NOVA IMS Information Management School

Polygon based Image Recreation using Genetic Algorithms

CIFO

Master Degree Program in Data Science and Advanced Analytics



https://github.com/bialv/Poligoes/tree/main

Group "Poligões" - All members contributed equally

R20201614 – Pedro Barão

20230567 – Adriana Costinha

20230577 – Ana Filipa Silva

20230755 – Beatriz Vasconcelos

Index

Problem definition

Currently, image recreation remains a complex task for several reasons. Since images are composed of pixels, each with a range of values representing colors, this leads to a high number of individual data points that need to be precisely adjusted, making the process highly computationally intensive. Genetic Algorithms excel in handling large and complex solution spaces, managing non-linear optimization, and introducing incremental improvements that can significantly enhance image quality. Given that this challenge is still seeking solutions and considering that genetic algorithms align well with the challenges we face, we have decided to take on this task and attempt to recreate a target image.

The aim of this project was to employ Genetic Algorithms (GAs) to recreate a target image using polygon-based images. This approached was inspired by the work of Sebastian Charmot[1], more specifically when it comes to the *Random Mutate Mutation*, *Blend Crossover* and the *Delta E* Fitness Function. By focusing on GAs, we explore their capability to optimize and evolve solutions that approximate the target image as closely as possible, therefore trying to minimize the difference between the original and the recreated image. We tried different implementations (e.g. crossover, mutation, initialization, and selection operators) compared them and tried to come up with an optimal solution.

Individual and Population

For the project's development we have decided to use the *charles* library developed in class, with minor adjustments that were in line with the specific problem we aim to solve.

Individual Class

Previous attempts used random RGB values for every single pixel[2].Because each pixel had different, random colors, there was no visual coherence across the image, resulting in a highly pixelated output. To avoid this effect, we chose a polygon-based representation as it provides a larger area, reducing the high frequency of color changes that occur when each pixel is assigned a random color and thereby decreasing the noise and visual clutter that led to pixelation. Furthermore, the edges of polygons form recognizable shapes that can be easily distinguished by the human eye, making the image more harmonious. Additionally, the larger area of polygons, compared to pixels, simplifies manipulations.

The class *Individual* represents a single potential solution in the form of an image which has randomly generated polygons. Although alterations to the image are actual modifications of its underlying array, we chose to define the image as our representation. This decision was made because most of the operators apply functions that are developed to work directly with the image itself, simplifying the representation and more accurately reflecting the actions taken to achieve the solution.

Population Class

We made only minor modifications to the *Population* class from the *charles* library, it still represents a collection of individuals and drives the evolution process of the population over multiple runs and generations by applying selection, crossover, and mutation operations to produce successive generations of individuals. Regarding the modifications made, we have decided to implement different types of elitism: *inner elitism*, where if the offspring resulting from the crossover does not have equal or better fitness than the parent with the best fitness, new parents continue to be selected and *replacement elitism* where, at the end of each generation, the individual with the worst fitness is replaced by the individual with the best fitness from the previous population. These two types of elitism will be

[1] SebastianCharmot/Genetic-Algorithm-Image-Recreation: From scratch Python implementation of a genetic algorithm that recreates a target image. (2024).
[2] ahmedfgad/GARI: GARI (Genetic Algorithm for Reproducing Images) reproduces a single image using Genetic Algorithm (GA) by evolving pixel values. (2024).

further tested and analyzed. In the end Fitness values of the most fit individuals in each generation are logged and saved to a CSV file for further analysis. Throughout the project, a fixed population size of 100 was used. After testing different population sizes, this one was able to strike the balance of a size that allowed for individual diversity, while not being too computationally expensive for the statistical requirements of this project.

Fitness functions

In this specific project, the fitness function evaluates the similarity between the generated image and the target image. All fitness functions explored quantify the difference between the target and the recreated image to minimize this difference.

The **Delta E Fitness** function uses the Delta E*76 formula and operates in the CIELAB color space, which calculates and mimics how different colors are perceived by the human eye.

The **Sum of Absolute Differences Fitness** function computes the fitness based on the sum of absolute differences between the pixel values of the individual's image array and the pixel values of a target image array.

The **Euclidean Color Distance Fitness** function calculates the Euclidean distance between corresponding pixels of the individual's image and the target image. By measuring the distance between two points in color space, this method tends to capture the perceptual differences in color between two images.

The **Root Mean Square Error Fitness** function calculates the Root Mean Square Error (RMSE) distance between the individual's image and the target image, quantifying the average magnitude of the error between corresponding pixels in two images.

