Image Recreation Using Evolutionary Algorithms

Chris Schilling

1.0 Introduction

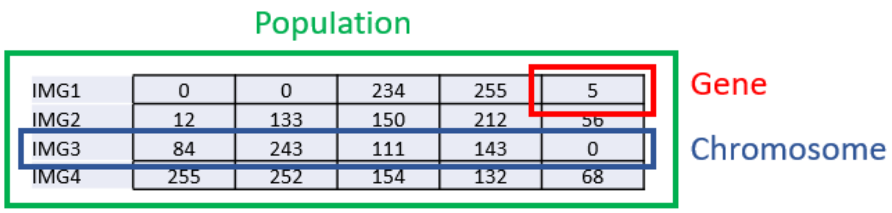
Evolutionary Algorithms are just as they sound. They generationally evolve with the goal of achieving a specific target. With this in mind, the goal of this project is to use a Genetic Algorithm, which is a type of Evolutionary Algorithm, to achieve the goal of recreating pictures in the most efficient way possible. These algorithms are interesting as they are fundamentally simple, in that they mimic the evolutionary experience we see in nature but can very quickly become cumbersome and slow. However, due to the compounding effect of generational evolution, they can be excellent tools to solve problems.

2.0 Project

Originally, this project began with the aspiration to determine the limitations of Evolutionary Algorithms with a program to solve grey scale images, but the initial testing demonstrated that this task was easily accomplished. As a results, the complexity of this project was increased by changing the task to solve for a full color image. This change to color images increased the overall complexity as color images have three pixel values per pixel for the RGB color scale necessary to create color images instead of one color scale per pixel for grey scale images. These pixel values range from 0 to 255 to numerically represent the variation in color tone that are present in each Gene in a Chromosome across the entire Population.

2.1 Genetic Algorithm Population

The purpose of a Genetic Algorithm is the generational evolution of the Population. The Population contains Chromosomes that represent one specific child image from a generation. A Chromosome is one individual image made up of Genes. Genes are the individual pixel values in each child image with values ranging from 0 to 255 (one set of values per pixel if gray scale and three sets of values per pixel if color). A visual of this can be seen in Figure 2.1.1. For this project with four different values were used for the Population: 50, 200, 500, and 2,000.



IMG1	0	0	234	255	5
IMG2	12	133	150	212	56
IMG3	84	243	111	143	0
IMG4	255	252	154	132	68

Figure 2.1.1 – Example of Population containing Chromosomes with Pixel Values for the Gene of each Chromosome

2.2 Genetic Algorithm Components

A Genetic Algorithm runs off the notion of natural selection which takes the fittest individuals from the population to produce offspring which inherit a combination of genes from the parents for the next generation. With this in mind, if the parents have a good fitness score, then a portion of their offspring should have an even better fitness score than their parents by inheriting those Genes that drive the improvement in the fitness score. This occurs over five stages which are: Initial Population, Fitness Function, Selection, Crossover and Mutation.

Starting off, the Initial Population is created based upon a specific Population value of Chromosomes (For example: 50, 200, 500 or 2,000). The Chromosome size is based on the number of pixel values within the Target Image. The algorithm then creates random values for the Genes in this Initial Population to produce an Image similar to the example shown in Figure 2.2.1.

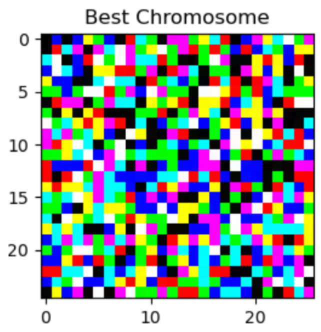


Figure 2.2.1 – Example Image of Initial Population

Then the program scores the Population created in a generation through the Fitness Function. The Fitness Function computes a fitness score for each Chromosome on how closely it matches the Target Image. In this program, we will be using three Target Images which can be seen in Figures 2.2.2 – 2.2.4.

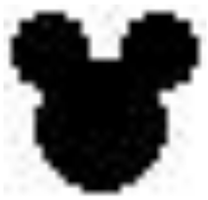


Figure 2.2.2 – Mouse Image, 26x25, 1,950 Pixel Values

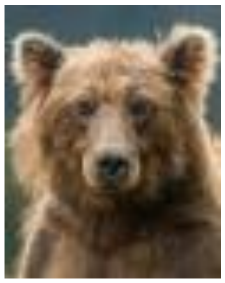


Figure 2.2.3 – Bear Image, 61x71, 14,091 Pixel Values

Figure 2.2.4 – Hillside Image, 60x40, 7,200 Pixel Values

To measure generational progress toward reaching the goal of duplicating the Target Image, this project will run three Fitness Functions with all three having the goal of reaching the target value of zero. A target value of zero means the Chromosome has no difference when compared to the pixel values of the Target Image. Fitness Model 1 computes the mean of the absolute pixel differences as its scoring method and is shown below in Formula 2.2.1. The second fitness function is very similar to the first but instead it utilizes the sum of the absolute difference between the Target Image and Chromosome and is shown below in Formula 2.2.2. Finally, the third fitness function fitness score is based on how many pixels are incorrect and is shown below in Formula 2.2.3.

$$Fitness\ Score = \bar{x}(|TargetImage - Chromosome|)$$

Formula 2.2.1 – Mean of the Absolute Difference Between Target and Chromosome

$$Fitness\ Score = \sum |TargetImage - Chromosome|$$

Formula 2.2.2 – Sum of the Absolute Difference Between Target and Chromosome

$$Fitness\ Score = \sum_{x \in A} (1 - \delta_{x,0})$$

Formula 2.2.3 – Count Nonzero from Absolute Difference Between Target and Chromosome using Kronecker Delta

Once the project has computed a fitness score across the population, it then performs a Selection process in which the program will take the top 50% of the fittest individuals (Chromosomes) who are closest to the goal of a fitness score of zero. This pool of the top 50% is then used for the creation of 90% of the next generation. The remaining 10% of the next generation is attributed to the Elitism aspect of the program where the top 10% of the fittest individuals from the current generation move directly into the next generation. This Elitism is included to ensure that   the program will retain the prior generation's most fit individuals.

After selecting the top 50% of individuals (Chromosomes) to be the parents of the next generation, this selection is used to create the offspring of then next generation with a method called Crossover. For this project, Crossover is going to be an even 50/50 split of Gene selection from the two parents. This means that for each Gene in the Chromosome, each new Gene in the offspring has an even chance of obtaining a Gene from either parent. For example, in Figure 2.1.1, if IMG1 and IMG2 are selected as part of the top 50%, the offspring of these two images are created from a combination of these two Chromosomes. For the first Gene in the next generation's Chromosome say parent 2's, or IMG2's gene is selected with a pixel value of 12 and is passed to the offspring then the child would have its first gene and the 50/50 random split would continue until all of the pixels are populated.

