Artificial Intelligence Assignment 2

Marina Smirnova Group 6

April 2020

Contents

1 Introduction

1.1 Execution and Launching

The submitted archive contains a report, files with source code and a folder with output pictures.

The program can be easily tested by execution of the *main.py* file (using PyCharm, for example). All you need to do is change *file_name* variable that holds a name of the input image. There is also an option to get a colorful image in RGB format or in gray-scale format. **Warning:** due to greater color space (3 or 4 channels) RGB/RGBA images (especially photos) take significantly more time to be reproduced (at least an hour). So, it's recommended to test it with *save_color = False* option. All output images for testing inputs are of gray-scale format. And estimated time to produce them is within 10-18 minutes.

1.2 Mandatory Questions

1. **What is the representation in your algorithm?**
   Samples are considered to be rows of colorful squares. And samples are represented as arrays with color components (one component is a 1, 3 or 4 dimensional array, depending on Gray-Scale, RGB, RGBA image formats) for each element.

2. **Which selection mechanism is being used?**
   The Rank Selection mechanism is used to choose samples that will participate in reproduction and creation of the new population. All samples are sorted in accordance with their fitness scores, and then 90 percent are selected as parents.

3. **Which image manipulation techniques have you applied?**
   No particular image manipulation techniques were used. The main idea behind the algorithm is to work with an array representation of a squared picture where square size decreases from 256 px till 4 px exponentially.

4. **What is the fitness function?**
   As a fitness criteria I use color difference calculated using *delta_e_cie2000()* function in *colormath.color_diff* library. Colors taken for computation: one from the sample and one from the array with dominant colors (constructed on the basis of input image). Then the difference is compared with the *color_threshold*. This threshold is appropriate if lays in the interval [2, 10]. The lower its value, the better fitness is obtained.

5. **Explain the Crossover function.**
   Crossover operation takes two row samples and produce a child by choosing only acceptable squares from the samples. If square of one parent is better (in terms of color fitness), then it is preferred over another parent's square. If none of squares suits, then mutation is applied to this particular square.

6. **Explain Mutation criteria.**
   Mutation is performed using a parameter $\mu\_coefficient$ that determines the range of variety for a randomly generated color. As a base color mutation function uses one of the colors from parents chosen for crossover. Then the generated color is equal: $new\_color = base\_color + randint(-\mu, \mu)$

Examples of output pictures can be found in "Test" Section. And answers for the rest of the questions (specifically about Art) are in the "Is it Art???" Section.

More details about implementation of the algorithm, mutation, crossover, fitness and selection are in the "Implementation" Section.

2 Implementation

This section introduces an Evolutionary Algorithm that was implemented for the assignment.

2.1 algorithm_execution.py

It is the main file in terms of representing the algorithm generally, as a whole.

•**The main idea behind the algorithm:** initial picture is repeatedly divided into areas (squares) of sizes in the range [256, 4] pixels (next size in the range is smaller than the previous in 4 times). So, first iteration contains samples with 4 square areas each of 256 px size. This is done to gradually increase level of detalization of output images.

•**Stages of algorithm execution:**

1. Calculate color matrix using original image.
   Color matrix is used as a perfect example in comparison with samples.
2. Compose the fittest sample using variation variables and selection.
3. Initialize new samples for the next iteration by mutating the best sample a little.
4. Divide samples (decrease the size of squares, increase number of squares).

The best picture of an iteration (for a particular square size) is built using a function *pic_execution()* which just executes *row_execution()* for all rows to get the best picture.

And ***row_execution()*** actually process the samples and develops new generations of samples. That's why a sample is considered to be a single row of squares. It is executed until *fitness_pass* condition is satisfied. **Each iteration includes:**

1. Comparison among row samples from the current population which is of the current *population_size*. As a result an array of individual fitness scores for each sample and an array with sorted fitness scores are calculated.
2. Then, top 90 percent of the population is chosen as parents to produce the next generation. The size of the next generation (next population size) is reduced by 10 percent with every new generation. And the next generation is composed of children only (no samples from the previous generation).
3. And, finally, function *choose_best_fitness()* finds a sample that passes the *fitness_threshold* parameter and returns its index in the array of samples (or -1, otherwise).