Selection approaches

The selection approaches were based on the ones developed in class. We used for four different selection algorithms: *Tournament*, *Ranking*, *Roulette-Wheel* and *Rank-Tournament*.
*Rank-Tournamen*t, as the name indicates, is a mixture of the *Tournament* and *Ranking* selections, selecting individuals with the highest rank within a tournament.

Genetic Operators

Mutation Operators

The mutation operators are based on introducing diversity through color, where the mean color difference serves as the function of comparison.

Scramble Patch Mutation: Scrambles the pixels of a randomly selected patch.

Random Mutation: Adds a random section in the shape of polygons to the individual's representation, with both their color and shape being random.

Swap Polygons Mutation: Randomly selects two polygon-shaped areas and swaps their positions in the individual's representation multiple times.

Swap Polygons Color Mutation: Randomly selects two polygon-shaped areas with different average colors and swaps their positions in the individual's representation multiple times.

Interpolate Color Mutation: Similar to random mutation, but instead of the color of the polygon-shaped area being random, it's a 50/50 blend of part of the original image and a new random color.

Crossover operators

Since our problem aims to mimic the color images of the target image as closely as possible, we have developed crossover operators that focus either on combining colors from the parents or on combining segments from their image arrays.

Blend Crossover: This crossover method blends two parent images together to create a new child image. A random blending factor is chosen between 0 and 1 to control the mix of parent features.

Per-Channel Crossover (Per-Channel XO): This crossover method creates a child image by combining the RGB channels from two parent images. For each channel, a random choice is made to take the channel from either parent.

Per-Channel Crossover with Normalization (Per-Channel XO-2): This crossover method is similar to the previous one, but with an added step to normalize the color channels. Normalization ensures consistent intensity levels across images, potentially enhancing the quality of the child image.

Two-Point Crossover (Two-Point XO): This crossover method combines segments from two parent image arrays to produce a new child image. Two random crossover points are selected, and segments between these points are swapped between the parents to create the child.

Patch Crossover (Patch XO): This crossover method creates a child image by copying a random rectangular patch from one parent and inserting it into the image of the other parent.

Testing Combinations

Fitness

For our genetic algorithm, each fitness function has different calculation methods and therefore different metrics, making direct graphical comparison unfeasible. With this in mind, we decided to use visual performance as the decision factor to account when deciding which fitness function should be selected. Although every single fitness function attempts to evaluate image similarity trough color, the Delta E Fitness function outperformed the others visually (Figure 1), once the Delta E*76 calculation uses the CIELAB color space, which is made to match how the human eye perceives colors. Therefore, we used this fitness function as the basis for testing the different combinations we present below.

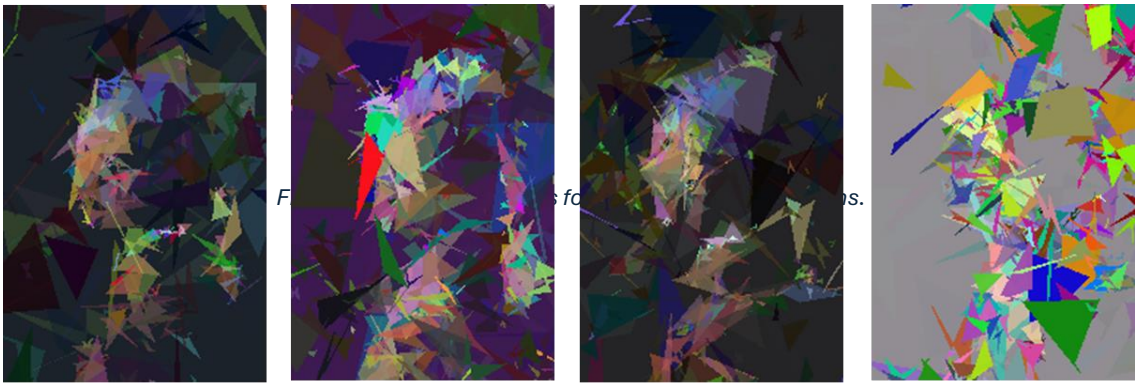

Figure 1- Image recreations for different fitness functions.

Selection

Since the outcomes of GAs can vary significantly between runs due to random factors such as the initial population, mutations, and crossovers, we conducted 15 independent runs, each for 1500 generations, and applied the median to smooth out these variations.

