UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS700 Research Report

A Study of Genetic Operations for Automated Design

Student number: u15323413

Supervisor:
Dr. Nelishia Pillay

October 2021

Abstract

The automated design of Neural Networks is an active area of research in the field of Computer Science, specifically Neuroevolution, which uses evolutionary techniques to evolve a neural network. However, the field of evolutionary techniques is even larger than Neuroevolution. As such, there is a lack of study into the effect certain genetic operations have on Neuroevolution. An in-depth study and comparison between operators can be of value as it may help grow the area of study and provide other researchers valuable information. This paper aims to, initially, explore in-depth the effect of various genetic operations used in a genetic algorithm for automated design. Finally, a comparison of the various genetic operators will be performed.

Keywords:

Automated Design, Genetic Algorithms, Genetic Operations, Neuro-evolution.

1 Introduction

Artificial Intelligence is an extremely valuable asset in the modern day and age. With the huge amount of data available to us as we live in the Information Age, as well as how busy the daily life of the modern human is, it is no wonder that we need ways of offloading the computation of tasks that require a lot of brain power. When attempting to create intelligent systems, nature is one of the best places to draw inspiration from, as nature itself has already solved the problem of intelligence over millions of years. Now that we have hardware powerful enough to run complex computation, we are capable of simulating processes that occur in nature.

One of the earliest inspirations we achieved was in the form of Evolutionary Algorithms [1]. This technique attempts to mimic the way evolution works in the natural world, by evolving a population of potential solutions to a problem, made up of tailored 'genes', through generations. Each generation attempts to acquire the best 'genes' from the previous one, until a solution is found.

Artificial Neural Networks (ANNs) are another field of study derived from natural systems, where an attempt is made to mimic the way neurons operate within natural creatures. The modern technique of ANNs through the back-propagation of errors through layers of connected neurons was popularized in 1986 [2] and has since been used across many domains to solve countless machine learning problems. However, there was no way to discern an optimal topology (the layout of neurons in a neural network) of these networks by means of an algorithm [3] until Neuroevolution.

Neuroevolution attempts to combine the two previous methods of AI, and is defined as the process of altering aspects of an ANN through the means of evolutionary algorithms [3]. This can help to find a way of achieving an optimal topology for an ANN amongst other things. The idea of evolving an ANN can be traced back all the way to the 1990s [4] and since then there has been extensive research performed on the topic [3] [5] [6] [7].

Throughout this research, various genetic operators have been used in the Evolutionary Algorithms used in Neuroevolution but there does not seem to be an analysis of these operators. This paper aims to produce such an analysis, which could prove valuable to researchers and anyone interested in the topic of Neuroevolution.

1.1 Layout

The paper is set up into the following sections:

Section 2 - Genetic Algorithms

This section gives an overview of the intricacies of the Genetic Algorithm method and how genetic operators are used within Genetic Algorithms.

Section 3 - Artificial Neural Networks

A brief overview of the workings of Artificial Neural Networks (ANNs) is given in this section for interested readers.

Section 4 - Neuroevolution

The concept of Neuroevolution is introduced in this section. Explaining how Genetic Algorithms are merged with ANNs to create better artificial intelligence opportunities as well as some methods for the implimentation of Neuroevolution

Section 5 - Methodology

This section gives an overview of the research methodology used within this paper.

Section 6 - Approach

An in-depth discussion of the experiment setup is given in this section.

Section 7 - Results and Discussion

This section contains the results of all the experiments performed using multiple types of genetic operators to evolve an ANN for classification of penguin species.

Section 8 - Conclusion and Future Work

Ideas for expansion on the work done in this paper are given in this section as well as a summary of the work done.

2 Genetic Algorithms

Streichart et al. describe genetic algorithms (GAs) as a branch of Evolutionary Programming that "exhibit the clearest mapping from the natural process of evolution onto a computer system, because they stress the coding of attributes into a set of genes."[8] The general process of an evolutionary programming is taken from natural evolution and involves a population of elements that, over generations, are evolved by calculating a fitness value for each element, selecting the strongest elements of a generation and using them to create a new generation through the application of various operators. This process continues until a solution to the given problem is found. *Figure 1* shows this process.

