03/06/2024

Image Recreation

GROUP PROJECT CIFO 2023/2024

Fabian Romero |20230503
Jannik Himmelsbach |20230550
Taha Ben Attia |20230742
Alex de Figueiredo |20201607

1. Project Overview

Group Name: Kafa Lesh
Github Repository URL: https://github.com/tahabenattia/Cifo-project
Link: tahabenattia/Cifo-project (github.com)

Our project aims to recreate a target black and white image using Genetic Algorithms (GAs), employing the Charles Genetic Algorithm Library we developed during the semester. We focused on implementing and optimizing GAs to generate an image that closely matches the target. The chosen target image is based on suitability for our GA-based recreation. Below we will discuss the reasons behind our key decisions, the impact on results, and potential improvements for future work.

2. Genetic Algorithm Design

2.1. Representation of Individuals

In this project, individuals are represented as a one-dimensional array with a length of 40,000. This corresponds to the input target image shape of 200x200 pixels. Each element in the array represents a pixel in the image, and valid values for each pixel range from 0 to 255, representing the grayscale intensity of the pixel.
This one-dimensional array representation is chosen for its simplicity and direct mapping to the 200x200 pixel image, ensuring ease of genetic operations. It allows flexible conversion back to the image format for accurate fitness evaluation against the target image. Additionally, this representation offers flexibility for implementing crossover or mutation methods that may require temporarily treating the data as a two-dimensional structure, facilitating more sophisticated genetic manipulations.

2.2. Fitness Functions

To evaluate the fitness of individuals in our minimization problem, several fitness functions were explored. The Mean Squared Error (MSE) method calculates the average of the squared differences between the individual and target pixel values, providing a measure of overall deviation. The Mean Absolute Error (MAE) method computes the average of the absolute differences, offering an alternative metric that is less sensitive to outliers. Additionally, the Structural Similarity Index (SSIM) method was used to assess image similarity based on luminance, contrast, and structure, with higher SSIM values indicating greater similarity. Each of these fitness functions was adapted to ensure lower values represent better fitness in the context of our minimization objective.

2.3. Selection Methods

In our project, we implemented and tested two selection methods: Fitness Proportionate Selection (FPS) and Tournament Selection. In the context of minimization, FPS selects individuals based on their fitness scores, giving higher probabilities to those with lower (better) fitness values. Tournament Selection involves randomly selecting individuals to compete in tournaments, with the fittest (lowest fitness) individual from each tournament chosen for reproduction.

2.4. Crossover Operators

We investigated various crossover methods to generate offspring from parent individuals, aiming to merge beneficial characteristics and improve the population's overall fitness.

2.4.1. Two Point Crossover

Two-Point Crossover involves selecting two points along the parents' representations and exchanging the segments between these points to generate offspring. This process mimics genetic recombination and is commonly used in genetic algorithms for image generation tasks. By swapping segments, it introduces diversity while preserving certain features present in both parents.

2.4.2. Smooth Two Point Crossover

Additionally, we've implemented a variant called Smooth Two-Point Crossover, which not only performs the two point crossover but also applies Gaussian smoothing to the boundary regions of the exchanged segments. The smoothing process helps in reducing abrupt changes and artifacts in the offspring images, producing more visually coherent results.

2.4.3. Block Uniform Crossover

Lastly, we explored Block-Based Uniform Crossover, which divides the parent images into blocks of a specified size and randomly swaps these blocks between the parents. By considering larger, uniform segments of the image, this method introduces a different level of genetic mixing, potentially capturing more significant patterns and structures from the parents.

2.5. Mutation Operators

To introduce variability and improve the evolutionary process, the following four mutation methods were implemented.

2.5.1. Salt and Pepper Mutation

The salt-and-pepper mutation method introduces noise to the image by randomly altering a proportion of the pixels to either black (0) or white (255).

