# Introduction to AI (IAI)

*Assignment 2. Innopolis University, Spring 2020*

**Name**: Hany Hamed
**Group Number**: BS18-06
**Professor**: Joseph Alexander Brown
**TA**: Nikita Lozhnikov
**GitHub repository link** (will be public after passing the submission time)

## Contents

IAI, Innopolis University, Spring 2020
Author: Hany Hamed

## Part1: Description & Clarifications

- Task Description: The task is to reproduce an image that is similar to the input image as a target image in an artistic way using Evolutionary algorithms.
- Task's Constraints:
  - Input image: 512x512 pixels
  - Generated image: 512x512 pixels
  - The algorithm is based on the usage of the Evolutionary algorithms.
  - No usage for external libraries that related to evolutionary algorithms.
- Solution insight: Personally, I like the concept of having images that consists of smaller images which means when you zoom in you will see small images and when you zoom out you can see the overall image as different images that consists of these small images. In my opinion, I think that this sort of images is piece of art that keep the colors and the overall image without big changes. Moreover, we can think in the task that we divide the target image into smaller pieces and if we get the best generation for these small images we will get the best image in that artistic way.
- Another idea for the solution but was not implemented is to use some art called "Arabesque" to create the small tiles/images, as the idea for the creation of the images from smaller real images was more interesting from my side, from python side I have found a way to draw it. This way could be implemented further for more artistic way.
- There is a README.md inside the folder of the submission that includes the steps to prepare and run the algorithm.
- Multiple tools has been implemented to the purpose of preparing the input and the dataset images.
- The structure of the codes is as following:
  - input directory: has the input images.
  - output directory: has the output of the generated images.
  - src directory: contains all the source codes that has been used
    - Extra tools directory: has the tools that is used for resize, change the format of the input image, ...etc.
    - Evolutionary Algorithm directory: includes EA.py that has EA class that contain all the functionalities for the algorithm (fitness, crossover, mutation, termination, training, generation functions), and main.py that has the main running code for the assignment.

    * Shared directory: includes the utils.py that contains the useful functions (e.g. changing from PIL.Image to numpy and the opposite, read and write images and gif for animation, preview the image, read the small images of the dataset from their directory, and Image class that is used to describe the member of the generated population and store the indexes of the small images as matrix to be the genes of the member (it contains construct_img() function that concatenate the small images based on the indexes that was provided and get the big image)
  - assets directory: that stores the scripts for downloading the dataset and the dataset itself.

## Part2: Solution Description

- The solution is based on reproducing the input images using small images (8x8 pixels) which means the big (generated) image will consist of (64x64) small images as the generated image is (512x512) pixels.
- To get these small images that can be a set of possible images that the generated image can consist from, it is a must to have a big set of images to ensure that the generated image will be not that different than the input image. For the previous reason, there was many solutions to get these solutions:
  (a) Use datasets that have small images, and my selection was on the training dataset CIFAR, however, it required to download it (I have used this script) and then resized as the original dataset is (32x32 pixels) which will not generate the exact image as there will be only (16x16) small images. It contains 50,000 images in the training dataset.
  (b) The other option is to use my personal images from my google drive as I have many photos but this will take more time and the first option is better as it can be applied to any dataset
- To ensure that the set of images that I have will be sufficient to create the big image, it has been decided to create a baseline script that is based on brute-force that can generate the image by selecting the best image by going over the 50k images and select the one which minimize the error between the small tile of the input image. This baseline script generate the best possible generated image, however, it requires to take 17 hours on my machine.
- The architecture of the code is as in any evolutionary code in the following order:
  (a) Generate the initial random population

(b) Run the fitness function on the population and get the score for each member of the population. The fitness function is simply MSE (Mean Squared Error) between the pixels of the generated image from the member of the population and the pixels of the input image as the art that it is targeted is related to the usage of the small images and concatenate them in order to get the overall image which represents the generated image.

(c) Run the selection function which select the minimum members from the score in the population to be the parents of the new population.

(d) Run the crossover function which generate the new offspring to complete the population to have the new population. Multiple crossover functions have been implemented:

  - One-point crossover to select two parents and swap some parts from each other to create one child of the offspring.
  - Uniform crossover to select the genes (index of the small image) from any of the two parents.
  - Greedy crossover to select the suitable gene from the two parent. This one get the shortest time and most effective result after tuning some parameters.

(e) Run the mutation function to add some randomness in the population genes by changing multiple genes in some of the population members.

(f) Run the termination function which tests the termination criteria which was reaching threshold from the error or not, if true, it will terminate the program and return the best generate image, if not, continue till the number of iterations is exceeded.