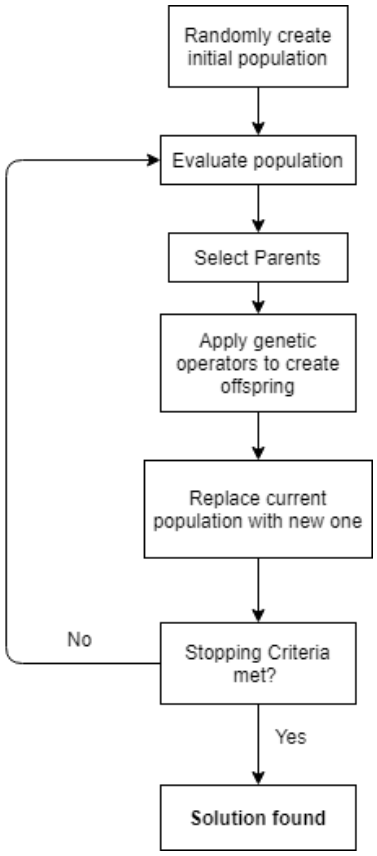

Figure 1: General EA Process

A Genetic Algorithm can be viewed as a multi-point search where the search space is a set of 'genes' referred to as a 'chromosome'. These 'genes' represent

the parts of the domain that are to be modified and are altered across generations through means of genetic operators. A solution is a 'chromosome' which contains a set of genes that solve the problem.

2.1 Genetic Operators

Genetic Operators are any operators used to modify the genes contained within a chromosome in Genetic Programming, therefore traversing the search space. Arguably the two main types of Genetic operators are crossover, where genes are swapped between parents to create offspring, and mutation, where genes are modified in a parent to create offspring. It is important to note that there are different variations in how Crossover and Mutation are applied, this is what this paper focuses on.

2.1.1 Crossover

Crossover aims to acquire successful genes from multiple elements of the population. Two of the strongest elements in the population are selected as parents for breeding. Genes are exchanged between the parents to produce offspring in hopes that over multiple generations, the strongest genes will remain in the population. Streichart et al. state that the operator "exchanges genes between the parents to generate the descendants."[8] This process is shown in *Figure 2*.

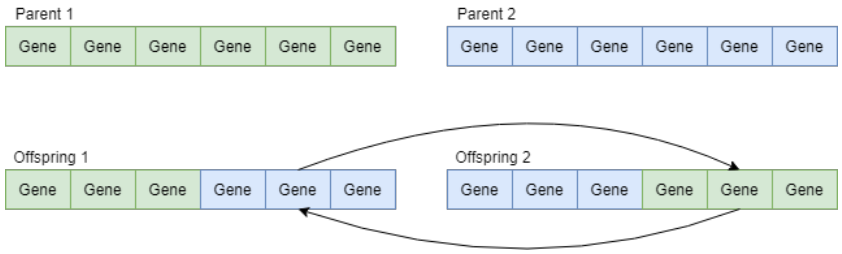

Figure 2: Crossover

2.1.2 Crossover Operators

Much like mutation, there are various methods of applying crossover to a population, one example is one point crossover, where a point in the chromosome is chosen and all genes to the right of that point will be crossed over, this is shown in *Figure 3*.

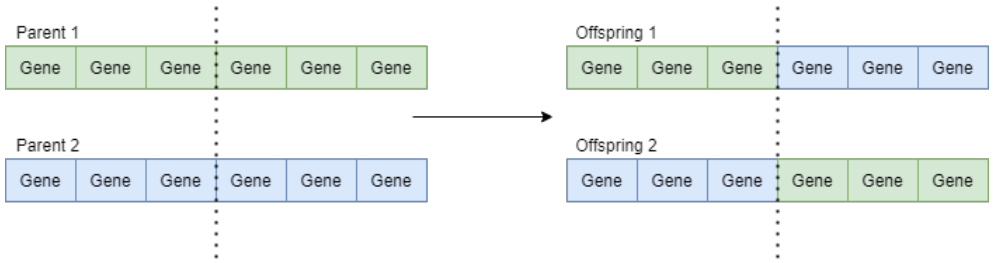

Figure 3: Single Point Crossover

An operator similar to one point crossover is two point crossover, where two points are randomly selected instead of one, and the genes between these two points are crossed over, see *Figure 4*.

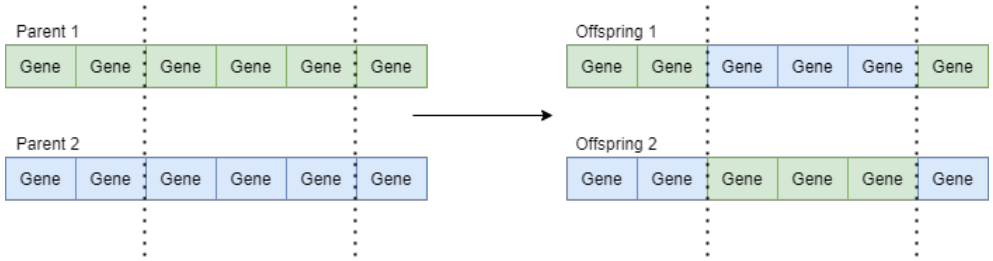

Figure 4: Two Point Crossover

Crossover can also be performed with a high level of randomness involved, where a percentage of genes are randomly chosen across the chromosome and swapped between the parents chosen for mating. *Figure 5* demonstrates this.