2.5.2. Random Shape Mutation

With the random shape mutation, a random geometric shape (circle, rectangle, or triangle) is added to a randomly chosen region of the image. The shape type and color (black or white) are randomly chosen. Moreover, for each shape type, specific parameters (such as center and radius for circles, or corner points for rectangles and triangles) are randomly generated within predefined ranges.

2.5.3. Edge Detection Mutation

The edge detection mutation method enhances the edges within the image. It employs a Sobel filter, a widely used method for edge detection in image processing. The Sobel filter calculates the gradient magnitude of the image intensity at each pixel, highlighting regions of rapid intensity change, which typically correspond to edges. The function applies the Sobel filter along both the horizontal and vertical directions to obtain edge magnitude information. These edge features are then added back to the original image, enhancing the contrast along edges and accentuating structural details.

2.5.4. Inversion Mutation

Lastly, the inversion mutation function reorders a portion of the representation, thereby introducing local changes in pixel order. It randomly selects a segment of the individual's representation and reverses the order of its elements. This reversal disrupts the existing structure and promotes exploration of new configurations within the population. The mutation is applied directly to the individual's representation, flipping the selected segment in place.

2.6. Elitism

We evaluated the evolutionary process both with and without elitism in the context of minimization. Elitism preserves the best-performing individuals across generations by protecting them from crossover and mutation.

2.7. Initialization Methods

Two initialization methods to create diverse starting populations were explored.

2.7.1. Random Pixel Initialization

The first method, Random Pixel Initialization, assigns random values between 0 and 255 to each pixel in the image, resulting in a highly varied initial population.

2.7.2. Random Pattern Initialization

The second method, Random Pattern Initialization, generates individuals using one of three distinct patterns: checkerboard, stripe, or circle, chosen with equal probability. The checkerboard pattern places random squares in alternating colors, while the stripe pattern creates vertical or horizontal stripes with varying widths. The circle pattern randomly places circles of different sizes across the image. Our intention was to potentially provide a richer set of starting points for the evolutionary process compared to purely random initialization.

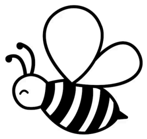
3. Evaluation Metrics

To evaluate the effectiveness of our genetic algorithm, we utilized several key metrics throughout the evolutionary process. We tracked the best fitness values, average fitness values, and population diversity across generations. The best fitness value indicates the lowest fitness score achieved, reflecting the closest match to the target image. The average fitness value provides an overall measure of the population's performance, highlighting general trends and improvements. Population diversity, measured as the standard deviation of fitness values, was monitored to ensure genetic variability and avoid premature convergence. These metrics were plotted over generations to visualize evolutionary progress. Additionally, we captured visual snapshots of the best individual at specific checkpoints during the evolution, allowing us to qualitatively assess the image recreation process and document improvements over time.

4. Results and Analysis

The implementation process involved systematic experimentation with various configurations to identify the optimal setup for image recreation. To find the best configuration, a base configuration was established, with which the impact of changes can be analyzed. The probabilities for crossover operations and mutations were tuned beforehand to its best performance.

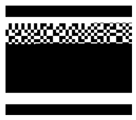
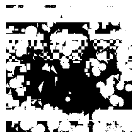
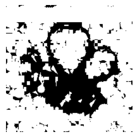
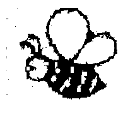
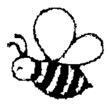
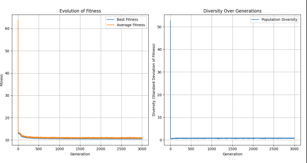
Target Image:



Base configuration for comparison and impact analysis:

