Neuro Evolution

Artificial Intelligence and Adaptive Behaviour Report

May 21, 2024

Nathaniel Bates

Table of Contents

[Introduction / Research 1](#_Toc167175295)

[*Overview of the Study 1*](#_Toc167175296)

[*Image recognition and Classification 2*](#_Toc167175297)

[*Evolutionary Algorithms Background 2*](#_Toc167175298)

[*Motivation for Using Evolutionary CNN 3*](#_Toc167175299)

[*Objectives of the Study 3*](#_Toc167175300)

[Hypothesis and Methods 4](#_Toc167175301)

[*Choice of Environment 4*](#_Toc167175302)

[*Image Pre-processing 5*](#_Toc167175303)

[*Agent Anatomy 6*](#_Toc167175304)

[*Sensors/Actions 6*](#_Toc167175305)

[*Brain Anatomy 6*](#_Toc167175306)

[*Adaptation Mechanisms 7*](#_Toc167175307)

[*Genetic Algorithm / CNN Parameters 7*](#_Toc167175308)

[*Fitness Measure 8*](#_Toc167175309)

[*Selection, Cross-Over, Mutation 8*](#_Toc167175310)

[*Research Topic 10*](#_Toc167175311)

[Results and Testing 10](#_Toc167175312)

[*Performance Evaluation 10*](#_Toc167175313)

[*Robustness and Generalisation 13*](#_Toc167175314)

[Discussion 14](#_Toc167175315)

[*Interpretation of Results 14*](#_Toc167175316)

[*Implications of Neuroevolutionary Algorithms 15*](#_Toc167175317)

[*Fail cases and Future Directions. 16*](#_Toc167175318)

[*Conclusion and Summary 17*](#_Toc167175319)

[References 18](#_Toc167175320)

[Appendix 18](#_Toc167175321)

# Introduction / Research

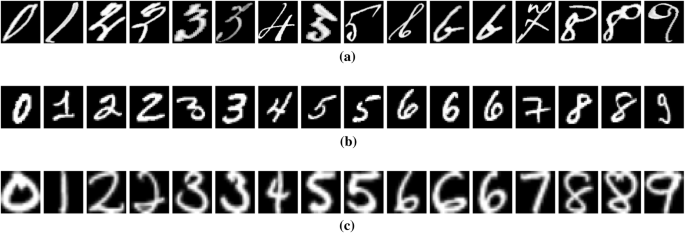
## Overview of the Study

My investigation will involve using an evolutionary genetic algorithm to design efficient neural networks for an image multi-classification task. I will be using convolutional neural networks.

I will be using python along with the **keras** library to design and test my neural networks along with **sklearn** library for data processing and image handling techniques.

## Image recognition and Classification

Image recognition is the simple task of classifying an image with a predicted label based on the pixel data. Binary classification involves only two classes, and multiclassification involves more than two classes. Pre-processing methods will be used to significantly reduce the amount of data we need to feed into the model from the output of a camera.



A graph of blue bars

Description automatically generatedFigure 1 The MNIST dataset for character image recognition.

The image dataset I will being using to evaluate my models come from the MNIST dataset and consists of hundreds of images for each character. For the simplicity of the study, I will only be training/classifying the numeric characters (0-9). In any image recognition task, the accuracy will always be proportional to the amount of training data given.

Figure 2 The number of images for each numeric character shows its quite balanced.

## Evolutionary Algorithms Background

A diagram of a process flow

Description automatically generated

Evolutionary algorithms allow us to find the individual most suited for a specific task – it spawns many singular unique variations of itself which are tested by the environment they are in. (Nicholson, 2023). They are inspired by the process of natural selection – where the fittest individuals with better traits are more likely to survive and reproduce.

They maintain a population of candidate solutions and change them from generation to generation through genetic operations such as cross over, mutation and selection. They are well favoured for optimization tasks where there is a large search space to cover.

Figure 3 Flowchart I made to show the process of my genetic algorithm.

## Motivation for Using Evolutionary CNN

I chose to use a CNN for the learning method (inside the Genetic Algorithm) as they are good for image classification tasks because “the concept of dimensionality reduction suits the huge number of parameters in an image” whilst producing highly accurate predictions. Other basic learning models such as FCNs (Fully connected neural networks) and RNNs (Recurrent neural networks) can’t outperform a CNN due to their disregard for spatial relationships / 2-dimensional image data.