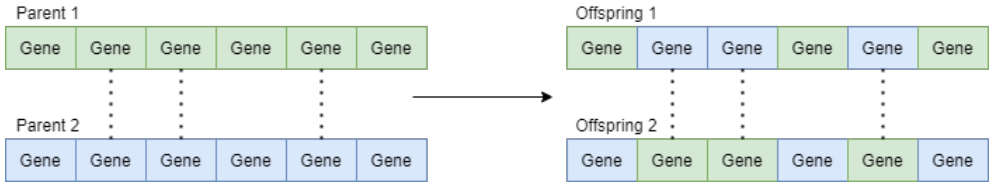

Figure 5: Random Crossover

2.1.3 Mutation

Mutation aims to change one or more genes of a parent randomly, introducing new combinations of genes into a population, expanding the search space. It "resembles the naturally occurring accidents that can happen when DNA is copied."[8] This is shown in *Figure 6*.

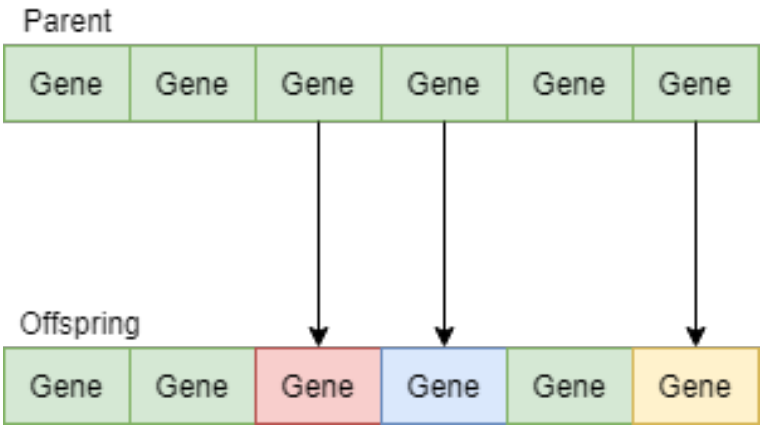

Figure 6: Mutation

2.1.4 Mutation Operators

Mutation operators all act on a single parent, and involve randomly changing the chromosome of the selected parent in some way. Some of the common mutation operators found across papers were: single-gene mutation, where a single gene is selected to be mutated, shown in *Figure 7*.

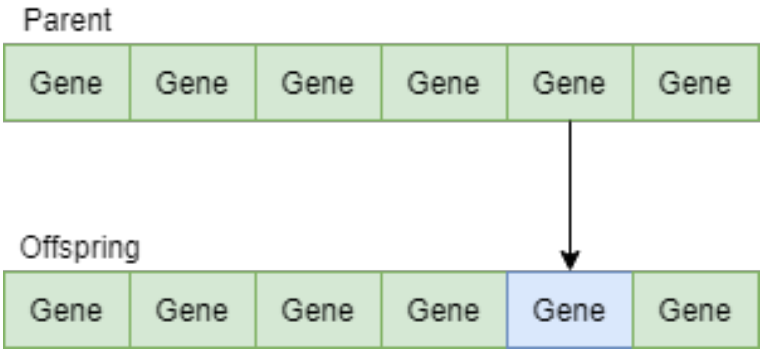

Figure 7: Single Gene Mutation

A slightly more advanced operator selects a gene for mutation, this gene and it's neighbouring genes are then mutated. This is shown in *Figure 8*