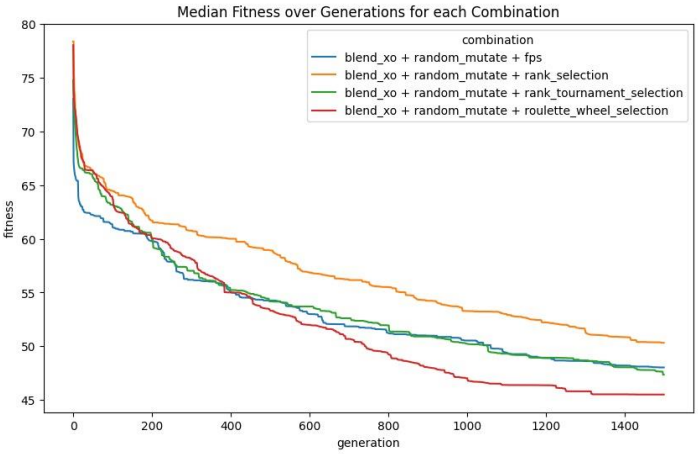

Figure 3 - Median Fitness over Generations for each Selection method

Firstly, we investigated which selection method led to better results, using a fixed mutation and crossover due to the fact that using all combinations would be very computationally expensive.
As shown in Figure 2, the R*oulette wheel* selection outperforms the others in achieving the lowest fitness value. Therefore, we have decided to use this selection algorithm when testing different combinations of crossover and mutation.

Elitism

As mentioned previously, two types of elitism were implemented and further analyzed: inner elitism and replacement elitism.

Replacement elitism is almost always used because it ensures that the best solutions from a generation are not lost due to random selection or crossover. Therefore, the population's overall fitness does not decrease from one generation to the next, potentially allowing the algorithm to converge faster towards an optimal solution.
Inner elitism is an approach that aims to create an offspring that exhibits equal or better fitness than that of their parents. This approach encourages the algorithm to focus more on areas of the solution space that are already known to yield good results. Thus, the algorithm can exploit these known areas while still exploring new possibilities through the crossover of successful genes.
Although these strategies seem to introduce only benefits, they raise concerns about potentially not maintaining sufficient diversity within the population and thus possibly leading to premature convergence at local optima, especially in the case of inner elitism. Because of this, we tested both the inclusion and exclusion of each type. From Figure 3,4 and 5, it is possible to observe that using both types of elitism leads to better fitness and lower variation, and therefore, were used to further test different operator combinations.

Mutation

In order to explore the mutation operator's performance, we conducted experiments with a fixed crossover operator (*Blend crossover*) and the five different mutation techniques. For the *Scramble Patch Mutation*, *Swap Polygons Mutation*, and *Swap Polygons Color Mutation* we observed no differences between each generated image when compared to the other mutations *Random Mutation* and *Interpolate Color Mutation* (Figure 6) indicating that the population might have lost genetic diversity too early in the run, leading to premature convergence. This lack of variation raised questions regarding the effectiveness of the crossover operation, even though it was actively being used.

Given our goal to minimize the mean color difference between an individual image and a target image, we reached a crucial conclusion. These three mutation operators did not introduce sufficient new information. They merely rearranged existing colors within the individuals' representations, rather than introducing new ones. Consequently, this led to an offspring that essentially retained the same characteristics as the parents (color) during the crossover phase, leaving the image unchanged.

To ensure diversity and drive convergence towards an optimal solution, it became clear that introducing new colors was essential. This requirement limited our effective mutation options to techniques like Random Mutation and Interpolate Color Mutation. These were the only methods we further tested to identify the most effective combinations.

Genetic Operator Combinations

After setting the selection method (R*oulette Wheel*) with fixed probabilities (0.80 for crossover and 0.15 for mutation) we proceeded to test combinations of all our crossovers with each of the mutations described above (Figure 7, Figure 8, Figure 9). The best-performing combination was determined based on three factors: the fitness value, its variance through runs and the convergence of the solution. Since this is a minimization problem, the optimal solution is the one that exhibits the lowest fitness value, indicating a smaller difference between the target and the generated image. Additionally, a low variance in fitness scores across multiple runs shows reliability, as it produces stable and dependable results.