Once the Crossover is completed for each Gene in the Chromosome, the project performs Mutation. Mutation is the random selection of a Gene in a Chromosome to a random value between 0 and 255 even if the parents did not have that value in either Gene for selection in Crossover. The mutation rate for these images is small at only 0.05% of the genes in the Chromosome, but this allows the project to find new solutions if it begins to stagnate. We then begin this process over again with returning to the Fitness Function process and repeat until the project reaches a fitness score of 0 or the maximum number of generations, which is set at 1,500 generations.

This process developed significantly longer run times when the color Target Image was introduced as the project had to process more values for each pixel while doing the comparison. Despite longer run times, the project still functioned effectively as an Evolutionary Algorithm~~.~~

3.0 Tests

For this project, an initial brief test of increasing population size on the Mouse Image in Figure 2.2.2 was performed. The purpose of this initial test was to determine if the increase in runtime needed to process a generation with a larger population was offset by the time saving in runtime from fewer generations being required to obtain an increasingly more accurate Population. These runs can be seen in Figure 3.0.1. From this figure, it can be seen that by increasing the population size, the project is able to get closer to the correct pixel count over fewer generations. But the cost of increasing population size is that this project has a time complexity of N*N meaning it is quadratic in nature since it has two loops which is proportional to the square of N. This means that a population of 250 takes 00:1:50 to run, a population of 500 takes 00:03:40, a population of 1,000 takes 00:06:20, and, finally, a population of 2,000 takes 00:12:30. To determine if this runtime cost could be moderated, this project also focused on running two version of this Evolutionary Algorithm: one single threaded and the other multi-threaded.

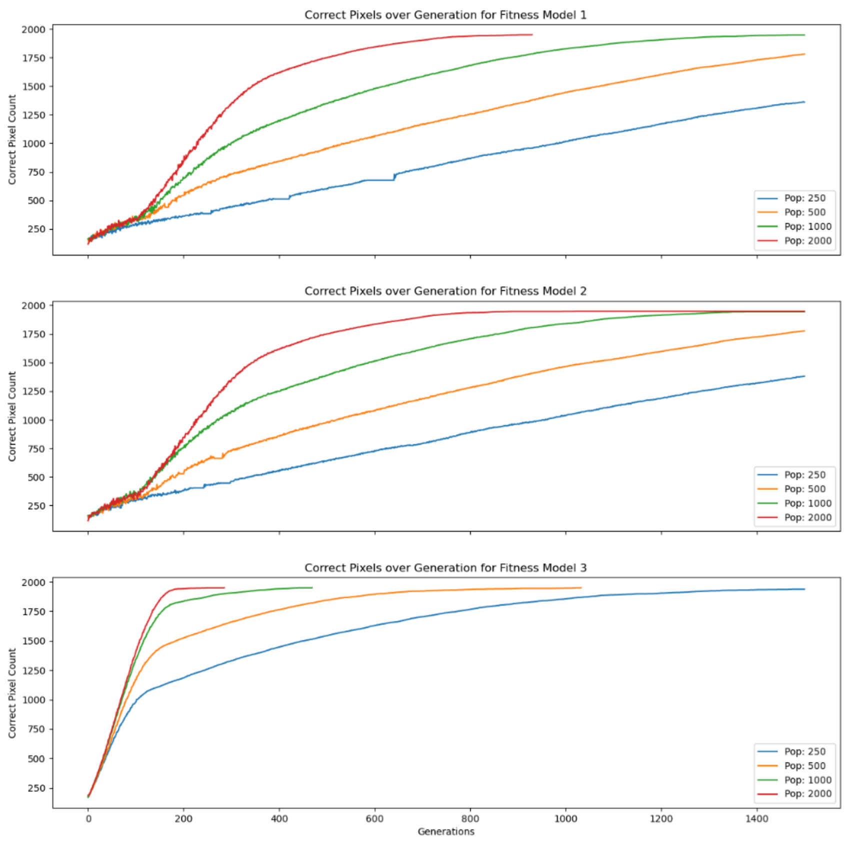


Figure 3.0.1 – Correct Pixel Count for Different Population Sizes

3.1 Multi-Threaded

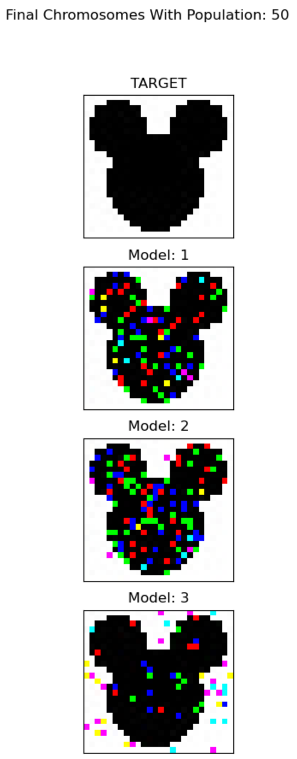
Multi-Threading will take our whole image and break it up into smaller chunks or groups for the project to individually work on in parallel. For this project, Multi-Threading has been incorporated by splitting up the program into 10 smaller groups of Genes for the program to concurrently process. For example, the Mouse Image in Figure 2.2.2 has 1,950-pixel values. With the single thread operation, the program is going to be working with a Population of size 200 with a Chromosome size of 1,950 Genes. With the Multi-Thread operation, the project splits the work into 10 smaller subroutines but with a population size still at 200 but with the image split up into 10 groups to be process concurrently with each group now having a size of only 195 Genes.

3.2 First Run

For the First Run, this project used the Mouse Target Image in Figure 2.2.2 with two different populations sizes of 50 and 200 and Threads of 1 (single threaded) and 10 (multi-threaded). The first Data Fields to be measured are Correct Pixel Percentage which looks at how many of the pixels (Genes) in the Chromosome perfectly match the Target Image Genes. The second Data Field is the Percent Similar to Target which looks at how close the Chromosome Gene is to the Target Gene. Finally, the third Data Field Time records the time each method took to run in Hours: Minutes: Seconds. This data can be seen in Figure 3.2.1 along with the Final Result Images seen in Figure 3.2.2 – 3.2.3.