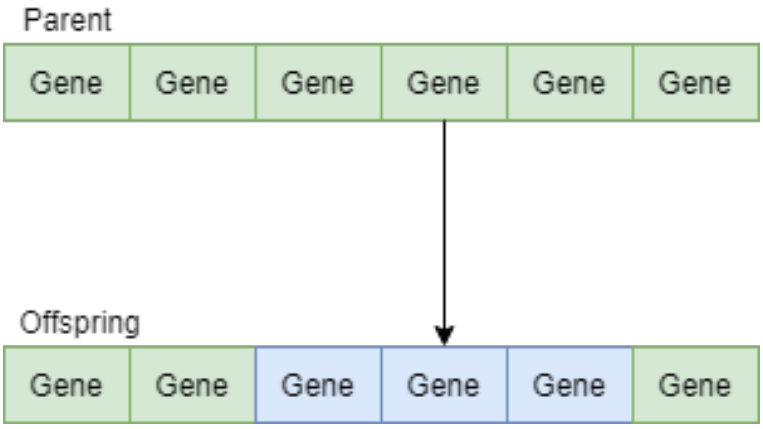

Figure 8: Neighbouring Gene Mutation

Another form of mutation is multiple gene mutation, where a percentage of genes are randomly selected for mutation, shown in *Figure 9*.

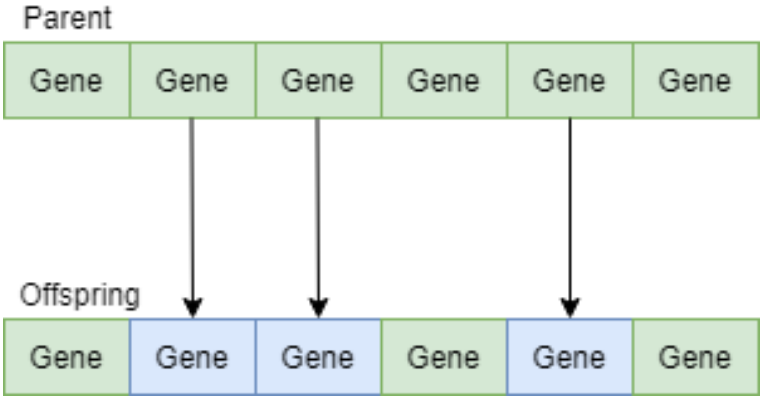

Figure 9: Multiple Gene Mutation

2.2 Genetic Operator Evaluation

As the number of Genetic Operators available increases, it is often not clear which ones will be best suited for the task being worked on. Takahashi et

al. attempted to create a method for the evaluation of Genetic Operators.[9] During their research, they stated "In most cases, there is not any analytical justification for the choice of a specific operator structure"[9] and concluded that "The question of what are the best operators has shown to be more intricate, since the operators have been shown to be performance-dependent one to another."[9] This shows a need for the field of analysis of genetic operators, specifically to aid the decision on what operators will achieve the best results when applied to a specific problem scope.

3 Artificial Neural Networks

Another machine learning method derived from nature is that of Artificial Neural Networks. These networks take inspiration from "the distributed, massively parallel computation in the brain that enables it to be so successful at complex control and recognition and classification tasks."[10] They are composed of multiple layers of 'neurons' connected to each other by mathematical values called weights. These weights are what are changed during training in order for the network to make predictions. The neurons in the network "are almost always simple transcendental functions whose arguments are the weighted summation of the inputs to the node"[10] *Figure 10* illustrates this model.

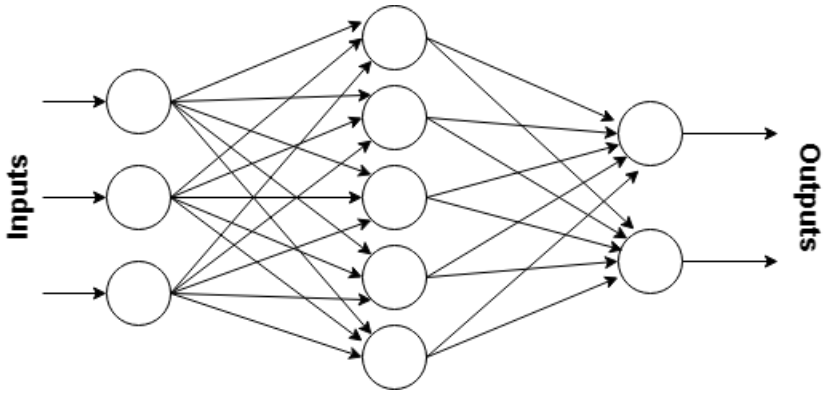

Figure 10: Representation of Neural Network

4 Neuroevolution

Baldominos et al. have extensively researched the history of Neuroevolution (NE), starting from its beginnings in 1989 where "early works in NE are concerned with evolving the weights of ANNs"[3] as well as some research into parameter tuning such as the learning rate of a network, rather than attempting to evolve their topologies. The idea for automated topology design of an ANN has been around since 1989 with some early research.[11]