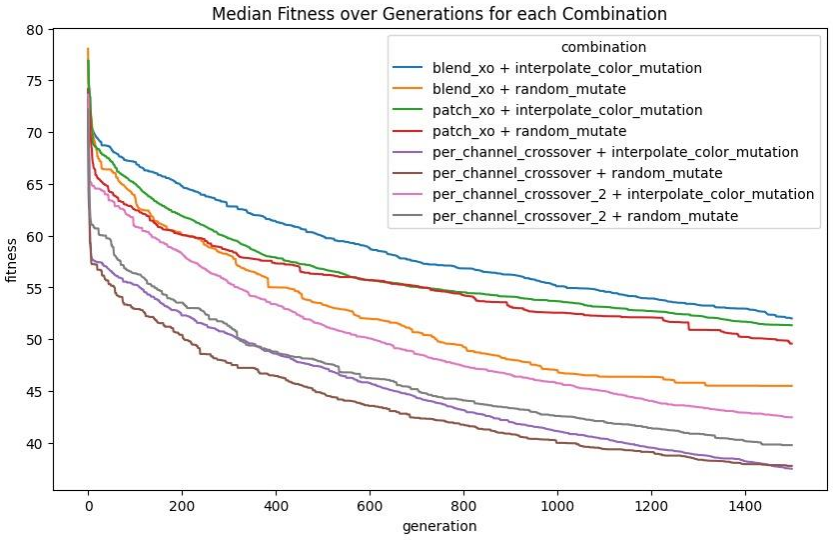

Figure 7 - Median Fitness over Generations for different combinations of Genetic Operators.

Looking at the Figure 7, 8 and 9 it is possible to see how different operators affect the convergence of our GA.

Different combinations of crossover and mutation operators exhibit different rates of convergence. *Per channel Xo with random mutate*, *Per channel Xo-2 with random mutate* and *Per Channel Xo with Interpolate Color Mutation show* a rapid decrease in fitness early on, while *Blend Xo with interpolate color mutation, Patch Xo with interpolate color mutation and Per Channel Xo-2 with interpolate color mutation* show more of a gradual decrease.

When it comes to convergence stability the lines representing *Per Channel Xo with random mutation, Per channel Xo with interpolate color mutation and Per channel Xo 2 with random mutation* are relatively smooth, suggesting stable convergence, while the remaining have more fluctuations in fitness.

The combination of per-channel crossover and random mutation demonstrated a swift decline in fitness scores at the initial stages. This combination not only converged more rapidly to a lower fitness value but also exhibited a lower variance (Figure 8, Figure 9). These characteristics position it as a promising candidate for the most optimal solution.

Lastly, we proceed to fine-tune the probability parameters for our best combination, testing 0.75, 0.80, 0.85 and 0.90 for the crossover probability and 0.10 and 0.15 for the mutation probability.

As shown on Figure 10 and 11 the parameters that were best for our combination were 0.80 for crossover probability and 0.15 for mutation, with an example of the recreated image obtained in Figure 12.

Conclusion

This project took on the challenge of recreating images through polygons, considering what was tried in similar projects and how we could innovate the problem. We successfully experimented with a varied set of crossovers, mutations and selections that showed promising and interesting results, with the choice of each operator heavily influencing the performance of the algorithm. In this case, the Per-Channel Crossover with a Random Mutation performed the best in all accounts. Although we don't have concrete proof on this subject, we have discussed why this combination appears to be the most effective. The use of the individual color channels makes it possible to optimize each color dimension separately and generate new color configurations, while preserving important visual features from the parent images. Since it only alters the color values and does not rearrange the pixels themselves, the fundamental structure and layout of the image remain intact. This allows for a smoother reconstruction of the target image, where the colors are consistently improving with each generation while new features are added at a slower pace by the random mutations.

However, our work also revealed some challenges. Working with images is an intensive computational task that limits not only the size of the populations, but also the size of the images used. Polygons struggle in images with a high level of detail because the size of the polygons used influences the capture of the image's features. Smaller polygons can capture more detail, enhancing the image's sharpness and precision. Conversely, larger polygons tend to smooth out details, resulting in an image that appears less textured but more uniform.
Finally, as expected and discussed above, the biggest difference in the image results was noted in the variance of crossovers and not in the mutations.

Furthermore, another challenge was related to the visual aspect. Although we focused on both technical and visual assessments, this field still requires further study and analysis. Visual comparisons can be misleading, as they are subjective and often involve comparing just one solution rather than the median of multiple runs. As exemplified in Figures 12 and 13, although the combinations for Figure 12 performed better according to our fitness function, Figure 13 visually appears to be superior.