The main challenge with utilising a CNN is that the model design is very influential in how well It works. There are loads of different layers and techniques used to create the optimal model and it can take a lot of background knowledge and time to find it. By using a evolutionary GA we solve the problem of hyper parameter tuning, model architecture complexity (Fergal Stapleton, 2020), avoiding local optima and good scalability. Additionally, this can be done for any CNN learning task - all that must change are the input and output layers, and the GA will do the rest.

## Objectives of the Study

The goal of the study is to explore and prove how evolutionary GAs can be used to optimize the architecture of Convolution Neural Networks, aiming to improve image classification accuracy while reducing the computation costs and training times.

It’s difficult to compare this solution to just making a simple model using task background information as models can be infinitely complex and each model is made uniquely to the task its built for. The simulation I plan to design, and implement could be used for any learning task simply by changing the training data and letting the genetic algorithm find which model architecture is the most effective,

# Hypothesis and Methods

In this section I will address my choice of environment, agent, and adaption mechanisms. I will also explore the main controlling factors/parameters and structure of the genetic algorithm as well as the pre-processing method used for the images.

## Choice of Environment

I will be using Google Collab as my environment choice due to its available option of using a hardware accelerator. Using a GPU makes training time **significantly** faster and for the task of neuron-evolution, where each individual is a complex CNN that needs to be fit with lots of training data that consists of several thousand pixels of data.

A screenshot of a phone

Description automatically generated

Figure 4 Google Colabs hardware accelerator GPU option.

Using a GPU is faster because they have parallel and batch processing capabilities that use thousands of cores simultaneously. They also have a high throughput and memory bandwidth which lets them perform a vast number of calculations concurrently which is ideal for large scale CNNs. Finally, deep learning framework libraries are highly optimized for GPU usage.

A graph of blue bars

Description automatically generated with medium confidence

Figure 5 Efficiency of GPUs Compared to CPU when fitting/training a CNN

## Image Pre-processing

The pre-processing steps allow me “enhance some features” from real-world, high-resolution images so they more closely resemble the MINST dataset. This will increase the accuracy and robustness of the algorithm. (Kumar, 2021)

A collage of images of a number one

Description automatically generated

Figure 6 How Pre-processing in my simulation works.

This pre-processing method takes a real world or computer image and converts it to NumPy array so it can be transformed. These are the key steps in the function:

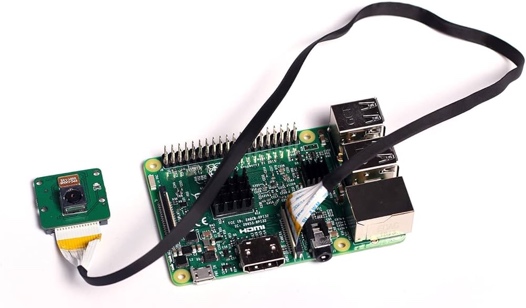
1. Scale the image down so the height is of 64 pixels (to match train data) but maintain resolution to not stretch the image.
2. Crop the image, so the width is also 64 pixels but ensure its cropped from the middle, so it captures the drawn character properly.
3. Grey scale the image so we can figure out which values lean to black, and which lean to white.
4. Get the flipped colour binary image (if pixel is blacker, make it white and vice versa).

## Agent Anatomy

The agent anatomy refers to the components and structure which allow my simulation to interact/respond with the environment it’s in. In this case, it involves getting raw image data to a predicted numeric character label.

### Sensors/Actions

The sensors for the robot which consider the physical environment will be simple standard cameras which pick up RGB images real world used for the image classification program. On a physical robot, the main action could be displaying the output classification on a small screen all controlled by a raspberry Pi - small cameras and displays could be attached to gain this functionality in a real robot other than just a simulation.



For the sake of simplicity and because I have no real experience building robots or with electronics, I will simulate the process using my phone/webcam camera to collect image data from the real-world environment.

### Brain Anatomy

CNNs is a deep learning model which acts as the “brain” of our system. It has the inputs consisting of the raw pixel data from an image, as well as the output actions which is the classification of said image. They are constructed from several perceptron’s that interact with each other by changing their weights based on the training information we feed them.

CNNs must have an input and output layer, the first layer must be a convolution layer which takes the 2d data, it will then have a flatten layer which forms it into a dense layer which then gives it to the output layer.

A green rectangular object with black lines

Description automatically generated

Figure 7 Best model architecture.