In the present, advanced neural networks are being used to solve a wide range of machine learning problems. As a result of the advanced hardware we have today, Baldominos et al. state that "The recent emergence of deep and convolutional neural networks has brought back the need for designing topologies that are suitable to tackle specific problems"[3]. This poses a new challenge for NE as "DNNs and CNNs can have hundreds of thousands or even millions of weights, and innovations must be introduced in NE for it to adapt to these new topologies."[3]

Modern NE still generally follows the general process of Evolutionary Algorithms, however each element of the population represents a different network topology or a network with different learning parameters, or a combination of both. As we know from section 3.1, each element of the population must have their fitness calculated individually, therefore each network will have to be trained and evaluated, which is computationally expensive. *Figure 11* shows this process.

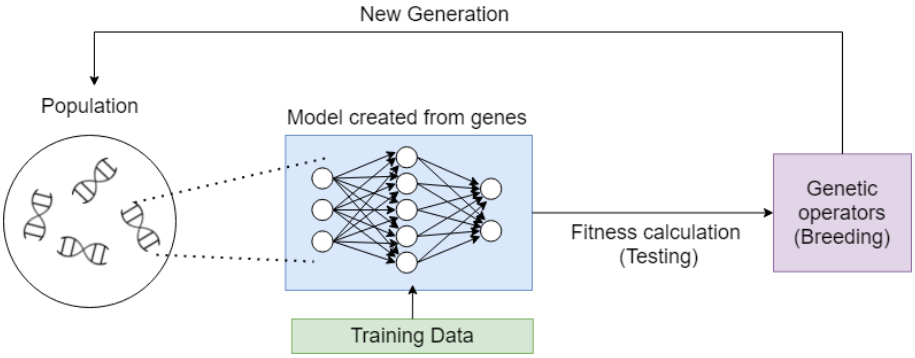

Figure 11: Evolution of Neural Networks

Neuroevolution is still being actively studied, and will continue to be necessary as the field of ANNs grows. Floreano et al. predicted in 2008 that

"neuroevolution will continue to provide efficient solutions to hard problems as well as new insights into how biological brains became what they are in physical bodies interacting with dynamic environments."[7] and recently, in 2019, Stanley et al. stated "Just as evolution did in the natural evolution of intelligence, we expect that neuroevolution will play an important role in our collective quest to create human-level AI and algorithms that endlessly innovate." [6] and 2020, Baldominos et al. ended their paper by stating "Much work is still to be done, but given the large applicability of CNNs and their success, the automatic evolutionary design of their topologies is a very promising area which must be tackled as of today."

Given the above, an in-depth analysis of genetic operations used in the field of NE could prove valuable to assist in further research.

4.1 Chromosomes in Neuroevolution

One of the most commonly used representations of a chromosome for the architecture of a Neural Network is a binary string. [12] [13] [14] [15] [16] [17] [18] [19] [20]
This string is a series of binary characters with different segments of the binary string representing different parts of an ANNs architecture. The string can represent basic parts of the architecture, Arifovic et al. used a string that 'consists of *lchrom* bits. The *lchrom* bits are divided into three parts. The first part of length *lw* is used to encode the initial weight range. The second part of length *lchroi* is used to encode what inputs will be used and the third part of length *lh* is used to encode the number of hidden units.'[12] However, the binary string may also be more complex, such as bit strings of length 42 [17] and of length 77 [16]

A similar representation include using the string to represent a connectivity matrix [14] [21] [22], where the string is turned into a two-dimensional matrix representing the connectivity structure of nodes in a neural network.

A representation similar to a binary string is an integer string[23], which follows the same concept of a binary string but is shorter due to characters represented as base-10 integers rather than binary digits. Hybrid representations, that use both binary and integer strings in their chromosome exist. [24]

Finally, the highest level of representation used is one where each gene in a chromosome represents an individual variable[25], each with it's own cat-

egories of allowed options, or restrictions on values placed in them. Goni et al. used variables that represented 'the network parameters, i.e. the connection weights and biases of the network nodes, the moment coefficient and the learning rate, the last two with the restriction of being positive values.'[25]

5 Methodology

5.1 Research Methodology

The methodology used for this research will be a 'Proof by Demonstration', described by Chris Johnson as research that aims "to build something and then let that artefact stand as an example for a more general class of solutions."[26] This approach differs from other Methodologies as "it is often the case that this approach ignores the formation of any clear hypothesis or conclusion until after the artefact is built."[26] Using this method, the operators being studied will be recreated and tested. Proof by Demonstration focuses on refinement of the operators used, meaning that initial results are used in later attempts to try and produce better results until a satisfactory result is achieved.