Looking forward, there are several paths to follow for improvement and exploration. Conducting more runs of the code would provide more statically significant results and expanding combinations tried could also find a better set of parameters to use. Another untouched area in this project was population diversity. While the use of elitism significantly improved the results of the algorithm, it would be unwise to discard the idea that enforcing such a strict form of elitism damages the diversity of the population and can lead to premature convergence.
Lastly, as referred above, the visual aspect should be further studied.

Appendix

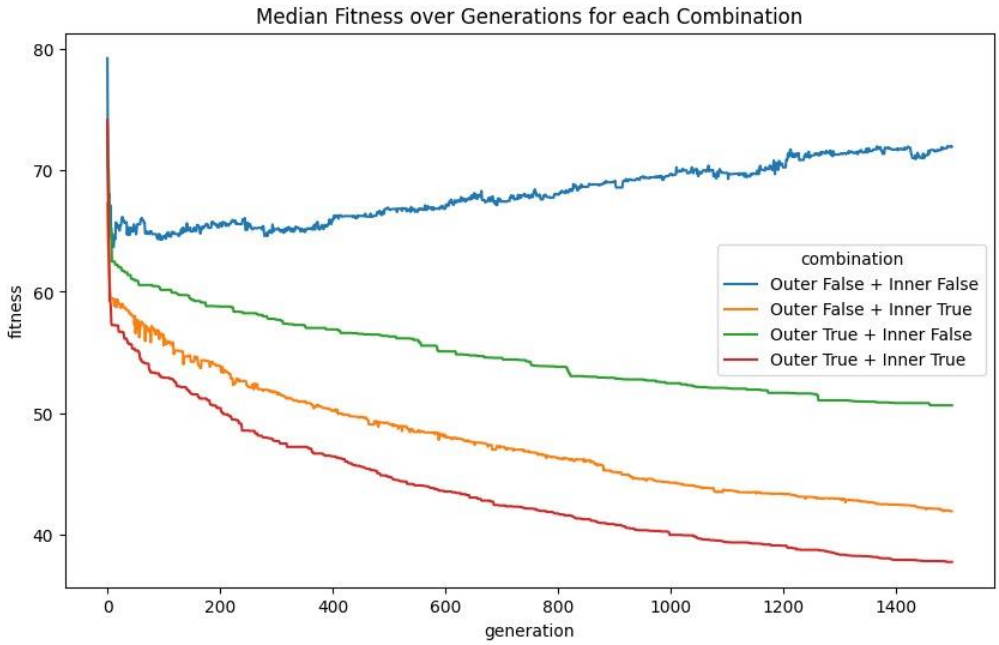

Figure 4 - Median Fitness over Generations for different combinations of Elitism.

