Exercise 3: Genetic Algorithm Variations

Due date: *October 30th, 2025 (push to GitHub repo before the lecture)*
General Requests: *Same as before.*

3.1 Genetic Algorithm Variations

Last exercise, we explored classical Genetic Algorithms: how to encode your problem, how to implement genetic operations, etc. This time, we'll try to improve our approach[3, Chapter 8] [1, Chapters 4-5][2, Chapter 3].

- use your genetic algorithm implementation. use any image in grayscale (e.g., this one), resize it to 16×16 pixels for starters (increase later).
- use a **suitable representation** of solution candidates for this problem.
- think about **initialization**: can you do better than random initialization?
- consider the **parent selection**, **crossover**, and **mutation strategies** you used before. implement one of the discussed strategies, each[1].
- regarding **survivor selection (innovation)**: how aggressively did you 'kill off' ancestors? try to improve your previous approach by implementing one of the discussed approaches
- for the **fitness function**, use pixel-wise square error with the example image or a metric of your choice[2]. if necessary, normalize your metric somehow to make results between different image sizes comparable.
- implement a **termination criterion** of your choice; if you use a threshold, trace the number of fitness function evaluations (as discussed in the second chapter). store the best individual and its fitness for each generation.

In contrast to the minimal problem dimension (four) of last time, we now try to optimize *every single pixel* in the image, making it a high-dimensional optimization problem[3].

3.2 Analysis and Reporting

Add a streamlit page on Genetic Algorithms to the documentation you created for the last exercise.
Your final documentation webpage should contain the following sections:

1. **Introduction** - Overview of possible algorithm variations: basics, strengths/weaknesses, complexity, ...
2. **Methods** - Describe your implementation: why did you select these specific strategies, pros/cons, relevant parameters/functions, critical design choices, ...
3. **Results** - Display results obtained via Genetic Algorithm (best individual you found, fitness of the best individual over generations, computation time, etc.);
4. **Discussion** - Analyze your findings: expected versus unexpected results, solution quality, efficiency, variations with parameter choices, etc. Talk about exploration versus exploitation and the influence the single strategies have on it. What are limitations or possible improvements you could think of, and why?

The documentation doesn't need to be absurdly long, but it should give a comprehensive overview of the topic that you can use later on to study for the exam.

References

[1] A. Eiben and J. Smith, Introduction to Evolutionary Computing. Springer, 2003.
[2] X. Yu and M. Gen, Introduction to Evolutionary Algorithms. Springer, 2010. [Online]. Available: https://jainakshay781.wordpress.com/wp-content/uploads/2017/12/introduction_to_evolutionary_algorithms-184996128x.pdf
[3] D. Simon, Evolutionary Optimization Algorithms. Wiley, 2013. [Online]. Available: https://research-1ebsco-1com-1195qzf320241.perm.fh-joanneum.at/c/kofjhs/search/details/4sh2uyq6wn?db=nlebk&db=nlabk

[1] Once again, good first choices for population size, mutation rate and crossover rate might be population_size $\in [20, 100]$, mutation_rate $\in [0.01, 0.1]$, and crossover_rate $\in [0.6, 0.9]$, but this depends on the specific problem at hand.
[2] Cosine Similarity often mentioned in the context of RAG, Structural Similarity from image processing, and optimal transport measures such as the Earth Mover's (Wasserstein) Distance might work nicely as well.
[3] For each single pixel, a process just like the one you saw in the 1D benchmark functions (square or sinusoidal functions) of assignment one happens. We don't plot this separately for thousands of pixels, though, and just look at how the image approximation changes across generations.