Picture	Population	Threads	Model	Correct Pixel Count	Total Pixels	Correct Pixel Percentage	Percent Similar to Target	Time	Generation Finished
Mouse	50	1	1	604	1950	30.97%	98.39%	0:00:46	1500
Mouse	200	1	1	1262	1950	64.72%	99.60%	0:01:35	1500
Mouse	50	10	1	1697	1950	87.03%	99.83%	0:00:47	1500
Mouse	200	10	1	1945	1950	99.74%	99.98%	0:01:47	1232
Mouse	50	1	2	603	1950	30.92%	98.24%	0:00:46	1500
Mouse	200	1	2	1258	1950	64.51%	99.57%	0:01:34	1500
Mouse	50	10	2	1697	1950	87.03%	99.56%	0:00:47	1500
Mouse	200	10	2	1949	1950	99.95%	99.99%	0:01:32	1130
Mouse	50	1	3	1348	1950	69.13%	59.68%	0:00:47	1500
Mouse	200	1	3	1901	1950	97.49%	96.82%	0:01:34	1500
Mouse	50	10	3	1823	1950	93.49%	85.86%	0:00:47	1500
Mouse	200	10	3	1950	14091	13.84%	100%	0:01:03	957

Figure 3.2.1 – Mouse Image Data



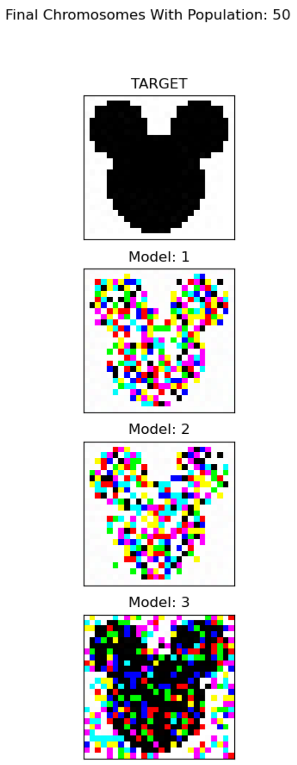
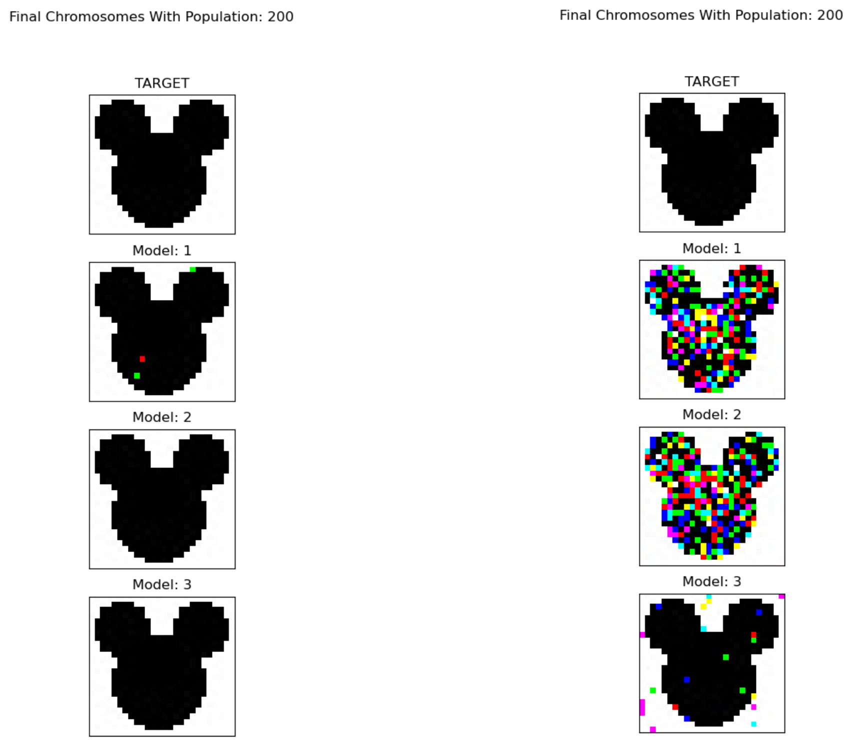


Figure 3.2.2 – Mouse Final Image for Population of 50: Multi-Threading on left and Single Thread on right


Final Chromosomes With Population: 200
TARGET
Model: 1
Model: 2
Model: 3
Final Chromosomes With Population: 200
TARGET
Model: 1
Model: 2
Model: 3
Figure 3.2.3 – Mouse Final Image for Population of 200: Multi-Threading on left and Single Thread on right

3.3 Second Run

For the Second Run, this project used the Hillside Target Image in Figure 2.2.4 with two different populations sizes of 500 and 2,000 and Threads of 1 (single threaded) and 10 (multi-threaded). The Data Fields are the same as the three Data Fields used in the First Run. First, the Correct Pixel Percentage which looks at how many of the pixels (Genes) in the Chromosome perfectly match the Target Image Genes. Second, the Percent Similar to Target which looks at how close the Chromosome Gene is to the Target Gene. Finally, Time which looks at the time each method took to run in Hours: Minutes: Seconds. This data is shown in Figure 3.3.1 along with the Final Result Images shown in Figure 3.3.2 – 3.3.3.

Picture	Population	Threads	Model	Correct Pixel Count	Total Pixels	Correct Pixel Percentage	Percent Similar to Target	Time	Generation Finished
Hillside	500	1	1	309	7200	4.29%	85.82%	0:05:19	1500
Hillside	2000	1	1	643	7200	8.93%	93.46%	0:18:47	1500
Hillside	500	10	1	2357	7200	32.74%	98.38%	0:06:30	1500
Hillside	2000	10	1	5916	7200	82.17%	99.81%	0:22:21	1500
Hillside	500	1	2	291	7200	4.04%	85.99%	0:04:54	1500
Hillside	2000	1	2	654	7200	9.08%	93.33%	0:17:27	1500
Hillside	500	10	2	2411	7200	33.49%	98.44%	0:05:38	1500
Hillside	2000	10	2	5923	7200	82.26%	99.82%	0:21:03	1500
Hillside	500	1	3	2408	7200	33.44%	36.74%	0:05:07	1500
Hillside	2000	1	3	4622	7200	64.19%	65.30%	0:18:30	1500
Hillside	500	10	3	5481	7200	76.13%	71.73%	0:05:14	1500
Hillside	2000	10	3	7158	7200	99.42%	99.22%	0:19:24	1500