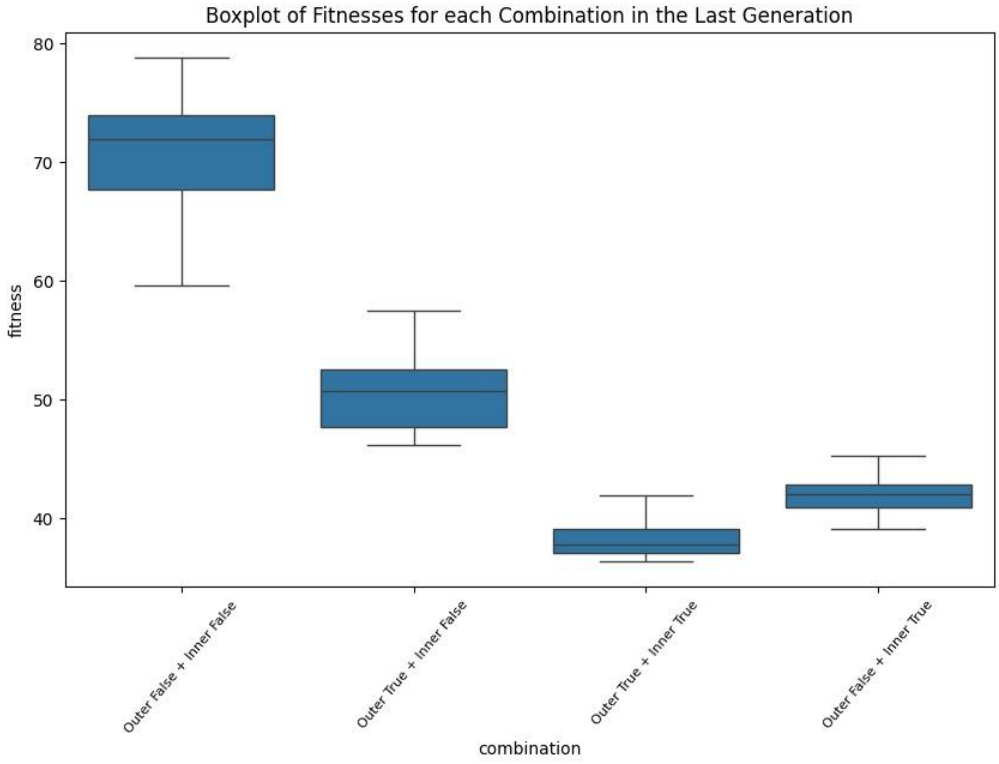

Figure 5 - Boxplot of the fitness variation for the last Generation for different combinations of Elitism.

	Combination	Variance of Fitness
3	Outer True + Inner True	2.199031
1	Outer False + Inner True	2.611556
2	Outer True + Inner False	10.057105
0	Outer False + Inner False	26.227328

Figure 5 - Fitness variation for the last Generations for different combinations of Elitism.

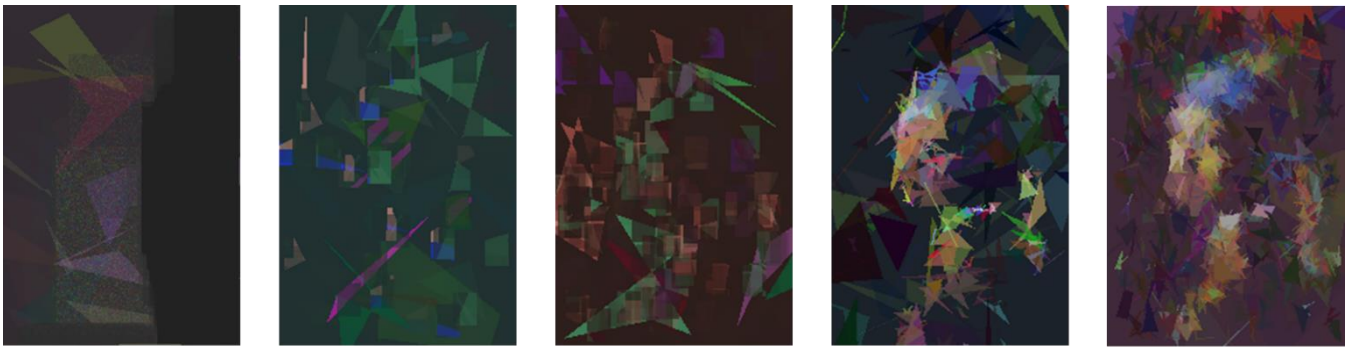

Figure 6 - Image recreations for different mutation operators.