Between the input and output, a CNN’s anatomy consists of several interacting layers which all have different purposes, parameters, and conditions. The order normally goes Conv -> Pool -> Flatten -> Dense. The convolutional layer is the core building block which carries most of the load, it works using the dot product between two matrices – one being a kernel of the receptive field (Mishra, 2020). The pooling layer is designed to reduce spatial size by delivering a summary statistic. The dense layer, also known as the fully connected layer, works as all the neurons are connected to each other and can use matrix multiplication. (Mishra, 2020)

## Adaptation Mechanisms

Adaption Mechanisms refer to the several methods and process a genetic algorithm will use to evolve a population of candidate solutions towards a better model over iterative generations.

### Genetic Algorithm / CNN Parameters

The parameters for the GA are responsible for how individuals are made, selected, evaluated and changed during the entire process.

|  |  |  |
| --- | --- | --- |
| **GA Parameter** | **What it controls** | **Effects** |
| Number of Generations | How many cycles of the GA are run before outputting the best model | Increasing it will take more time but increase model accuracy |
| Number of train images for each model | How many images are fed to the models to test accuracy during the population evaluation step | More images will take more time but will get a more accurate accuracy for each model |
| Number of Individuals | How many individuals are in each generation cycle. | More individuals will mean a wider spread and increase exploration. |
| Number of Parents | How many parents are selected from the individuals | Will increase the genetic diversity and change convergence rate |
| Number of Offspring / Number of Random Ratio | Ratio of how many new individuals is randomly made to how many are created from the parents | Changes the exploration and convergence rate. With more random individuals there will be more exploration. |
| Cap accuracy (Optional) | If on, it will exit the GA algorithm if a model reaches the set cap accuracy | Having this could save time on useless generations if a model is “good enough” preventing overfitting |
| Mutation Rate | This changes how often a genome is selected to be modified | With a higher mutation rate, more CNN layers will be modified and will increase diversity/exploration |
| Mutation Strength | This controls how much the selected genome is changed | Will impact how much a induvial will change when they are mutated |

The parameters for the CNN Control how each individual CNN model are initialised at generation 0 (and in general practise) – which will indirectly affect the final output model.

|  |  |  |
| --- | --- | --- |
| **CNN Parameter** | **What it controls** | **Effects** |
| Filter/Unit/Node size | The number of nodes, kernel shape e.g. (3, 5). General numeric layer parameters. | Larger filters capture more complex patterns and increase capacity, can increase overfitting. |
| Max number of Conv2D layers | Maximum number of convolutional layers | Increasing this can allow model to learn more complex features, but can increase overfitting |
| Max number of Dense Layers | Maximum number of full connected layers | More dense layers can help model learn intricate patterns in data, but lead to overfitting |
| Max number of Pool Layers | Maximum number of pooling layers | Pooling layers help reduce spatial dimensions and extract dominant features – too many can lose detail |
| Layer Activation Type | Activation function applied to each layer | Different activations like ReLU, sigmoid, tanh, etc affect how the network learns and converges |
| Regularisation Strength | Strength of the regularisation applied (can be l1 or l2 – I used l2) | Stronger regularisation penalizes complex models, reducing overfitting but potentially underfitting. |

### Fitness Measure

An individual’s fitness is an accurate metric used to evaluate how well an algorithm performs against its predictive goal (Nicholson, 2023); it will tell us how effective the CNN is based on its genotypes (layers). This measure is used to select the parents which are allowed to reproduce for the next generation.

I considered using computation time as a partial contribution in the fitness score and weighting it along with the accuracy but decided that the significant measure is the model’s accuracy as processing times are subjective. To keep it simple and not add unwanted complexity, the fitness measure is simply the accuracy \* 100 (so 0.9 accuracy would give a fitness score of 90) similarly to how (Nivedh, 2020) did.

### Selection, Cross-Over, Mutation

Through the **selection** phase, the most fit parents are selected from the population to pass on to next generations (Yang, 2014). In this example, I have set the number of parents to 2. These parents are then used in the cross over function. I decided to use this rank selection process rather than tournament or roulette wheel selection as there are not many individuals in each generation (I talk about why later)

A graph of blue rectangular objects

Description automatically generated with medium confidence

Figure 8 Selection selects the fittest parents to be used for selective reproduction.