5.2 Metrics

In order to measure the performance of the various genetic operators being experimented upon, metrics that measure how effectively an operator creates children from one generation to the next need to be used. Therefore, the chosen metrics for the experiment are as follows:

- Improvement chance: The chance an operator has to improve the ANN model from one generation to the next.
- Improvement amount: When there is an improvement across a generation, how much the model's accuracy improved is recorded.
- Final Accuracy: This metric will help to determine the overall success of an operator.

5.3 Choice of Data-set

As this study relies on building and training a large amount of artificial neural networks, a small dataset was chosen to ensure that enough experiments

could be run. It was also necessary to ensure that high categorical accuracy would be possible, as such it was important to find a data set that is fully separable. Taking the above into consideration, the Palmer Penguins[27] dataset was chosen. This data set contains various attributes of three distinct penguin species, namely: Chinstrap, Gentoo and Adelie. This dataset consists of 5 values per entry:

- Bill length, in mm
- Bill depth, in mm
- Flipper length, in mm
- Body mass, in g
- Species of penguin

The ANNs created will be trained on this data to classify the species of the penguin based on the 4 features given.

5.4 Data Pre-processing

As we are using an ANN for classification, the species will be one-hot encoded and the ANN given 3 output nodes. One hot-encoding is the process where each class is represented as an array with the same length as there are number of classes, where each value is 0, except for the position the class represents, which is changed to a one. In this case, the penguin species are modified as follows:

- Chinstrap –>[1, 0, 0]
- Gentoo –>[0, 1, 0]
- Adelie –>[0, 0, 1]

The data was then spilt into a training and testing set, with a ratio of 9:1. i.e 90% of the data was used for training purposes and the remaining 10% was used for testing the trained models.

5.5 Representation

In this paper, the architecture of an ANN will be created using a chromosome consisting of multiple variables. The chromosome will consist of, the

activation function used for hidden and output neurons, the optimization algorithm used for training, the number of nodes per hidden layer, the number of hidden layers, the inclusion of dropout, and lastly the number of epochs that the network will train for. The different possibilities for each gene are displayed in the following table:

Variable	Possible Parameters
Activation	{relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exp}
Optimization	{sgd, rmsprop, adam, adadelta, adagrad, adamax, nadam, ftrl}
Hidden Neurons	{3-20}
Hidden Layers	{0, 1, 2}
Dropout	{True, False}
Epochs	{5-30}

6 Approach

6.1 GA parameters

The following parameters were used for the Genetic Algorithm responsible for applying the genetic operators to the ANN chromosomes:

Parameter	Value
Population Size	40
Parent Selection	Tournament Selection
Tournament Size	5
Mutation Rate	0.3
Crossover Rate	0.7
Termination Criteria	10 Generations

A fixed generation number was decided upon via empirical testing. It was found that the algorithm converged within 10 generations every time.

6.2 Experiments

As the aim of this paper is to analyze various genetic operators, a baseline experiment will be performed, followed by additional experiments with the genetic operators used being the only difference between experiments.
Each experiment will involve 10 runs of the Genetic Algorithm, each run with a unique number assigned as the seed for the random number generator used for the GA. Single-gene crossover will serve as the crossover operator used

for the baseline experiment, where only a single gene is randomly selected to be swapped between parents. Likewise, single gene mutation will be used as the baseline mutation operator, where a single gene is randomly selected for mutation.

Once the baseline experiment is completed, two additional experiments will be conducted for crossover operators, one where the crossover operator used will be single point crossover, and the other using random crossover, both explained in *Section 2.1.2*.
A final two experiments will be conducted where the mutation operator now becomes the subject of interest, the two additional mutation operators experimented with are neighbouring gene mutation and multiple gene mutation, both explained in *2.1.4*.

7 Results

Every experiment conducted was successful in producing a network that achieved a prediction accuracy greater than 95%, with majority of experiments producing trained models that achieved a prediction accuracy of greater than 99% . A breakdown of the experiments and the resulting analysis are given below.