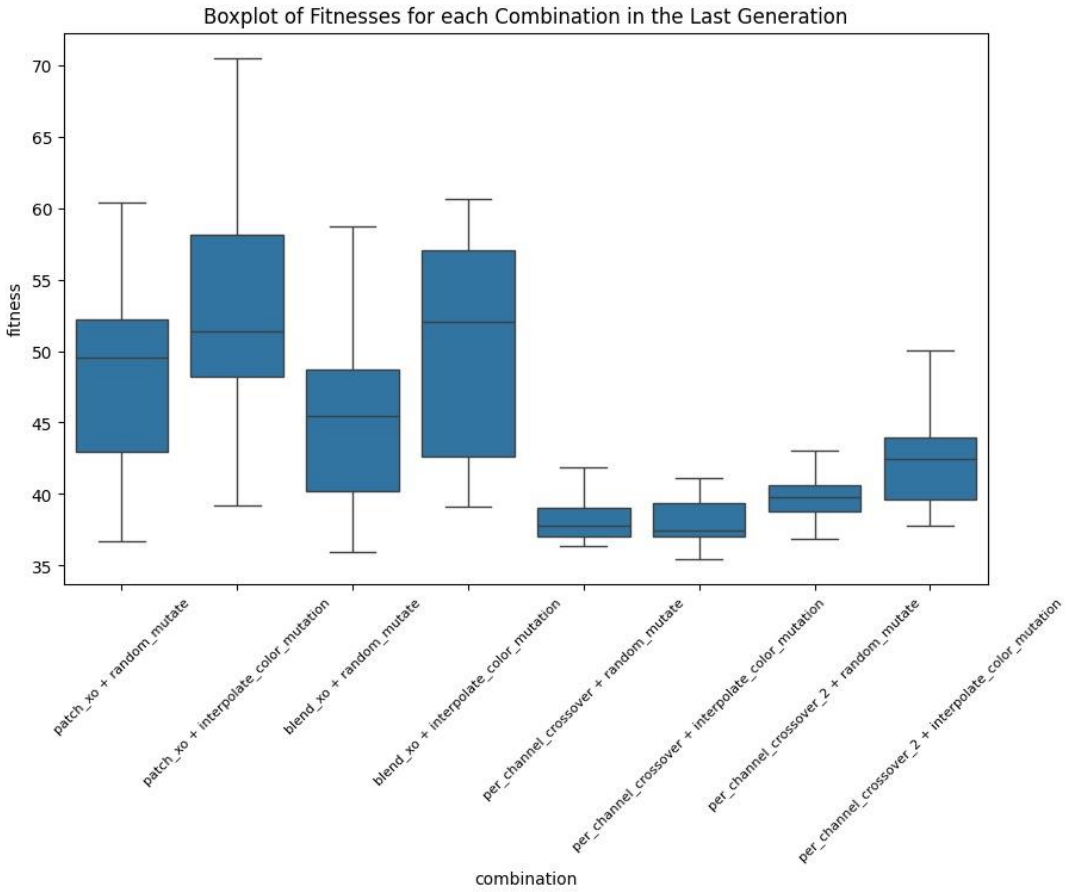

Figure 8 - Boxplot of the fitness variation for the last Generation for different combinations of operators.

	Combination	Variance of Fitness
5	per_channel_crossover + random_mutate	2.199031
4	per_channel_crossover + interpolate_color_muta...	2.467392
7	per_channel_crossover_2 + random_mutate	2.809539
6	per_channel_crossover_2 + interpolate_color_mu...	13.005323
3	patch_xo + random_mutate	37.425397
1	blend_xo + random_mutate	37.767581
0	blend_xo + interpolate_color_mutation	58.544510
2	patch_xo + interpolate_color_mutation	88.268894

Figure 9 - Fitness variation for the last Generation for different combinations of genetic operators

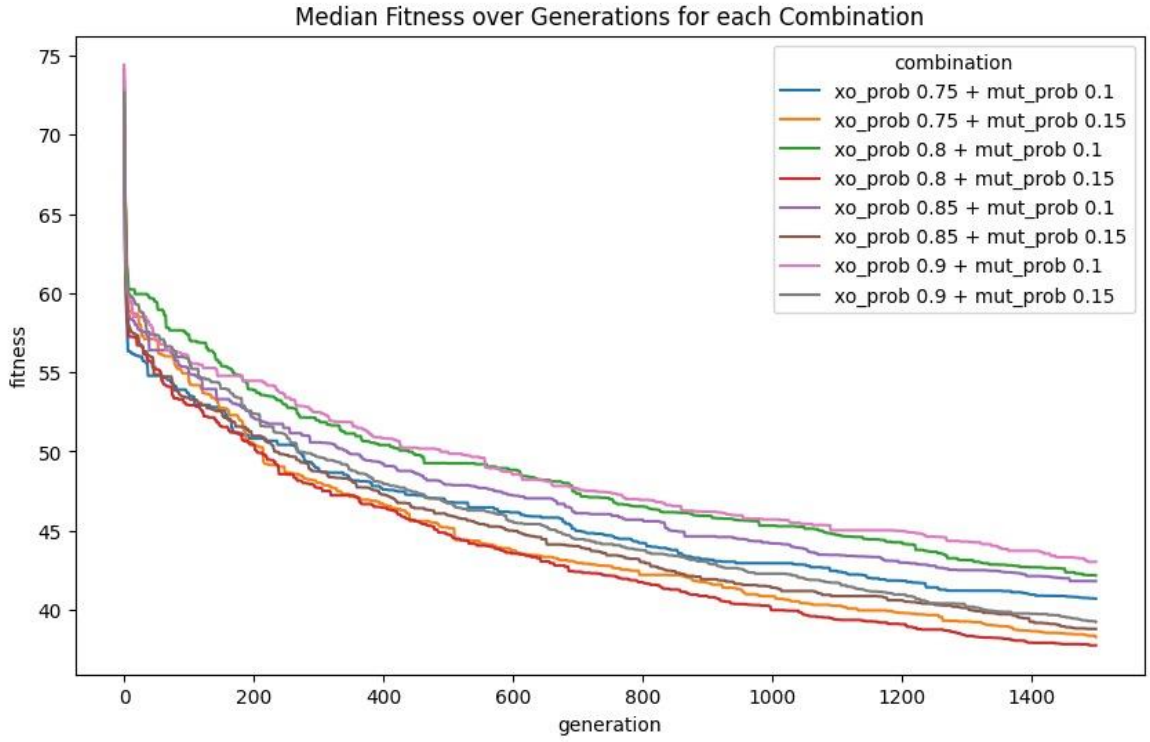

Figure 10 - Median Fitness over Generations for different combinations of mutation and crossover probabilities.

