *n* digits to represent $2^n$ different networks. This will be sufficient enough to represent any possible network given enough bits, therefore it will

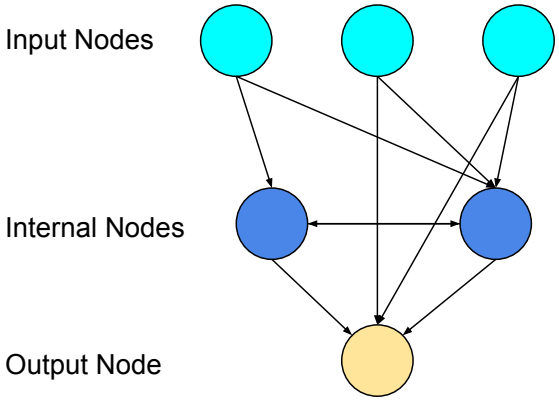



Figure 2: An example Boolean network with 3 input nodes, 2 internal nodes, and an output node.

provide enough genetic diversity to allow for large populations of distinctly different artificial organisms.

Based on the desired amount of nodes in the network, the population of genotypes is generated with binary strings with a length *l* based on the following equation:

$$int = NumberOfInternalNodes$$

$$input = NumberOfInputNodes$$

$$l = (3 + input + int) * (1 + int)$$

This guarantees that the genotype can be split into equally sized "genes" and be used to represent the calculations needed to evaluate the network at each time step. There will be one gene per internal node and one for the output node (1+int). This is because the input nodes are reserved for specific inputs while testing and are representative of the current environmental stimuli. Each gene consists of 3 bits, which are converted into a Boolean function (Figure 3), and the rest of the bits are indicative of each of the non-output nodes. These nodes are taken as inputs when performing the function indicated by the first 3 bits (Figure 4). By following these rules set by the genotype, the Boolean network acts as the phenotype of the organism which can be observed and studied based on its behavior. Each Boolean network can have the same amount of nodes but differ in the internal connections and the inherent function designated to each node. This will provide a differing range of performance in a population which makes the networks function as needed when placed in an environment for genetic learning.

For this experiment, each member of the population was graded based on its ability to discern whether a given stimulus represented a "dangerous" environment or a "safe" environment. In terms of the Boolean networks, they would need to successfully discern whether a given set of binary inputs contained an equal or greater number of "1"s than "0"s. This represents a "dangerous" signal and requires a

| Operation Table | |
|---|---|
| Code | Operation |
| 000 | "0" |
| 001 | OR |
| 010 | XOR |
| 011 | AND |
| 100 | "1" |
| 101 | NOR |
| 110 | XNOR |
| 111 | NAND |

Figure 3: Each 3 bit operation code used to calculate the next value of the internal nodes and output node.

Gene: $[b_0 b_1 b_2 b_3 \ldots b_{n-1}]$
Operation Code
Input Node Indication

Genotype: $[g_0 g_1 \ldots g_i]$

Figure 4: Individual components of a genotype.

$$n = 3 + InputNodes + InternalNodes$$

$$i = InternalNodes$$

response of the output node to be "1" or it will fail for that specific environment. For a biological example, this could be representative of chemical or temperature sensors around a single-celled organism. If the cell were to detect a temperature or chemical concentration that would threaten its survival, it could trigger a response that would help increase its chances of survival. Identification of an organisms environment is the behavior that this experiment is trying to simulate. In addition, the experiment seeks observe how behaviors like this may evolve within various organisms.