7.1 Baseline

To show the effectiveness of the Genetic Algorithm with the baseline operators mentioned in *Section 6.5*, a run was chosen to graph the improvement across generations. *Figure 12* takes the best performing model from each generation and shows the minimization of loss: As the loss of a model decreases, it's accuracy increases. *Figure 13* shows the accuracy of the best model from each generation, taken from the same experiment as the loss graph. The best chromosome after 10 generations from the same experiment, whose resulting model achieved a prediction accuracy of 99.6% is as follows:

Variable	Value
Activation Function	exponential
Optimization Algorithm	adam
Number of Hidden Neurons	16
Number of Hidden Layers	2
Dropout	False
Training Epochs	21

Figure 12: Loss change across generations

7.2 Crossover Results

After performing the crossover experiments, there was no noticeable difference in the rate at which the crossover operators were improving the children, however the general trend in the amount improved, when an operator did improve the result, the crossover operators that shared more genes between parents performed better with regards to the amount of improvement across a generation.
The results are as follows:

Operator:	Baseline	Single Point	Random
Average Chance for Improvement:	61.59%	59.63%	59.92%
Average Improvement Amount:	2.82%	3.72%	3.12%
Best Final Accuracy:	99.6%	100%	100%
Average Final Accuracy:	99.35%	99.26%	99.13%

7.3 Mutation Results

The mutation experiments show that using operators that mutate more than one gene of a chromosome have better improvements on accuracy over generations than operators that mutate less of a chromosome. Multiple Random mutations had a significalnlty higher chance to improve the population across a generation, however the average final accuracy after 10 generations was lower than experiments which used other mutation operators.

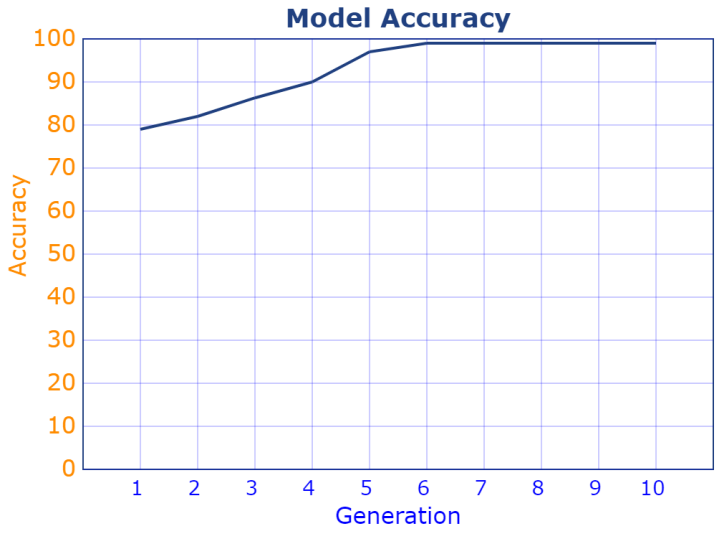

Figure 13: Accuracy change across generations

Operator:	Baseline	Neighbouring	Multiple Random
Average Chance for Improvement:	61.59%	54.84%	71.87%
Average Improvement Amount:	2.82%	4.47%	3.53%
Best Final Accuracy:	99.6%	99.6%	100%
Average Final Accuracy:	99.35%	99.20%	97.83%

7.4 Observations

It is worth noting that every operator used was able to produce great results in the GA, with every combination achieving a max accuracy of over 99% after 10 generations of application. The lowest average accuracy was achieved by the random gene mutation, this may be caused due to too many genes being mutated, moving the chromosome away from an optimum solution.
The crossover operators all performed well, showing that many forms of crossover are viable, as long as they are given a decently sized population for the application of these crossover operators.
Some of these results are dependant on the initial population created, as populations that were created with high accuracies to begin with would have less room for improvement. However, no initial population recorded an accuracy lower than 70%.

8 Conclusion

8.1 General Discussion

The aim of this study was to investigate and analyze the effectiveness of various Genetic Operators applied as part of a Genetic Algorithm created for the evolution of the architecture of an ANN for classification purposes. A study into their effectiveness will help the growing field of neuroevolution and assist researches in the choice for their genetic operators when creating Neuroevolution systems. Experiments were set up to evaluate various genetic operators. The chosen network was a fully connected ANN for data classification. The data chosen to be classified was data on penguin species in the palmer islands based on physical statistics captured by scientists. It was found that no significant difference occurred between different crossover operators, as long as there was enough of a population for crossover to operate on, they all performed well. Mutation operators showed the biggest difference in the workings of the Genetic Algorithm across generations. More mutation of genes was preferable, however too much mutation did not allow the generation to converge to the global optimum more often than other operators.

8.2 Future Work

This study leaves open the questions of whether or not the same observations will be present when these experiments are performed on a different data set or on more advanced ANNs such as Deep Neural Networks and Convolutional Neural Networks. A wider range of genetic operators could also be experimented on to widen the scope of the operators experimented upon in this paper.
Based on the results obtained, work could be done to develop new genetic operators that have the benefits of multiple operators together.