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generatedDuring the **Cross Over** stage, the parents are merged by swapping parts of another in chromosomes (Yang, 2014). I do this in my algorithm by swapping the dense and convolution layers. This would help to avoid local optima by generating diverse offspring which can introduce and explore new solutions.

Figure 9 Two best parents from the selection process. Figure 10 The resulting offspring.

A graph with red and blue lines

Description automatically generatedIn the **Mutation** stage, individuals in the offspring undergo a random mutation that’s controlled by 2 parameters: mutation rate and mutation strength. The function works by going through each layer and checking if a random decimal (0 – 1) is greater than the mutation rate. If so, the layer is mutated according to the strength – it can change the kernel size, number of nodes, regularisation strength and pool size (depending on layer). Mutation provides a mechanism for escaping local optimum (Yang, 2014).

Figure 11 How mutation works to add genetic diversity into the GA.

## Research Topic

My hypothesis is that the application of a genetic algorithm to the optimisation of Convolutional Neural Networks can significantly improve the performance of CNN models for specific tasks by effectively exploring the search space of both hyper parameters and model architecture. This proposes an effective solution to finding an optimal well-suited model for any task whilst avoiding randomly (or taking lots of time) building and testing models.

Research questions I aim to answer from this study:

1) Do the CNN models from the GA work well to unseen data?

2) Does the GA explore the search space properly?

3) Is it better than traditional methods such as Grid search?

# Results and Testing

In this section I will present the results obtained from testing the performance of both the GA and the final model it created. I will reflect and evaluate using new unseen data and data from the real-world environment.

## Performance Evaluation

A graph with blue lines and dots

Description automatically generatedIn this evaluation I gave my genetic algorithm 10 generations of 5 individuals, with a 2 selected parents per cycle with offspring ratio of 0.8 and a mutation rate of 0.9 with mutation strength of 0.2.

You can see from the diagram that the model’s best fitness is generally increasing until it reaches around 90 (where the cap could be considered useful, so it doesn’t go back down whilst exploring) where it struggles to find a better model and the exploration aspect brings it back down whilst searching for a better model. With more generations and individuals, theoretically it would be a less sharp graph but, as I will mention in the scalability and efficiency section, it would take a significant amount of time/power.

Figure 12 How accuracy increases with iterative generations.

In this evaluation I gave my genetic algorithm 10 generations of 5 individuals, with a 2 selected parents per cycle with offspring ratio of 0.8 and a mutation rate of 0.9 with mutation strength of 0.2.

**A diagram of a number of people

Description automatically generated with medium confidence**

Figure 13 First generation of randomly initiated individuals.

The first generation shows that many of the models aren’t performing very well, except for one model which has 7x the fitness of all the other models. What makes it stand out is that it has the pooling layer between each of the two convolutional layers.

**A diagram of a diagram

Description automatically generated with medium confidence**

Figure 14 Fifth generation, with selected and developed individuals.

By the filth generation we can see that the average accuracy is significantly higher and that the middle pooling layer from the first-generation model stuck around suggesting that it may be an evolutionary trend in models with high finesses. All individuals’ models have taken a similar architecture.

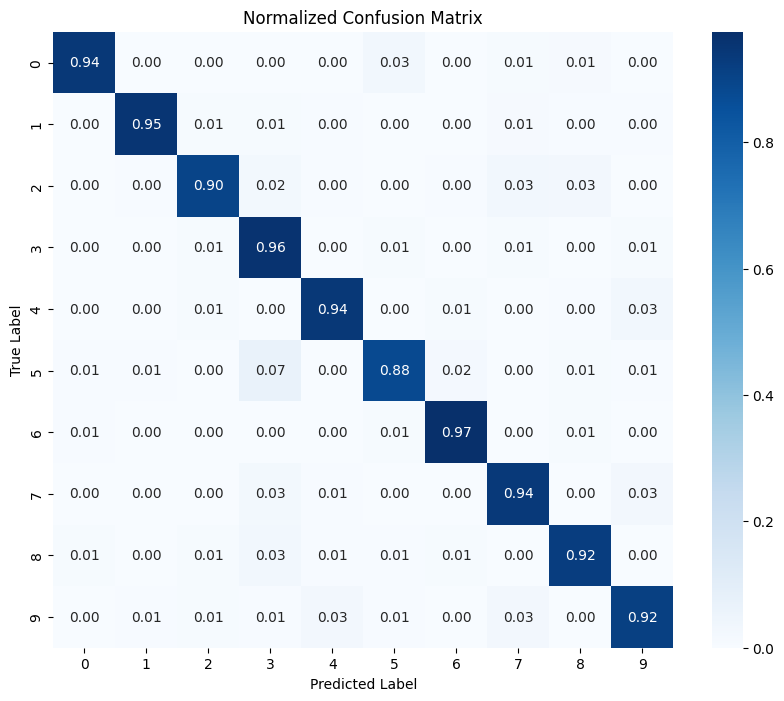


Figure 15 Confusion matrix for model on validation data.