As a way to simulate the effects of natural selection on generations of a population, a metric for success needs to be created so members of the population can be directly compared to one another. This is why a fitness function which analyzes a member's response stimuli must be developed and used as a standardized measurement for success. This will help decide which genotypes should continue on to the next generation of the population. The function uses the binary network that is created from a genotype and exposes it

in all possible environments and all possible starting states for the internal nodes. In total, this can represent $p$ possible starting states as described by the following equation:

$$p = 2^{int+input}$$

The output of the network is evaluated at each time step to see if it matches the expected result based on the input values. If an output remains constant for specific number of steps, then it will be considered a success. This genotype will be given the minimum score for that set of inputs (with lower scores being preferred to higher scores). If a Boolean network is not able to put together a string of successful outputs by the predetermined stop time, it will be considered a fail. It is given the max number of steps as its score for that set of inputs. After all sets are tested on a specific Boolean network, that genotype is assigned a fitness value based on the sum on its scores for each set. This will allow the genotype to be ranked among all the genotypes within that generation of the population.

Once all the fitness values are calculated, the population then goes through a set of steps to prepare a new population for the next generation. The idea is that the better performing genotypes will be kept and slightly modified in an attempt to improve the overall fitness of the population.

The first step in this process is to set aside the top performers so that their genotypes are not lost between generations. This guarantees that the top performer of each generation will be at least as good as the previous generation's top performer.

The next step is to decide on which genotypes will serve as the "parents" for the next generation. Instead of generating an entirely new population of random genotypes, this allows for specific traits to be passed down through the generations. This process also favors traits in to combine traits that help a genotype to succeed to produce a better performing genotype than each of its parents. Each parent is selected at random with the selection being weighted based on each genotype's fitness score.

Once parents are selected, they have a chance to either directly pass down their own genotypes to their offspring, or "reproduce" with each other to provide two different genotypes. This reproduction step acts similarly to the process of genetic recombination in sexually reproducing organisms. A break point in the genotype is randomly decided and the genes that remains on one side of that break point will be switched with the other to create the two new genotypes for the next generation. An example of this can be seen in (Figure 5). Each genotype is separated into distinct genes with an operation code and a set of input nodes. It is for this reason why the chosen break point will be made to fall either between genes or between the operation code and the input node indicators in the same gene. This is an attempt to not break apart crucial functions that may be the reason for a given genotype's success.

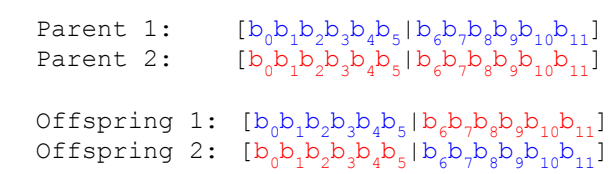



Figure 5: Two parent genotypes being recombined to generate two offspring genotypes.

Finally, each offspring is put through a process which gives a random chance for each bit in its genotype to undergo a mutation, which in this case would take the form of a bit flip from 1 to 0 or 0 to 1. This naturally happens in nature for various reasons and allows for advantageous mutations to arise and spread throughout a population. If this step was not utilized in the simulation, it would be possible for sub-optimal genes to remain in the population with no new adaptations emerging.

## Experimental Results

When determining the success of a population, the best performing genotype is chosen to represent the population. The percent of correct detections is used as the metric for comparison between those of the previous and future generations. The x-axis is indicative of the number of generations that have passed since the start of the experiment and the y-axis represents the correct detection rate of the best performer. The resulting graphs from the experiment show different trends based on the initial variables chosen. After averaging the simulations of multiple populations, a few trends tend to pop up regardless of these initial variables:

- There is a constant upward trend in detection rate because the best performing genotypes were automatically passed down to the next generation of the population. Therefore the only difference in one generation's best performing genotype from the previous would be a fitness increase.
- Long stretches without an increases in performance were likely to occur throughout generations. This indicates that no genotypes in the current population improved beyond the standard set by the best performer in the previous generation. If the best performer in a population was a different genotype every generation, the experiment would only need to be run for a few generations and it would find a genotype with a success rate of 1.0 every time.
- Similar to the previous trend, the majority of improvements in the population occur within the earlier generations since the chance of a population improving upon the previous generation decreases as the performance of the population as a whole increases. This is because the fitness of the population is capped at a detection rate of 1.0

and the room to improve shrinks as the generations improve. This means that the highest likelihood of observing an improvement between generations occurs between the $1^{st}$ and $2^{nd}$ generations. The least likelihood of observing an improvement occurs between the penultimate and the last generations.