2.2 import_export.py

It is a separate file for reading an input image and transforming an output array into an image.

The program allows you to process images in such color formats as RGB, RBGA and Gray-scale(jpg, png). The input image itself is represented as an array of color values. That's why a parameter *dimension* is important for processing (possible values are 1, 3, 4).

- **Import**.

The function takes name of an input file and argument *save_color* to produce colorful or gray-scale image.

- **Export**.

The output image is transformed from an array representation. The major peculiarity of the obtained picture is that it's drawn in the pixel style (as 16 or 32 bit art).

2.3 initialization.py

It is a file to set all needed parameters before execution of the algorithm. It is done to ease the process of changing parameters and add more flexibility.

- **List of important parameters:**

1. *dimension* - number of components in the color of a single pixel.
2. *population_size* - the initial size of the population of samples.
3. *size_array* - array with sizes of areas/squares (in pixels). It can be used to adjust the level of "pixelization".
4. *fitness_threshold* - holds value of an acceptable color difference to stop evolution and pick the best sample.
5. *delta_parameter* - holds value of an acceptable color difference between a sample and original color matrix. It is used in crossover.
6. *mu_coefficient* - parameter for mutation. The greater value of μ the wider range of color mutation of a square.
7. *k_clusters* - parameter needed to build a color matrix. It is 1 by default.

- ***init_dominant_color_matrix()***

Also, there is a function to calculate a color matrix using original image. This matrix has size corresponding to the square size of samples. It consists of colored squares filled with a dominant color for this particular square (area of the picture). The dominant color is calculated using *KMeans* method from the *sklearn* library.

2.4 fitness.py

It is a file to perform all comparison operations between samples.

- *choose_best_fitness()*

This function picks out the fittest row (sample) from the array of samples. It computes an array with scores for each row's square. Then, scores of all squares are compared with the parameter *fitness_threshold* one by one. If scores are lower or equal to the given border, index of this sample is saved and overall score is saved as *best_fitness*. So, a row (sample) with the lowest scores for individual squares and the lowest overall score is chosen.

- *row_fitness()*

This function simply returns overall fitness score of a given sample and an array with individual scores for each square in the row. The overall score is a sum of scores for each square.

- *color_difference()*

The score value is calculated using delta E (CIE 2000 Standard) metric. So, score is simply a delta E difference between colors of a square from the color matrix and a square from the sample. The range [2, 10] for delta E is considered to be acceptable, as it's not very noticeable.

2.5 variation_variables.py

It is a file to perform crossover and mutation, and produce the next generation.

- *mutate_color()*

The function gets *mu_coefficient* and color as an input. The mutation is done by adding/subtracting a random integer value (the range $[0, \mu]$) to/from the color.

- *make_crossover()*

The function takes two samples as parents and produces a child by taking best squares from both parents. If squares of both parents are not acceptable by *delta_parameter* then mutation over the closest color (of one of the parents) is done.

- *create_next_gen()*

The function returns a new population of children composed using *make_crossover()* function. The crossover is done over the best parent and all other parents. One more crossover is performed over the second and the third best parents. So that the number of new samples is equal to the number of given parents.

- *mutate_best_sample()*

This function is used to produce a population for a new iteration when the best picture for the current square size is built. This best picture serves as a base for creating the population of picture samples by mutating square colors a little.

2.6 division.py

It is a file to perform division of a picture array into areas (with decreasing sizes) to improve the speed of execution.

- *divide_pic()*

The function gets a sample (as an array) with squares of a particular size. And it increases twice the amount of rows and columns in the array. So, each square is turned into an area of 4 squares after that. Values of colors are copied.