- The parameters that has been used in the program:

  *Note: GX means Greedy crossover, UX means Uniform crossover*

  - Dataset selection and number of the images in the set of possible images. [CIFAR dataset, 50,000 images]
  - Dimensions of the small images. [(8x8)]
  - Number of iterations. [For GX: 5, UX: 50000]
  - Population size. [For GX: 10, UX: 100]
  - Number of parents selected from the population. [0.2]
  - Number of parents that is used in crossover. [2]
  - Mutation probability. [0.1]
  - Termination Threshold. [500] (It has been selected based on evaluation the baseline generated image and observations of the experiments)

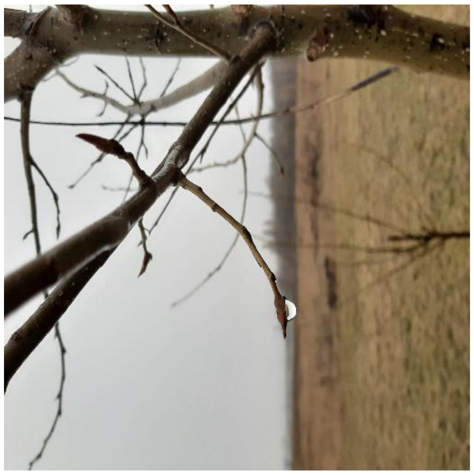
## Part3: Results

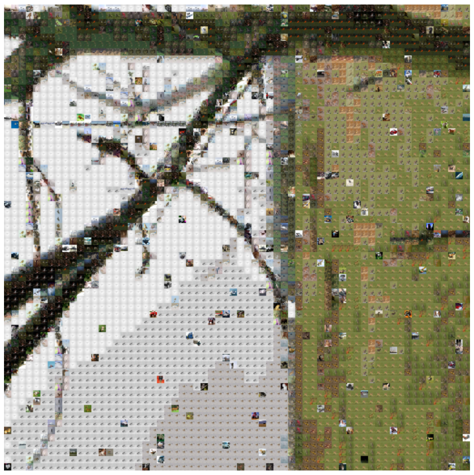
*The generated images form the greedy crossover is being generated faster in 3.5 hours, it can take more hours to get better results but after 3.5 the generated image would be acceptable, on the other hand, the other crossover functions takes more than 12 hours to compute the generated image.*

*There has been multiple tuning for the previous parameters in order to get the suitable and fastest results*

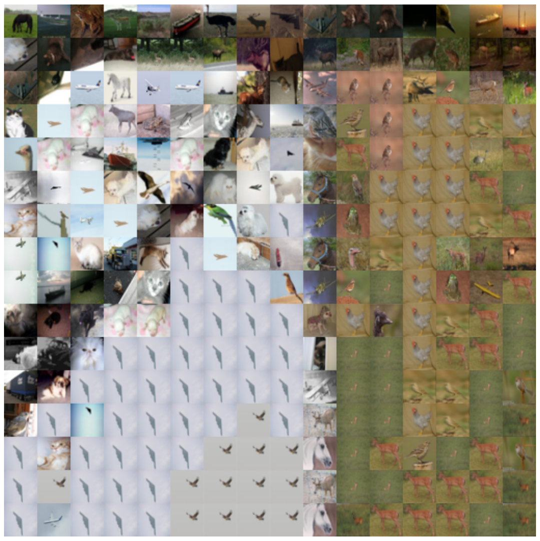
*Note: the experiments were tested on photos from Innopolis that has been captured by phone and resized to 512x512*

- Experiment 1: **Real image**

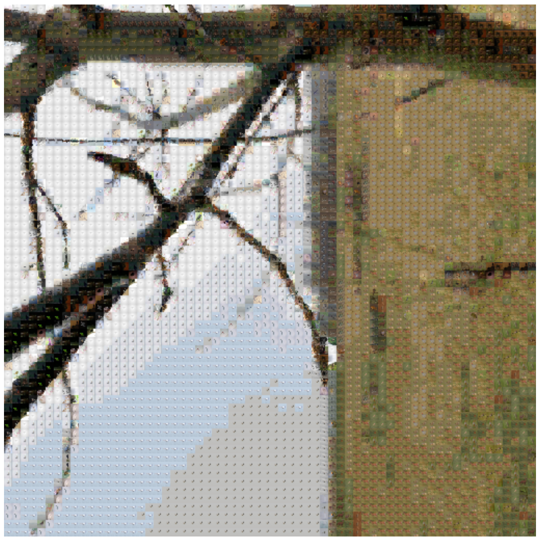




**Generated image**



**Baseline Generated image (32x32 pixels small images)**



**Baseline Generated image (8x8 pixels small images)**

**(Figures: 1) Experiment 1**

Experiment 1: could be better by running more iterations (more time to compute), decrease the mutation probability.

## Part4: Testing Images

## Summary

In the end, multiple ways and tools has been implemented to reproduce image by generating it using Evolutionary and Genetic algorithms. The artistic way that has been explained in this report has kept the colors of the original image and kept the meaning of the original image.

## References

- TutorialsPoint on Genetic Algorithms
- CIFAR dataset
- Used script to download the CIFAR dataset
- Intro to Genetic Algorithms
- Greedy Crossover