Figure 3.3.1 – Hillside Image Data

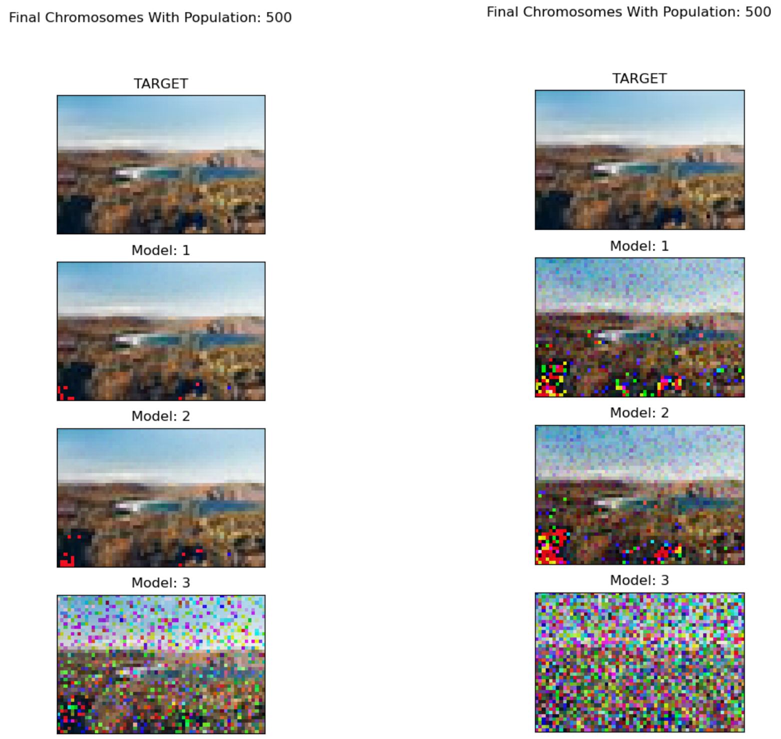

Final Chromosomes With Population: 500
TARGET
Model: 1
Model: 2
Model: 3
Final Chromosomes With Population: 500
TARGET
Model: 1
Model: 2
Model: 3

Figure 3.3.2 – Hillside Final Image for Population of 500: Multi-Threading on left and Single Thread on right

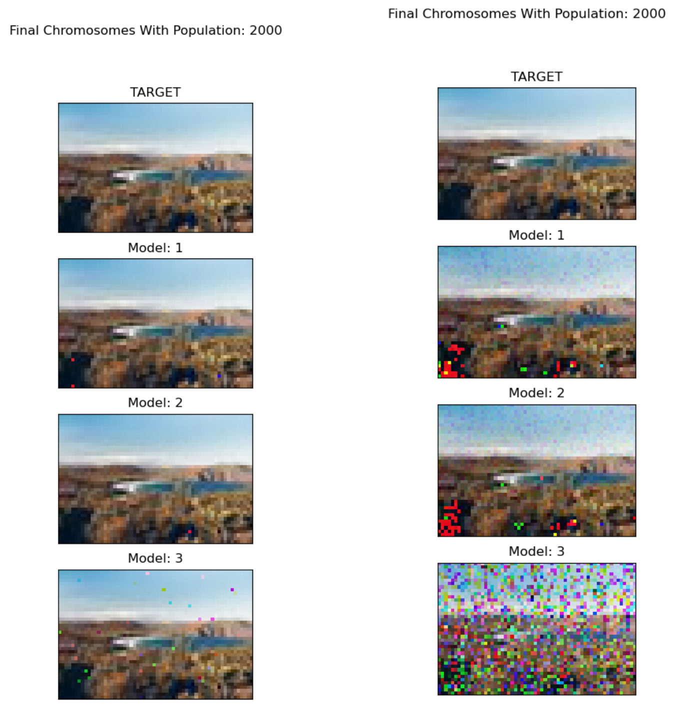

Final Chromosomes With Population: 2000
TARGET
Model: 1
Model: 2
Model: 3
Final Chromosomes With Population: 2000
TARGET
Model: 1
Model: 2
Model: 3

Figure 3.3.3 – Hillside Final Image for Population of 2000: Multi-Threading on left and Single Thread on right

3.4 Third Run

For the Third and final run, this project used the Bear Target Image in Figure 2.2.3 with two different populations sizes of 500 and 2,000 and Threads of 1 (single threaded) and 10 (multi-threaded). The Data fields are again the same as the previous runs in that the items to take note of are Correct Pixel Percentage, the Percent Similar to Target and Time each method took to run in Hours: Minutes: Seconds. This data can is shown in Figure 3.4.1 along with this are the plots for each data field for the 2,000 population run along with the Final Result Images for both the 500 population run and 2,000 population run which is shown in Figures 3.4.2 – 3.4.7.

Picture	Population	Threads	Model	Correct Pixel Count	Total Pixels	Correct Pixel Percentage	Percent Similar to Target	Time	Generation Finished
Bear	500	1	1	422	14091	2.99%	72.37%	0:07:03	1500
Bear	2000	1	1	816	14091	5.79%	86.73%	0:26:15	1500
Bear	500	10	1	2193	14091	15.56%	95.19%	0:06:09	1500
Bear	2000	10	1	5989	14091	42.50%	98.77%	0:23:37	1500
Bear	500	1	2	367	14091	2.60%	72.55%	0:06:45	1500
Bear	2000	1	2	873	14091	6.20%	86.81%	0:25:22	1500
Bear	500	10	2	2176	14091	15.44%	95.17%	0:06:11	1500
Bear	2000	10	2	5904	14091	41.90%	98.74%	0:22:55	1500
Bear	500	1	3	3498	14091	24.82%	0.00%	0:07:10	1500
Bear	2000	1	3	6929	14091	49.17%	31.99%	0:27:46	1500
Bear	500	10	3	9346	14091	66.33%	52.19%	0:05:44	1500
Bear	2000	10	3	13734	14091	97.47%	96.43%	0:21:31	1500