This confusion matrix shows us the models’ instances of correct predictions and false predictions. From this we can see that the model mistakes 5s for 3s as the most common mistake (0.07). This is most likely because they share a similar shape/pattern.

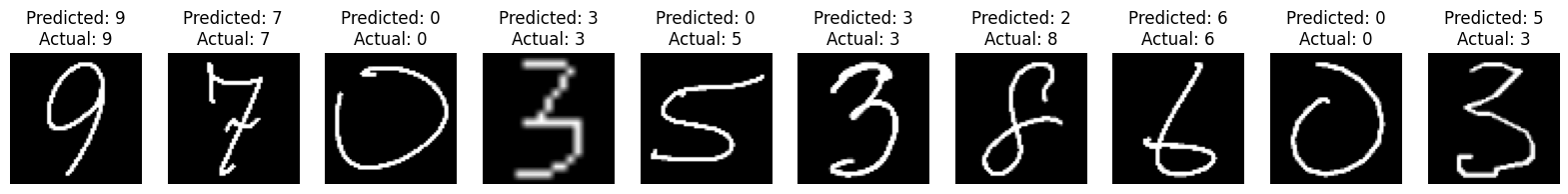


Figure 16 Some random test images with their actual label and the models predicted label.

Here are some example test images from the held-out validation set, above is the models predicted label and below is the correct label. You can vaguely notice that in the instances where an image is classified incorrectly, the pattern/shape of the character is like what its wrongly predicted as e.g. for the last character (3) you notice, due to the flat top and curly bottom that it could resemble a crooked 5. This is supported by the statement in Figure 14.

**A black and white image of a number

Description automatically generated**

Figure 17 Models prediction of a photo image from camera sensor.

Here I have drawn a character using a black pen on some white paper to test the model with real environment image data. You can see the pre-processing method works very well to abstract the image (without capturing the lined paper) and the model successfully predicts the correct class!

## Robustness and Generalisation

To test my algorithms robustness and generalisation, I used the best model to predict separate image test data (not from the MNST train set). This character set includes lots of different handwriting styles and will test the true capabilities of the model in real life.