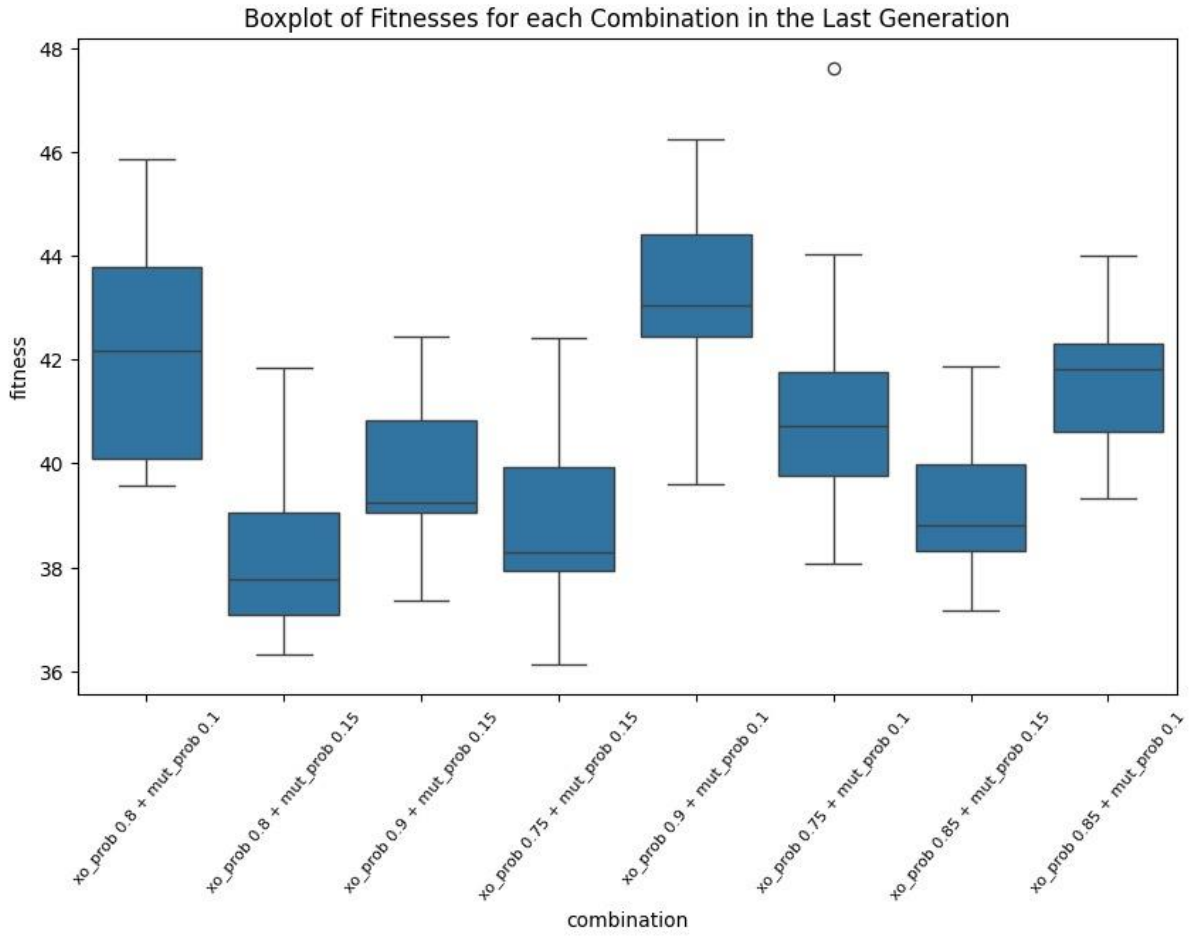

Figure 11 - Boxplot of the fitness variation for the last Generation for different combinations of crossover and mutation probabilities.

Figure 12 – Image obtained from the combination Per Channel Xo with Random Mutation.

Figure 13 – Image obtained from the combination Per Channel Xo with Interpolate Color Mutation.