- For the purpose of this experiment the Boolean networks that get created have only one output node. Therefore if you took any randomly generated genotype in the first generation, you'd expect it to provide a correct detection roughly half the time. Now with a large population, the odds of having the best performing genotype with a detection rate of roughly 0.5 is highly unlikely. All of the experiments, as long as the population wasn't too small, would have a starting success rate significantly higher than 0.5.

When preparing to run this experiment, a variety of variables need to be established and changing any of them can cause a noticeable difference in the final results (numbers in parenthesis are used in the control) (Figure 6):

- **Population Size (100)**
- **Number of Generations (100)**
- **Number of Experiments (10)**
- **Number of Input/Internal Nodes (3/5)**
- **Recombination Rate (0.2):** Rate at which the genotypes of selected parents are recombined to create the genotypes for their offspring. Otherwise the offspring will have the same genotype as their parents.
- **Mutation Rate (0.1):** Rate at which each bit of an offspring's new genotype is flipped.
- **Keep Rate (0.1):** Top percent of the population that will be automatically passed on to the next generation.
- **Correct Output Length (5):** Number of time steps with consecutive correct outputs by a Boolean network to be considered a correct detection. This is to prevent outputs of Boolean networks with oscillations or limit cycles to also be considered correct. It could be the case that they only had a correct detection for a short amount of time.
- **Total Output Length (20):** Number of recorded time steps of a Boolean network before it is deemed unsuccessful. This gives the network enough time to process the inputs. A large Boolean network could use a feedback mechanism that requires a few steps before it displays its final output.

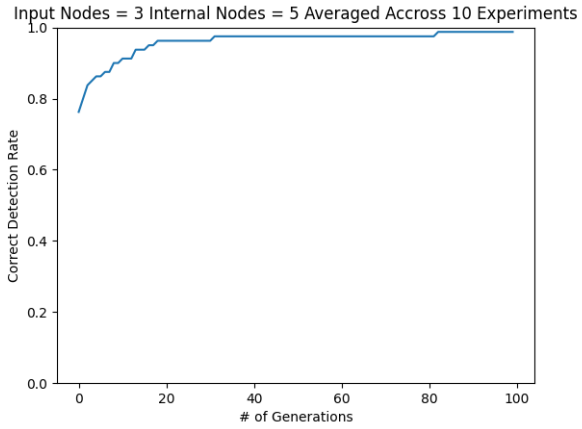



Figure 6: Results of the genetic learning algorithm when using the control variables.

## Population - Figures 7 and 8

An increase in the population size will allow for more genetic variety at the start as well in the genotypes of the offspring in subsequent generations. Not only is a "perfect" genotype (one having a correct detection rate of 1.0) found quicker, but the population at the beginning has a jump start of around a 0.9 success rate. Something that should be noted is that out of the 10 times a 1000-member population was used, all 10 found a "perfect" genotype within about 15 generations. This shows that the success rate of the best performing genotype in the first generation approaches 1.0 as the population increases. There will be a point where the population is high enough that it becomes statistically improbable that at least one genotype does not produce a Boolean network which has a success rate of 1.0. This can be attributed to the finite nature of Boolean networks and that there can be multiple "perfect" genotypes for a given configuration.

## Internal Node Count - Figures 9 and 10

Setting up the Boolean networks to have less internal nodes will simplify both the network connections and the genotype required to create the network. This shrinks the possible genotypes by a factor of $2^n$ for every $n$ nodes removed. This would then increase the chances of finding the better performing genotypes. However, it can be speculated that there could be less "top performing" genotypes available because of the lack of computational variety that comes with removing nodes from a Boolean network. There are less nodes and also less functions that can be used for finding a correct output for the given input.