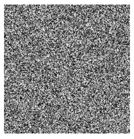
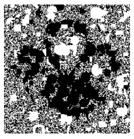
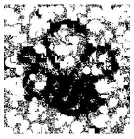
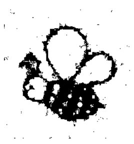
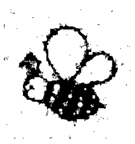
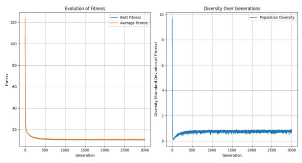
Population Size: 50	Generations: 3000
Initialization Method: Random Pattern	Fitness Function: Mean Absolute Error (MAE)
Crossover Probabilities: 0.6 (first method), 0.4 (second method)	Mutation Probabilities: 0.2 (first method), 0.15 (second method)
Crossover Methods: Two-Point Crossover, Block Uniform Crossover	Mutation Methods: Random Shape Mutation, Salt-and-Pepper Mutation
Selection Method: Tournament Selection	Elitism: Enabled

Base Configuration Best Individual Results:

Generation:	1	300	600	1500	3000	Metrics
Base Configuration						

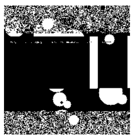
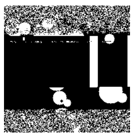
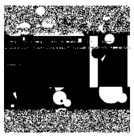
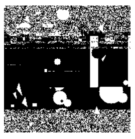
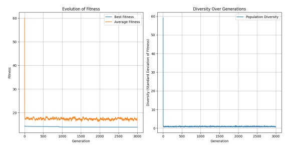
In the following configurations, the very left column of each table, specifies the change in configuration made compared to the base configuration.

4.1. Impact of Different Initialization Methods

Generation:	1	300	600	1500	3000	Metrics
Random Pixel Initialization						

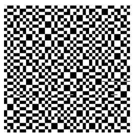
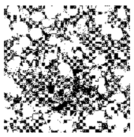
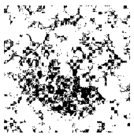
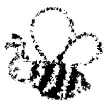
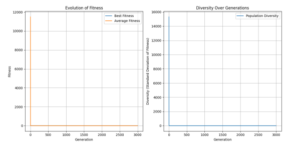
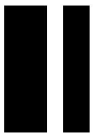
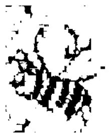
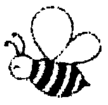
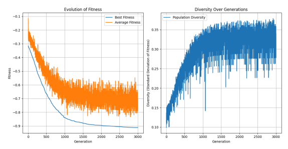
Observation: Less clean results.

4.2. Impact of Different Selection Methods

Generation:	1	300	600	1500	3000	Metrics
FPS Selection Method						

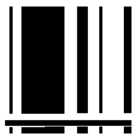
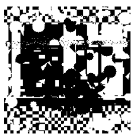
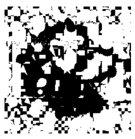
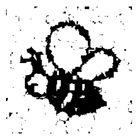
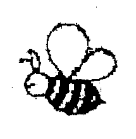
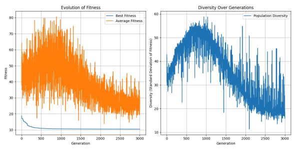
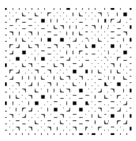
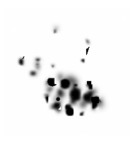
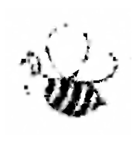
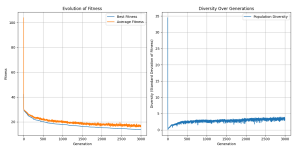
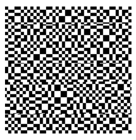
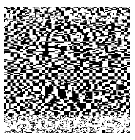
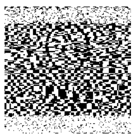
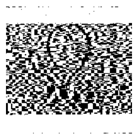
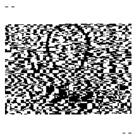
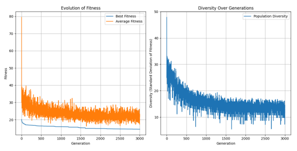
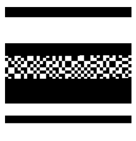
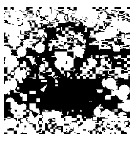
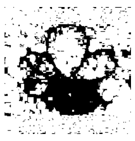
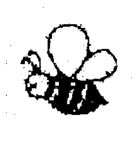
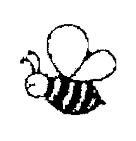
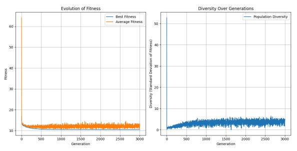
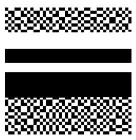
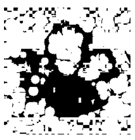
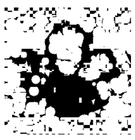
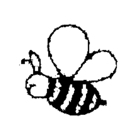
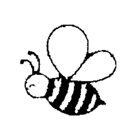
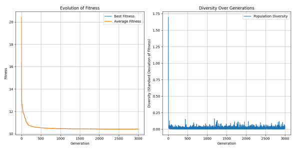
Impact: No convergence to target picture.