Figure 3.4.1 – Bear Image Data

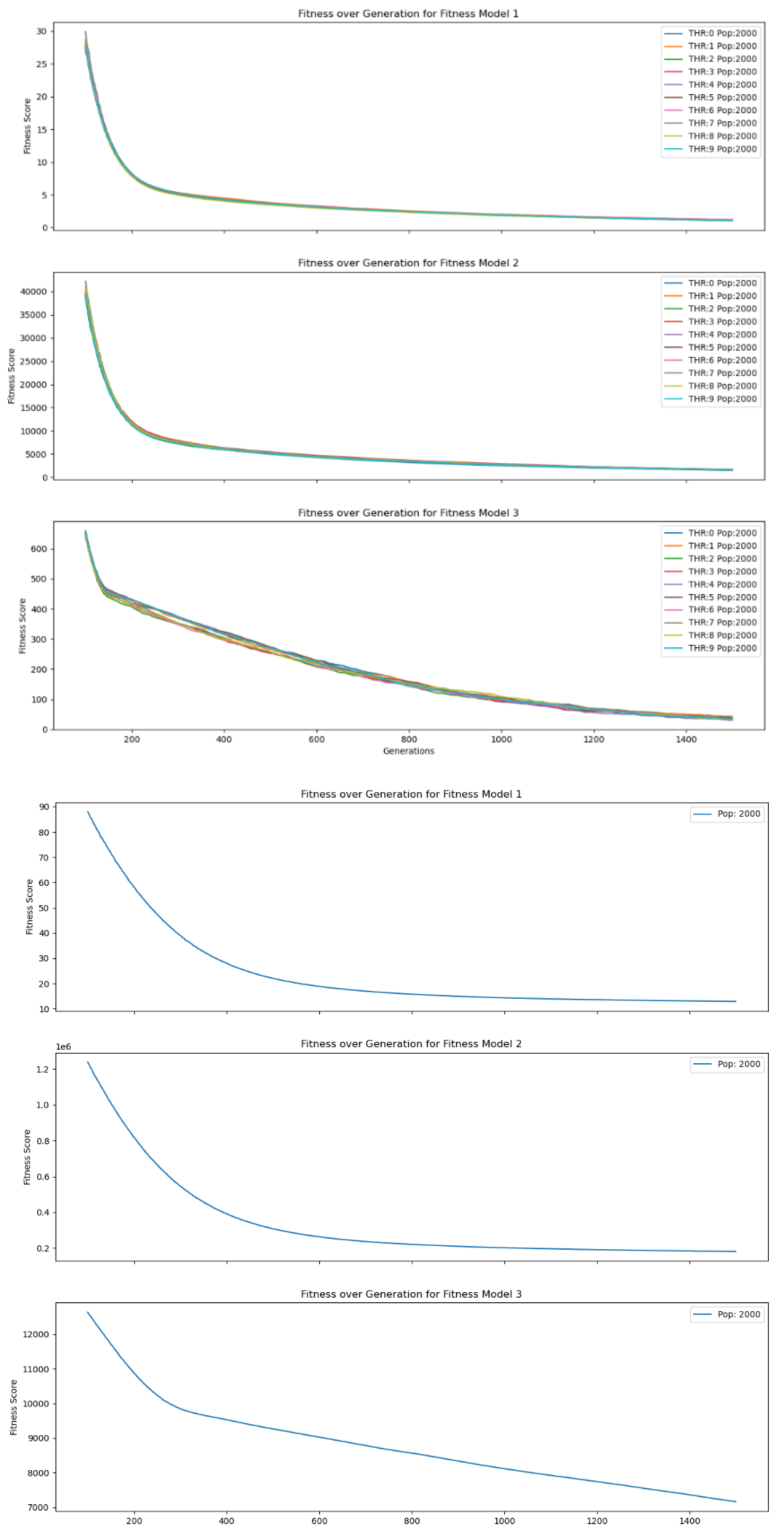


Figure 3.4.2 – Bear Fitness Value over Generation: Multi-Threading on Top and Single Thread on bottom

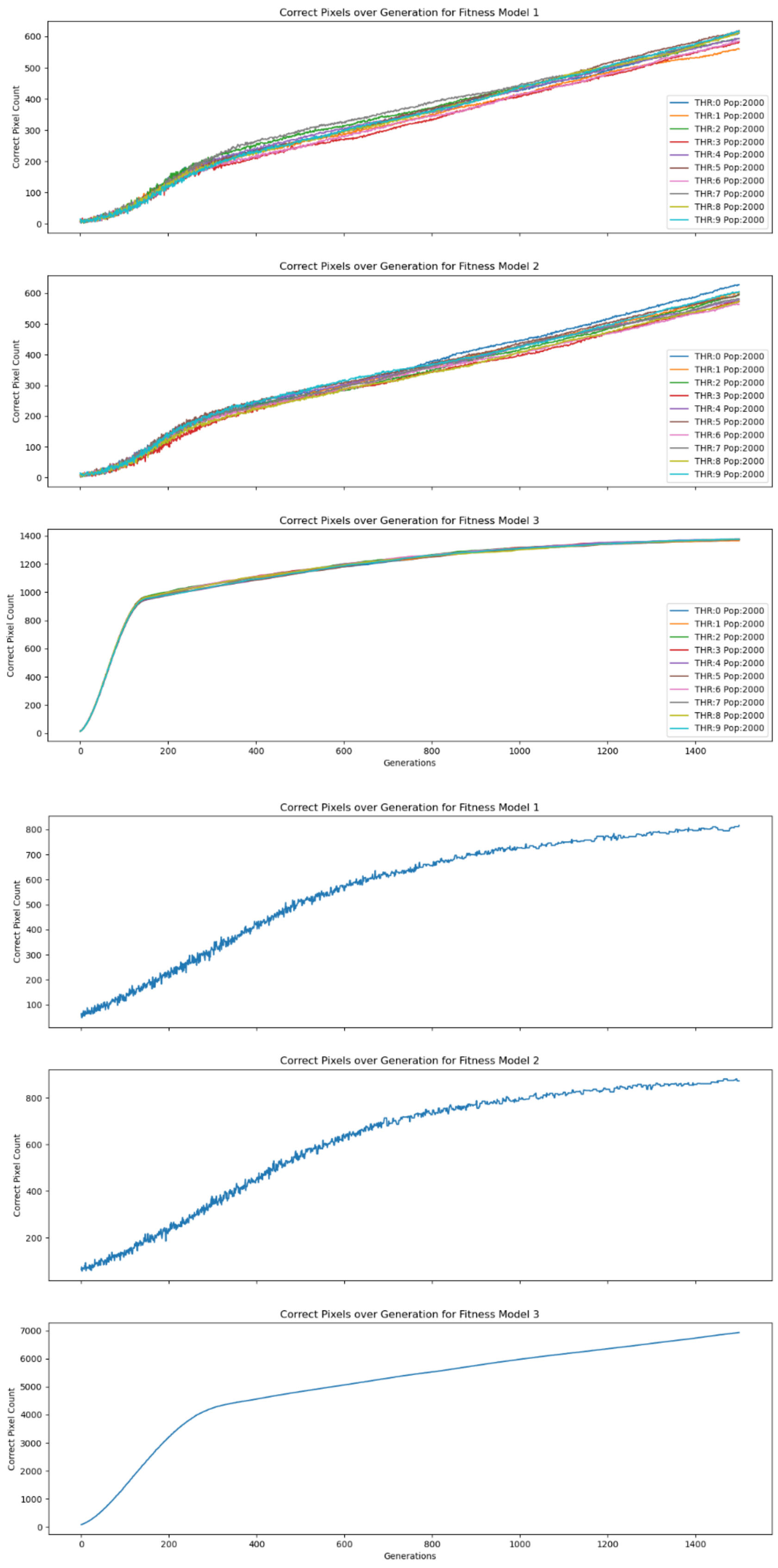


Figure 3.4.3 – Bear Correct Pixel Count over Generation: Multi-Threading on top and Single Thread on bottom