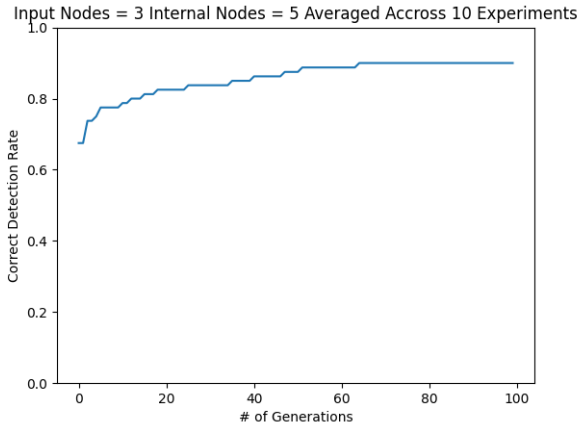



Figure 7: Results of decreasing the population size to 10.

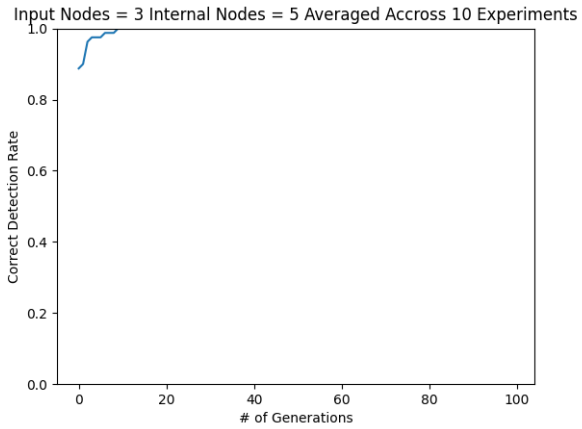



Figure 8: Results of increasing the population size to 1000.

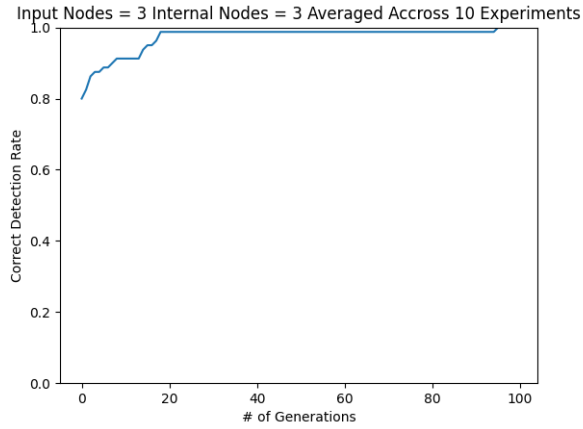



Figure 9: Results of decreasing the internal node count to 3.

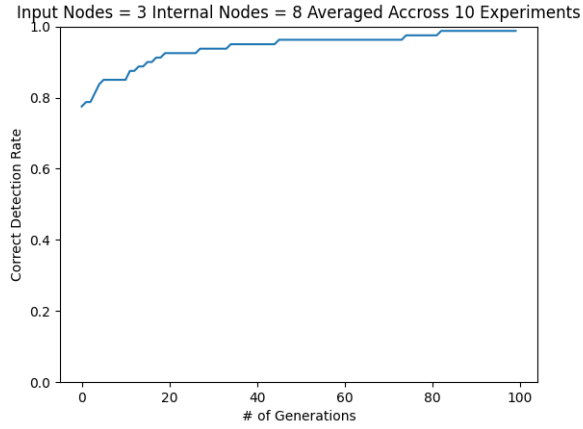



Figure 10: Results of increasing the internal node count to 8.