A black squares with white numbers

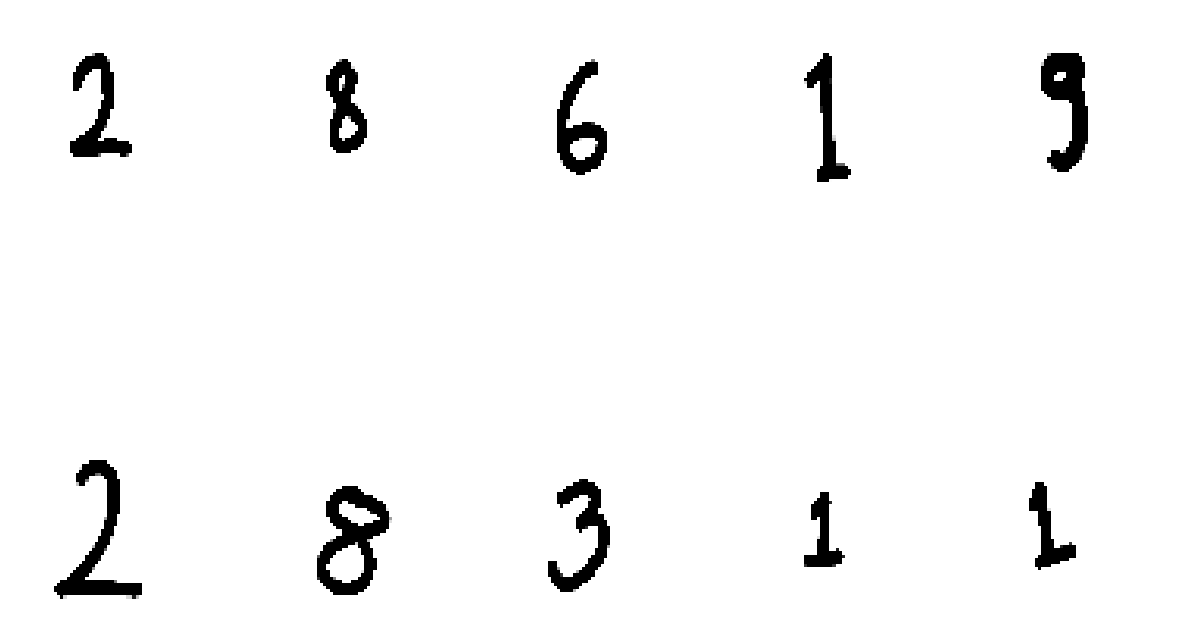
Description automatically generated

Figure 18 Same test data but pre-processed for the model.

Figure 19 The test data images (see references)

Unfortunately, the model didn’t seem to perform as well on the new data with an accuracy of only 0.2, which could suggest the algorithm suffers from overfitting despite the regularisation added. I attempted to increase regularisation which helped a little, but I believe the issues lies with my limited model complexity and training sizes due to heavy processing times. (Assume 0.1 as worst accuracy possible as 10 labels and randomly guessing would statistically get you 0.1). This issue may also lie in the training samples themselves, the MNIST characters seem to be similar sized whereas the testing samples seem to have a more chaotic style.

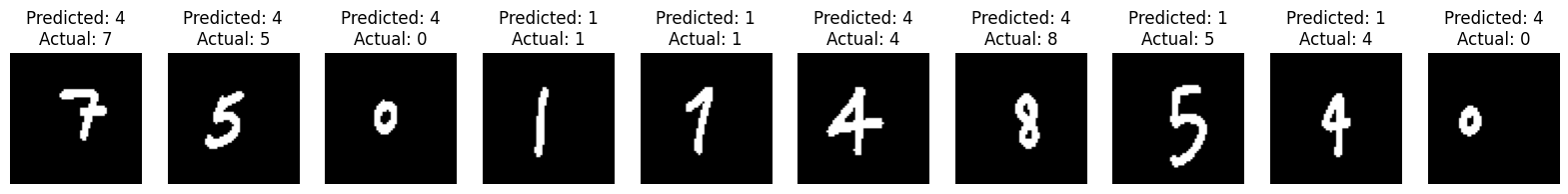


Figure 20 Poor performance on unseen more complex handwriting style character data set.

# Discussion

In this section I will discuss and reflect upon the results found and analyse what was done well and what I would add in the future. I will also dive into what makes the final model well suited for the task which is why it resulted in it being the fittest individual.

## Interpretation of Results

A diagram of a computer

Description automatically generated

This is the final model from the generation in Figure 13. You will notice that the accuracy is slightly higher than the one in the generation photo (fitness = accuracy \* 100) – this is because, to save processing time, after the best model is found it is then refitted/trained with **all** the data available, meaning it will gain a higher accuracy.

The final model has an accuracy high enough to be considered effective, though it is hard to benchmark if the process of evolution would have been easier than simply thinking about the task and trying a few different models. What makes this model so efficient is the 2 Dense layers which allow it to capture complex decision boundaries which occur in image data as well as the pooling layer between the Convolutional layers which lets the network progressively reduce the spatial dimensions of the feature maps whilst learning hierarchal representations of the input data (reducing overfitting).

A graph of loss and loss of a curve

Description automatically generated with medium confidence

Figure 21 Effect of learning on Loss and Accuracy (overfitting).

As you can see in Figure 20, the final model is prone to overfitting. During epoch 3 – 4, you can see that the validation accuracy plumets whilst the training accuracy remains stable – because of this we end up with a final accuracy lower than what had been previously obtained.

Overall, figure 11 and figure 14 prove that the algorithm works in the sense that the final model both accurately classifies test data and that it was created from the GA after a certain amount of searching was done.

## Implications of Neuroevolutionary Algorithms

Neuroevolutionary algorithms are contributing to the advancement of AI by further automating am essential design process in the construction of complex models. Sometimes a search space is too large for either human or simple brute force approach to handle, GAs overcome this with their adaptability and evolutionary approach. Further, they aren’t task dependant, meaning it can be deployed on any learning task using a complex deep learning model.