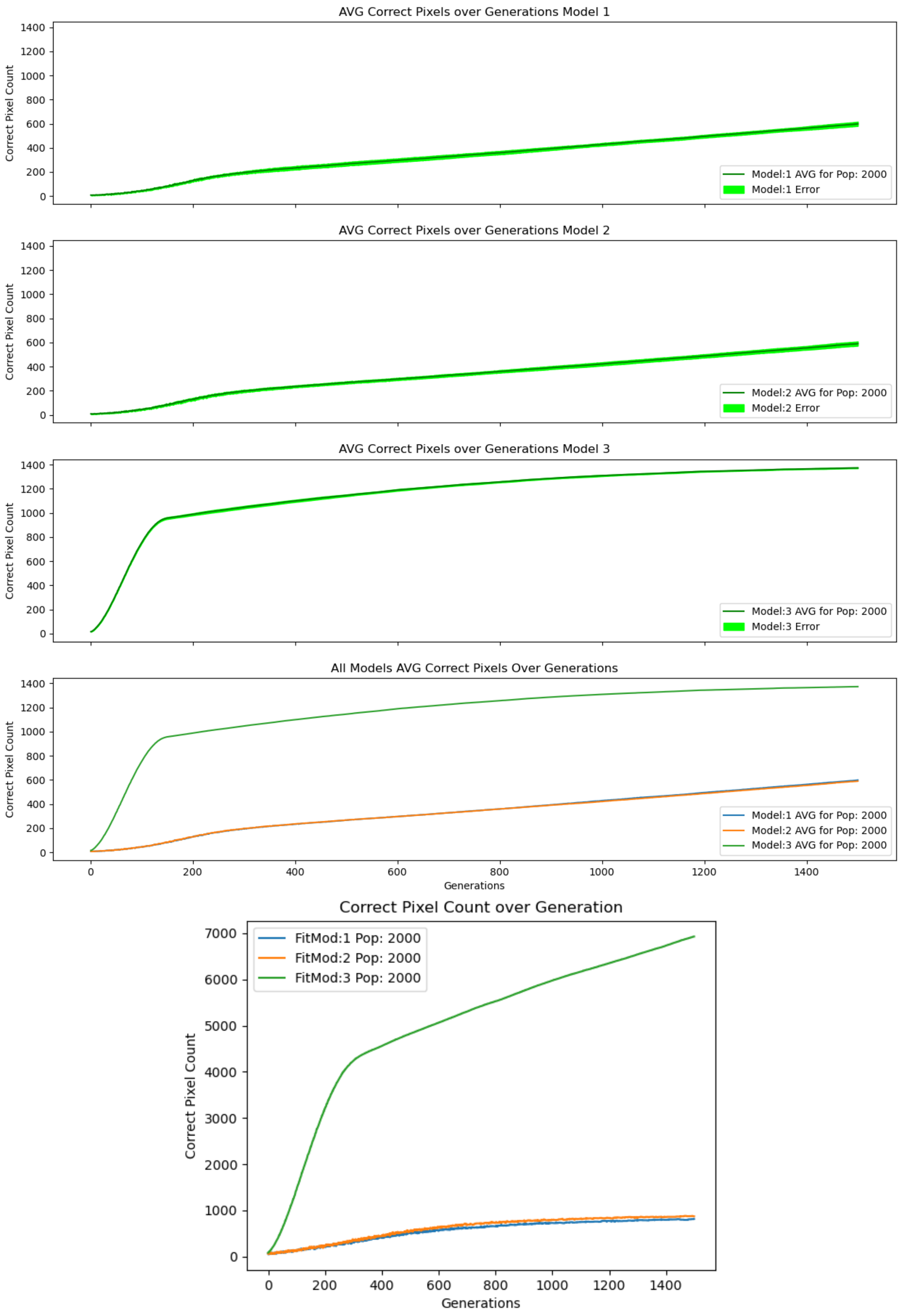
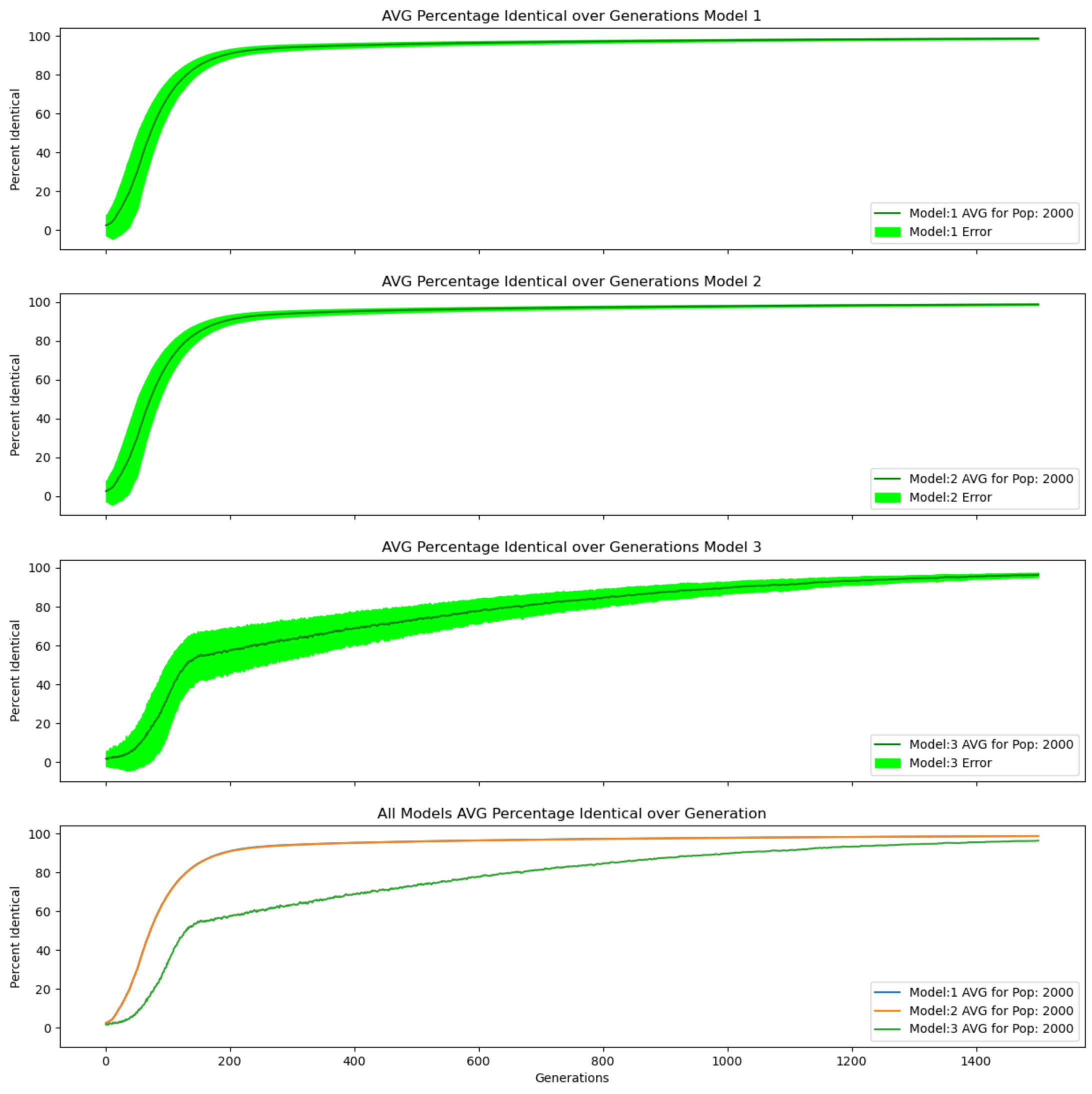