## Input Node Count - Figures 11 and 12

The number of input nodes seems to be the most sensitive value when it comes to the result. An increase in input nodes calls for an increase in test cases for the Boolean network to be tested with. This gives the network more opportunities to fail and less of a chance for any given network to get closer than the current top performing network to a 1.0 successful detection rate. For these sets of experiments, the performance for the 5-input networks averaged just over a 0.8 detection rate. On the other hand, decreasing the number of input nodes from 3 to 2 drastically increased the speed at which the algorithm was able to find a "perfect" genotype. With only 4 different inputs to worry about, a network can realistically be made up of 2 input nodes and 1 output node which calculates the OR of the two input nodes. This network has a detection rate of 1.0 through testing. Having 5 internal nodes leaves a lot of redundant nodes in the network and the algorithm can stumble into a "perfect" genotype within a few generations.

## Recombination Rate - Figures 13 and 14

The only trend discernible from changes in the recombination rate is that a "perfect" genotype was found noticeably faster when the recombination rate was increased. These experiments favored previously successful generations and attempted to combine parts of their genotypes that were successful. With a lower recombination rate, breakthroughs will happen through mutation more often than normal since the parents' genotypes will be passed down without many changes through the generations.

## Mutation Rate - Figures 15 and 16

Along with recombination, mutation is the other main factor when it comes into generating new offspring. They are

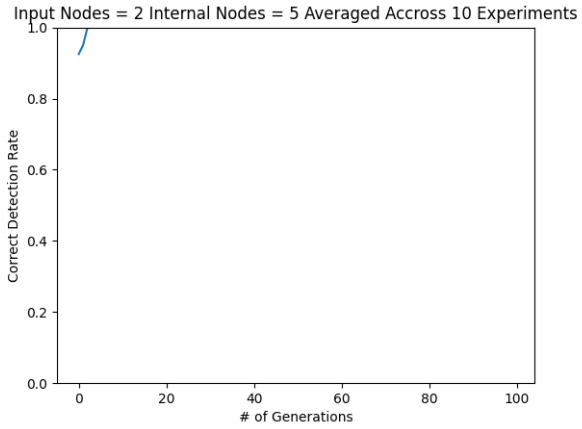



Figure 11: Results of decreasing the input node count to 2.

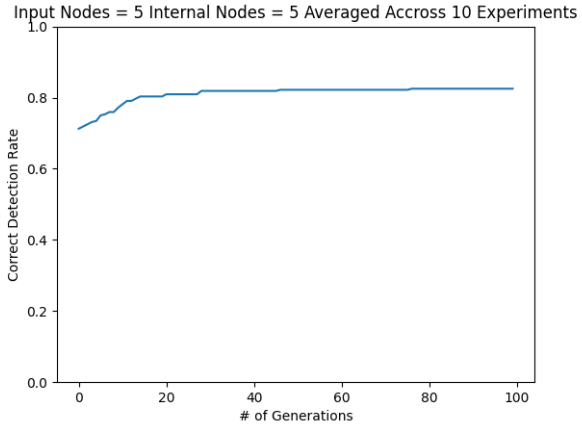



Figure 12: Results of increasing the input node count to 5.

both the main reasons why genetic breakthroughs happen throughout the generations. However, if the mutation rate was set to 0.0, the population would get stuck with only the genes they started with. Mutation rate acts similarly to recombination rate but in reverse. When mutation rate is increased, breakthroughs are more reliant on the mutation of offspring genotypes instead of recombination. Having a 0.5 mutation rate is almost like having an entirely new offspring with little genetic similarity to each of the parents. Relying on random chance rather than previous success does hinder the success of the population, and it shows that it will take longer to find one of the "perfect" genotypes.