References

[1] T. Back and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[3] A. Baldominos, Y. Saez, and P. Isasi, "On the automated, evolutionary design of neural networks: past, present, and future," *Neural Computing and Applications*, vol. 32, no. 2, pp. 519–545, 2020.

[4] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[5] R. Miikkulainen, "Neuroevolution," 2010.

[6] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[7] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[8] F. Streichert, "Introduction to evolutionary algorithms," *paper to be presented Apr*, vol. 4, 2002.

[9] R. H. Takahashi, J. Vasconcelos, J. A. Ramírez, and L. Krahenbuhl, "A multiobjective methodology for evaluating genetic operators," *IEEE Transactions on Magnetics*, vol. 39, no. 3, pp. 1321–1324, 2003.

[10] M. H. Hassoun *et al.*, *Fundamentals of artificial neural networks*. MIT press, 1995.

[11] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms.," in *ICGA*, vol. 89, pp. 379–384, 1989.

[12] J. Arifovic and R. Gencay, "Using genetic algorithms to select architecture of a feedforward artificial neural network," *Physica A: Statistical mechanics and its applications*, vol. 289, no. 3-4, pp. 574–594, 2001.

[13] S. W. Stepniewski and A. J. Keane, "Topology design of feedforward neural networks by genetic algorithms," in *International Conference on Parallel Problem Solving from Nature*, pp. 771–780, Springer, 1996.

[14] D. Dasgupta and D. R. McGregor, "Designing application-specific neural networks using the structured genetic algorithm," in *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp. 87–96, IEEE, 1992.

[15] J. Tarigan, R. Diedan, Y. Suryana, *et al.*, "Plate recognition using backpropagation neural network and genetic algorithm," *Procedia Computer Science*, vol. 116, pp. 365–372, 2017.

[16] A. Bhandare and D. Kaur, "Designing convolutional neural network architecture using genetic algorithms," in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pp. 150–156, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[17] O. A. Abdalla, A. O. Elfaki, and Y. M. AlMurtadha, "Optimizing the multilayer feed-forward artificial neural networks architecture and training parameters using genetic algorithm," *International Journal of Computer Applications*, vol. 96, no. 10, pp. 42–48, 2014.

[18] H. Chung and K.-s. Shin, "Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction," *Neural Computing and Applications*, vol. 32, no. 12, pp. 7897–7914, 2020.

[19] D. Tian, J. Deng, G. Vinod, T. Santhosh, and H. Tawfik, "A constraint-based genetic algorithm for optimizing neural network architectures for detection of loss of coolant accidents of nuclear power plants," *Neurocomputing*, vol. 322, pp. 102–119, 2018.

[20] B. ul Islam, Z. Baharudin, M. Q. Raza, and P. Nallagownden, "Optimization of neural network architecture using genetic algorithm for load forecasting," in *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, pp. 1–6, IEEE, 2014.

[21] K.-m. Park, D. Shin, S.-d. Chi, *et al.*, "Variable chromosome genetic algorithm for structure learning in neural networks to imitate human brain," *Applied Sciences*, vol. 9, no. 15, p. 3176, 2019.

[22] M. Dam and D. N. Saraf, "Design of neural networks using genetic algorithm for on-line property estimation of crude fractionator products," *Computers & chemical engineering*, vol. 30, no. 4, pp. 722–729, 2006.

[23] D. Venkatesan, K. Kannan, and R. Saravanan, "A genetic algorithm-based artificial neural network model for the optimization of machining processes," *Neural Computing and Applications*, vol. 18, no. 2, pp. 135–140, 2009.

[24] N. Jiang, Z. Zhao, and L. Ren, "Design of structural modular neural networks with genetic algorithm," *Advances in Engineering Software*, vol. 34, no. 1, pp. 17–24, 2003.

[25] S. Goni, S. Oddone, J. Segura, R. Mascheroni, and V. Salvadori, "Prediction of foods freezing and thawing times: Artificial neural networks and genetic algorithm approach," *Journal of food engineering*, vol. 84, no. 1, pp. 164–178, 2008.

[26] C. Johnson, "What is research in computing science," *Computer Science Dept., Glasgow University. Electronic resource: http://www. dcs. gla. ac. uk/˜ johnson/teaching/research_skills/research. ht ml*, 2006.

[27] A. Horst, "Introduction to palmerpenguins." https://allisonhorst.github.io/palmerpenguins/articles/intro.html.