Figure 3.4.4 – Bear Correct Pixel Count over Generation Combined: Multi-Threading on top and Single Thread on bottom



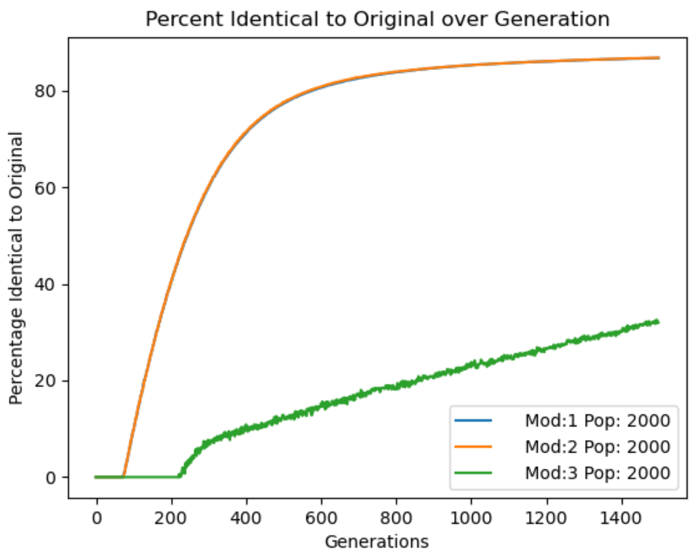


Figure 3.4.5 – Bear Percent Identical Pixel over Generation Combined: Multi-Threading on top and Single Thread on bottom