4.3. Impact of Different Fitness Functions

Generation:	1	300	600	1500	3000	Metrics
MSE						
SSIM						

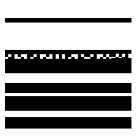
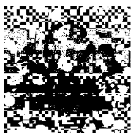
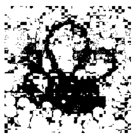
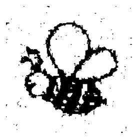
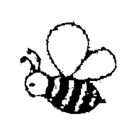
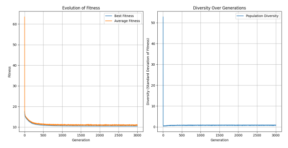
Observation: MSE with worse results, SSIM with slightly better results but longer runtime (higher computational cost).

4.4. Impact of Different Crossover and Mutation Combinations

Generation:	1	300	600	1500	3000	Metrics
Crossover 2: Smooth Two Point Crossover						
Crossover 1: Smooth Two Point Mutation 2: Edge Detection						
Mutation 1: Edge Detection Mutation 2: Inversion						
Mutation 2: Inversion						
Crossover: Just Two Point xo for both Mutation: Just Random Shape for both						

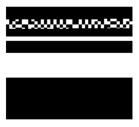
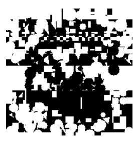
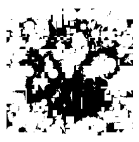
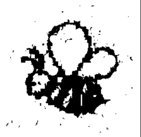
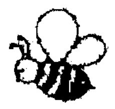
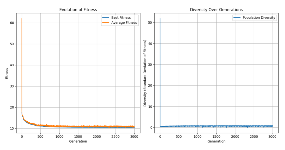
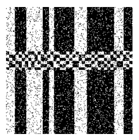
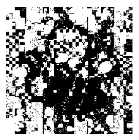
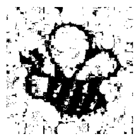
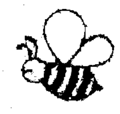
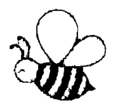
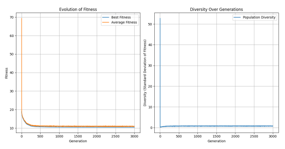
Observation: No significant improvements compared to the base configuration.

4.5. Impact of Elitism

Generation:	1	300	600	1500	3000	Metrics
No Elitism						

Observation: Worse results without elitism.

4.6. Impact of Different Population Size

Generation:	1	300	600	1500	3000	Metrics
Population size = 25						
Population size = 100						

Observation: A higher population brings better results, but not significantly compared to the base configuration, while having more computational costs.