They are powerful in domains of reinforcement learning and have many real-world applications such as the adaptive, non-linear control of physical devices – these algorithms have been used to create and evolve models to “drive robots, automobile’s and even rockets”. (Prof. Risto Miikkulainen, 2013)

There are ethical and legal issues surrounding AI with algorithmic bias, transparency, accountability, and potential social impacts. Further, evolutionary GAs are still very computation expensive, requiring a strong GPU (Figure 4) and a lot of time.

## Fail cases and Future Directions.

A number written in white on a black background

Description automatically generated

Figure 22 some fail cases of the model wrongly predicted some of the MNIST validation data.

In the future, some ideas to expand and improve upon what I have tested and developed are:

* Add **normalisation** layers and **drop out** layers into the genotypes of individuals to allow a greater exploration of the search space potentially increasing accuracy of final model.
* Make it more robust to more styles of handwriting. I could do this by feeding it more diverse training data and maybe augment new data by transforming current train data to make models more robust to strange angles and perspectives that you will find in the real world.
* Attempt to tune the Genetic Algorithm parameters, such as number of individuals, to find what is most effective. I didn’t attempt this as its task dependant and would have taken a VERY long time to properly test and produce accurate/reliable results.
* Test the GA with more classification or learning tasks to see that it can adapt and create effective models for any situation. Some other ideas I could use it for are Natural Language Processing tasks like sentiment analysis or facial alignment or face recognition.

## Conclusion and Summary

Here I will answer the questions that I previously proposed in the Research topic section.

Do the CNN models from the GA work well to unseen data?

* The CNN model produced works very well on the held-out validation testing data and are accurate with hand drawn images, but the model fails to work on a smaller handwritten dataset due to it being overfitted. With more time I could perform tests on increasing the regularisation genotype to allow for more regularisation to test if it improves the model’s robustness.

Does the GA explore the search space properly?

* The GA explores the search space efficiently, allowing for a large and diverse range of models. Additionally, it utilises mutation (Figure 10) and adding random new models to make sure the GA avoids local optima. If you observe Figure 12 you will notice all the models have unique attributes.

Is it better than traditional methods such as Grid search?

* Grid search can be useful for simple and small-scale optimization problems, neuro evolution GAs are better for searching high dimensional search spaces and offer a more intelligent and adaptive approach to find optimal solutions compared to a grid search which simply goes through each hyper parameter.

In conclusion, applying genetic algorithms to optimise convolution neural networks significantly improves model performance for specific tasks. This method efficiently explores a large search space of hyper parameters and model architectures, outperforming exhaustive and random searches. Consequently, it reduces the time and computational effort required to find optimal models, demonstrating that GAs is a highly effective tool for tailoring CNN models for specialised tasks.

# References

de Campos, T. a. B. B. a. V. M., 2009. Character recognition in natural images. [Online]   
Available at: https://www.kaggle.com/datasets/dhruvildave/english-handwritten-characters-dataset?resource=download&select=Img  
[Accessed 14 5 2024].

Kumar, B., 2021. Image Preprocessing — Why is it Necessary?. [Online]   
Available at: https://medium.com/spidernitt/image-preprocessing-why-is-it-necessary-8895b8b08c1d  
[Accessed 14 21 2024].

Mishra, M., 2020. Convolutional Neural Networks, Explained. [Online]   
Available at: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939  
[Accessed 14 5 2024].

Nivedh, S., 2020. Optimization of CNN Architecture using Genetic Algorithm for Image Classification. [Online]   
Available at: https://sainivedh.medium.com/optimization-of-cnn-architecture-using-genetic-algorithm-for-image-classification-5c48f25dac9c  
[Accessed 14 5 2024].

Prof. Risto Miikkulainen, D. J. L., 2013. Neuroevolution. [Online]   
Available at: http://www.scholarpedia.org/article/Neuroevolution#Applications  
[Accessed 14 5 2024].

Yang, X.-S., 2014. Cross-over Mutation. [Online]   
Available at: https://www.sciencedirect.com/topics/computer-science/cross-over-mutation  
[Accessed 14 5 2024].

Yann LeCun, C. C. C. J. B., 1994. THE MNIST DATABASE of handwritten digits. [Online]   
Available at: http://yann.lecun.com/exdb/mnist/  
[Accessed 12 5 2024].

# Appendix

Please see **AIABNueroEvolution.ipynb** google colab notebook for full code.