## Keep Rate - Figures 17 and 18

The keep rate, for the purpose of this experiment, is very helpful when it comes to having consistent results. This is the sole reason why the plotted correct detection rates form

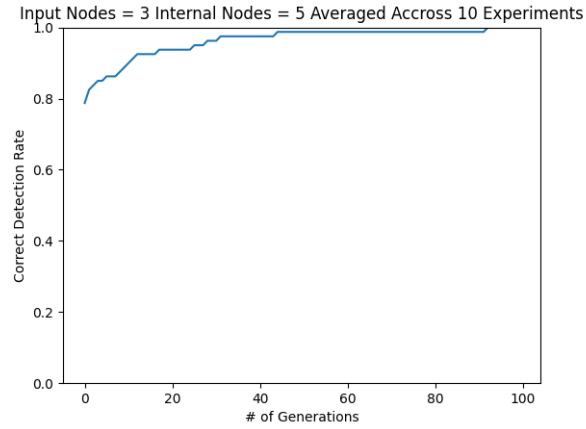



Figure 13: Results of decreasing the recombination rate to 0.1.

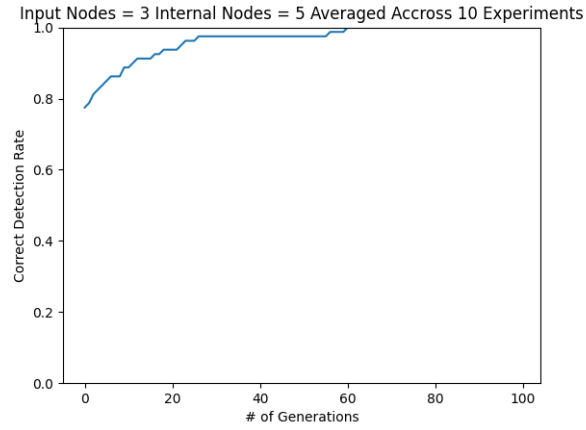



Figure 14: Results of increasing the recombination rate to 0.6.

an increasing function. It allows the population to start off with a certain number of genotypes that have already proven to be successful to some extent. This allows for desirable traits to stay within the gene pool and be used for recombination and mutation in the future generations. Reducing the amount of genotypes kept for the next generation will allow for more variety among the remaining genotypes. This also removes some of the better performing genotypes that still have desirable traits. Between the two results which vary in keep rate, it is interesting to note that the higher rate results outperformed the lower rate results in the beginning. However, the lower rate results show that a "perfect" genotype was found in all 10 experiments by around generation 70. The higher rate results show that the algorithm was not able to succeed on every experiment by the end of the 100 generations. This could be due to the low number of exper-

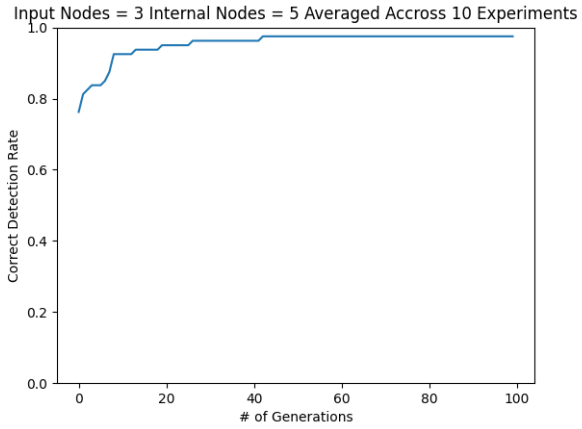



Figure 15: Results of decreasing the mutation rate to 0.05.

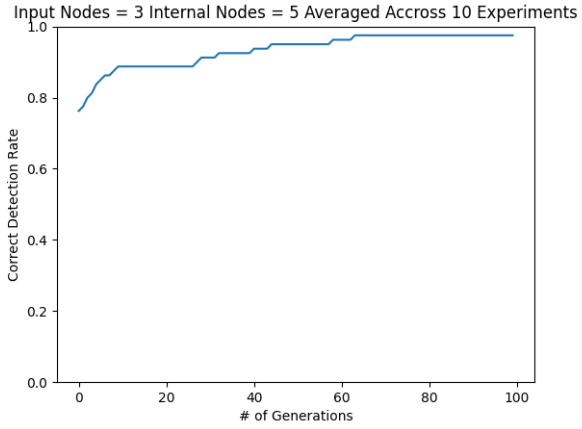



Figure 16: Results of increasing the mutation rate to 0.5.