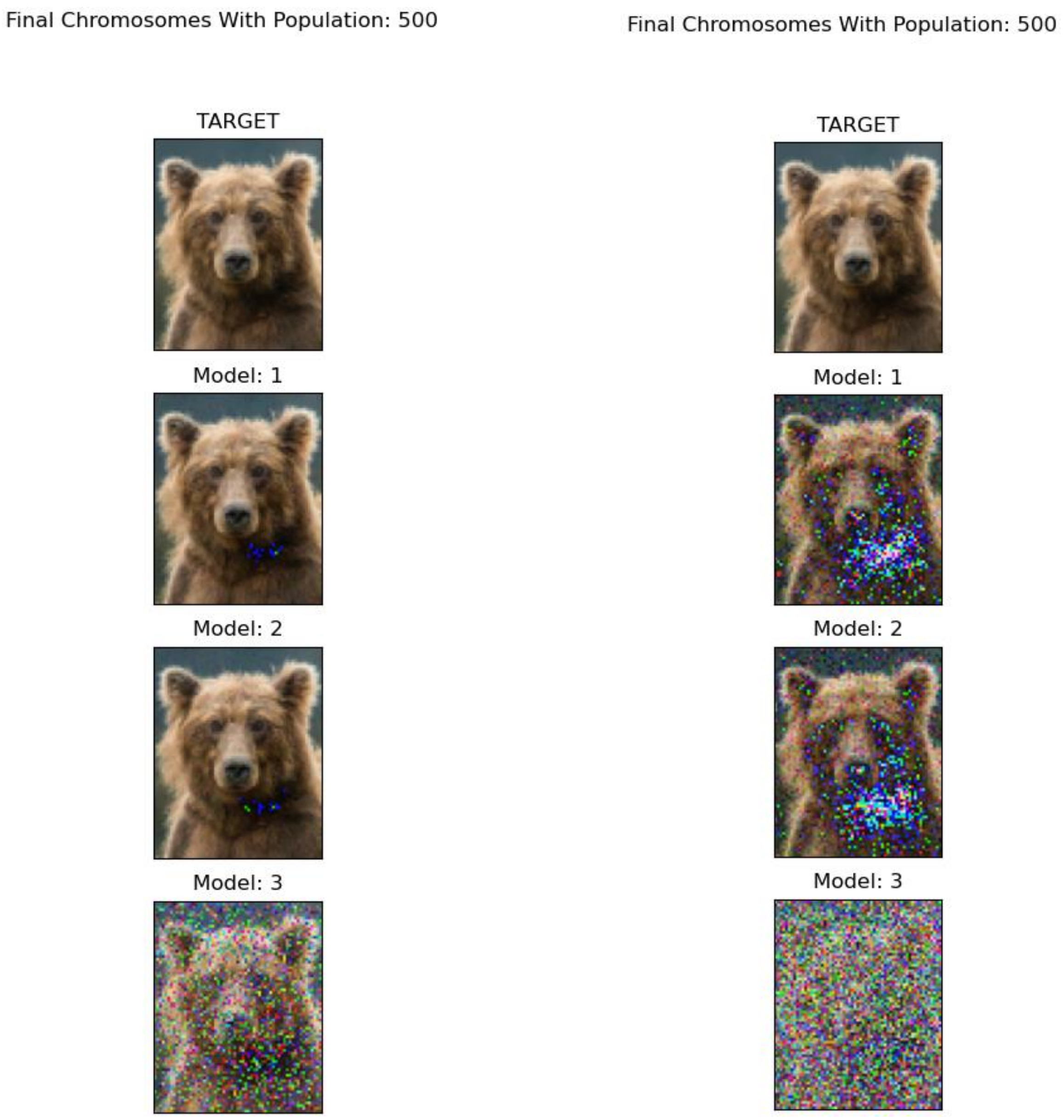
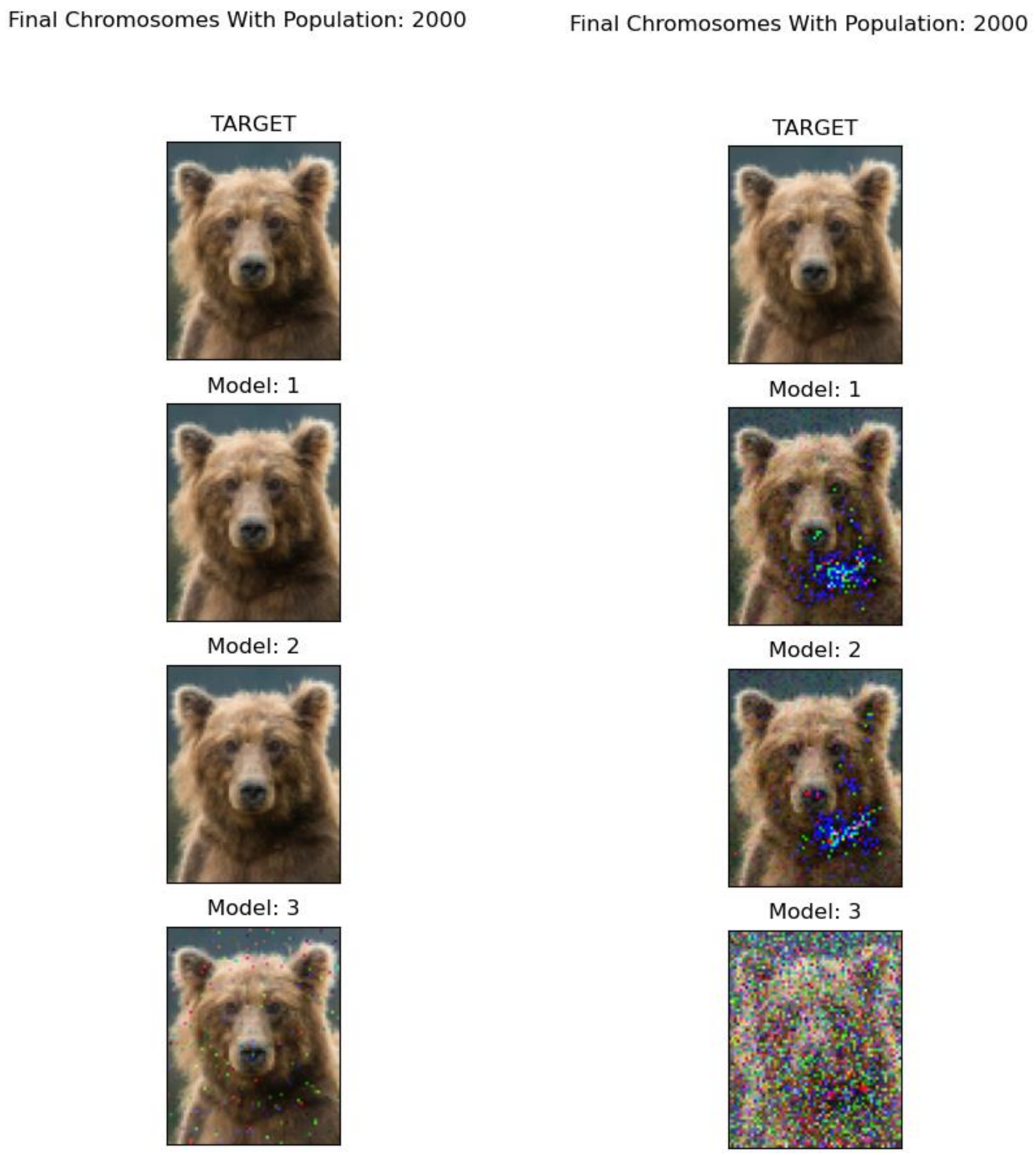

Final Chromosomes With Population: 500
TARGET
Model: 1
Model: 2
Model: 3
Final Chromosomes With Population: 500
TARGET
Model: 1
Model: 2
Model: 3

Figure 3.4.6 – Bear Final Image for Population of 500: Multi-Threading on left and Single Thread on right


Final Chromosomes With Population: 2000
TARGET
Model: 1
Model: 2
Model: 3
Final Chromosomes With Population: 2000
TARGET
Model: 1
Model: 2
Model: 3
Figure 3.4.7 – Bear Final Image for Population of 2000: Multi-Threading on left, Single Thread on right

4.0 Conclusion

The project's success in recreating an image is being measured by the progress of the three Data Fields. However, visual accuracy is best determined by the Percent Similar to Target metric and less so by the highest Correct Pixel Percentage. For all three runs, Models 1 and 2 performed the worst in obtaining a good score for Correct Pixel Percentage. In addition, Models 1 and 2, along with single threaded operations, had longer overall runtime which is a negative as well. Model 3 did result in a higher overall average of Correct Pixel Percentage in all comparisons against the first two Models along with having the shortest runtime for both Single Threaded and Multi-Threaded processing. But to the naked eye, Models 1 and 2 shine as they are able to get their pixel values close enough to the Target Image to clear identify the Target Image for instance in Figure 3.4.7 for Multi-Threaded both Models1 and 2 look nearly identical to the Target image even though they only obtained a Correct Pixel score of around 41%-42%. Even in the smaller population run in Figure 3.4.6, Models 1 and 2 still get to within 98% of the original Target Image with only one or two pixels looking out of place.

From these results, it can be observed that using Multi-Threading to break up a large image into smaller groups allowed this Genetic Algorithm to work more efficiently by parallel processing ten smaller groups rather than series processing the whole image at once. Another benefit from introducing Multi-Threading is shorter run times needed to process an entire Population. Finally, Multi-Threading produced better results in duplicating the Target Images.

In conclusion, a preference on when to use each of these Models and Threading option comes down to the user's goal. If the goal is to create the most accurate representation of a picture, Model 3 demonstrated that it will more effectively and efficiently achieve an exact or near exact copy of the Target Image. If the goal is to develop a visually close duplicate, Models 1 and 2 allow for any improvement to a pixel value that moves the Gene value closer to the Target Image. This is why Models 1 and 2 look so close to the original Target Image even though they have a low percentage of correctly matched pixels.