4.7. Conclusion of Impact Analysis

The base configuration, which included a population size of 50, 3000 generations, random pattern initialization, specific crossover and mutation probabilities, tournament selection, and enabled elitism, served as a benchmark for comparison.
Initialization Methods: The random pattern initialization consistently outperformed the random pixel initialization, which produced less clean results.
Selection Methods: Tournament selection proved more effective than fitness proportionate selection (FPS), as FPS did not converge to the target image.
Fitness Functions: The mean squared error (MSE) function yielded worse results compared to mean absolute error (MAE), while the structural similarity index (SSIM) showed slightly better results but at a higher computational cost.
Crossover and Mutation Combinations: Different combinations, such as smooth two-point crossover and edge detection mutation, did not show significant improvements over the base configuration. Using just two-point crossover and random shape mutation for both crossover methods and mutations also did not yield better results. Other combinations resulted in worse outcomes or did not even converge.
Elitism: The presence of elitism was beneficial, as its absence resulted in significantly worse outcomes.
Population Size: Increasing the population size to 100 provided better results but with slightly higher computational costs, while reducing the population size to 25 led to worse results.

Overall, the base configuration struck a balance between performance and computational efficiency. The results emphasized the importance of structured initialization methods, tournament selection, and elitism in achieving better image recreation outcomes. Adjustments to population size and fitness functions offered some improvements, but these came with trade-offs in computational resources.

4.8. Best Configuration

The best-performing configuration, based on the previous analysis is chosen to be:

Population Size: 100	Generations: 3000
Initialization Method: Random Pattern	Fitness Function: SSIM
Crossover Probabilities: 0.6 (first method), 0.4 (second method)	Mutation Probabilities: 0.2 (first method), 0.15 (second method)
Crossover Methods: Two-Point Crossover, Block Uniform Crossover	Mutation Methods: Random Shape Mutation, Salt-and-Pepper Mutation
Selection Method: Tournament Selection	Elitism: Enabled

5. Improvement Suggestions

Our project aimed to recreate black and white images using a genetic algorithm, and we succeeded. However, there's significant potential for improving the model to recreate colorized images. Extending it to color images would involve several key modifications to our current architecture. If you want to check the code that targets the colorized images, please refer to the fabian-pocs branch in the GitHub repository[1].

The primary difference between black and white and color images is the number of channels. Color images typically have three channels (RGB), but for better human color perception, we considered using the HSV (Hue, Saturation, Value) color space, which aligns more with how we perceive colors, making it suitable for our fitness function.

To accommodate color images, the fitness function must evaluate more than one channel. Our approach was to use the HSV color space, where the fitness function calculates the difference between the target and generated images in terms of hue, saturation, and value. This approach helps capture the perceptual differences and similarities between the images better.

With the shift to color images, we also needed to adapt our genetic operations—mutation and crossover. For mutation, we introduced a function that applies changes to the HSV channels independently, like adjusting the hue, saturation, or value of randomly selected pixels and adding random shapes with varied colors to maintain diversity. For crossover, we extended our functions to handle three channels by ensuring the crossover points and blocks applied to all three channels simultaneously, preserving color coherence in the offspring.

We conducted preliminary experiments using a population of 60 individuals, a tournament selection with low pressure (tournament size of 3), and ran the algorithm for 2000 generations. The results were promising but not entirely accurate. The generated images showed significant potential, indicating that with further refinement, we could achieve high-quality colorized images.

To optimize the evolution process for color images, we recommend:

1. Hyperparameter tuning: Experiment with different population sizes, selection pressures, mutation rates, and crossover probabilities to find the optimal configuration.
2. Advanced genetic operations: Develop and test more sophisticated mutation and crossover strategies specifically designed for color image generation.
3. Extended generations: Increase the number of generations to allow the algorithm more time to converge towards an accurate representation of the target image.
4. Dynamic fitness scaling: Implement adaptive mechanisms for fitness scaling that adjust based on the algorithm's progress to enhance convergence.

By pursuing these improvements, we believe it's possible to achieve a robust and efficient model capable of accurately recreating color images, extending the applicability and effectiveness of our genetic algorithm.

Division of Labor: In our group, everyone contributed to all aspects of the project, including implementation, experimentation, and analysis, ensuring a collaborative and comprehensive approach.

References

[1] Group Kafa Lesh, NOVA IMS. (2023). Repository’s branch hosting the code for colorized image generation using a genetic algorithm. Retrieved from https://github.com/tahabenattia/Cifo-project/tree/fabian-pocs