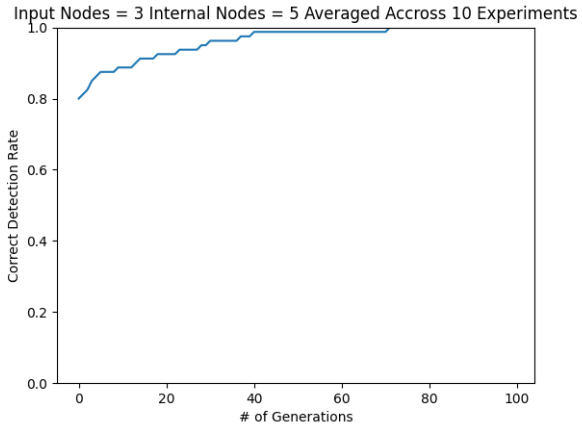



Figure 17: Results of decreasing the keep rate to 0.05.

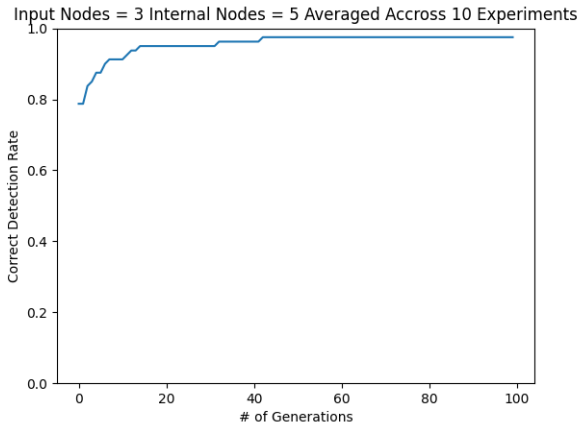



Figure 18: Results of increasing the keep rate to 0.3.

iments which would make an outlier more noticeable. This observation calls for further experimentation to confirm the findings.

## Future Work

Creating this algorithm was only a start in what could be a larger and more exploratory project. Further research could be conducted on specific organisms whose nervous systems closely resembles a Boolean network and use that as a starting point. The computational complexity for Boolean networks starts to get out of hand when adding only a few additional nodes to the system. This is one main reason that a revision of the genetic learning algorithm would need to take place if any further work is performed using the algorithm. There are many ways in which the code could have its computational complexity reduced. This would allow for larger Boolean networks to be observed in a genetic learning environment without needing as many computational resources and time to complete the experiments. These changes would also allow for an increase in the number of experiments that can be run for a given set of starting parameters.

There is also the idea of considering the substitution of the Boolean network for a neural network. One main difference being the increased variety of values and computations.The current algorithm would need to be heavily modified, due to its discrete nature, to incorporate a network of this complexity. Slight adjustments to genetic values would be allowed in this type of algorithm and could produce smoother results in the final graph as opposed to the current ones which involve sharp jumps in performance between generations.

During the creation of the algorithm there were assumptions made that could be examined and tested further. For example the recombination of parent genotypes into offspring genotypes can be turned into a more intricate func-

tion. In such a function, multiple slice points can be made to see if it has an affect on the population trends. The parent selection metric could also be adjusted to exclusively consider those that perform in the top percent of the population instead of selecting from the entire population. In addition, the Boolean networks themselves can be compared to Boolean networks of differing sizes. There could be a penalty set in place for having a network that is more complex than necessary. This would help narrow down how complex a network really needs to be to perform a specific function. Studying the effect of varying the mathematical constants for each experiment can help in future experiments with more complex networks using this model.