- *divide_samples()*

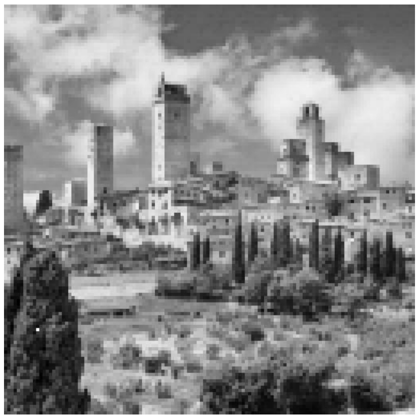
The function just executes *divide_pic()* for each sample in the given population.

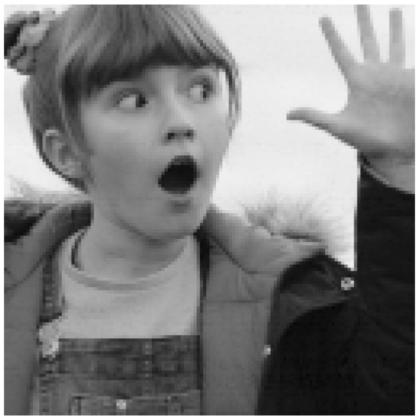
3 Tests

This section contains the actual results or output images produced by the algorithm (with square size 4 by 4 pixels).

(a) Input

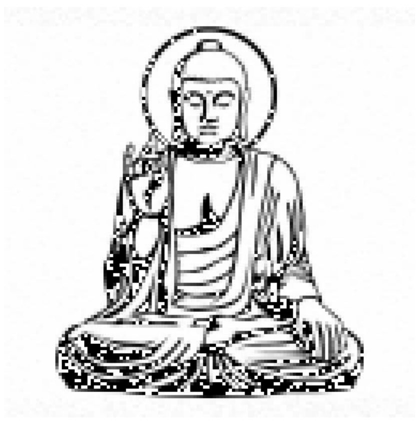
(b) Output

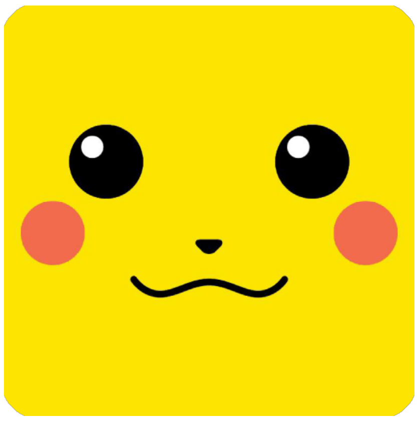


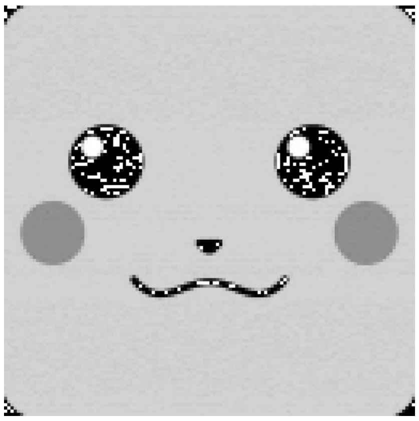


(a) Input

(b) Output







4 Is it Art???

The question is: "Can a picture produced by an Artificial Intelligence be considered a true piece of Art?"

This question is definitely philosophical one. And it causes a lot of arguments for and against.It is worth to mention that an answer a lot depends on how we define creativity and art itself. So, in this section I'd like to share my own opinion on this dilemma. Of course, I used several sources, such as articles of experts in the sphere of AI and Neuroscience, to actually come to a certain conclusion. And I claim that this is not Art.

If producing art means finding new ways to solve creative problems, then AI has already achieved that. However, Dr Romy Lorenz (cognitive neuroscientist developing brain-computer interfaces and Artificial Intelligence) points out that true artistic creativity differs from creative problem solving in that it requires a shift in perspective that machines do not appear to have the capacity for. And true Art, actually, should not be produced, other than created.

Artistic creativity is about turning an introspective thought into a medium, whether it's a sculpture or a piece of music. It's about taking an abstract form and making it concrete.

"But AI has no internal world and it has no need to create its desires or fears." Dr Lorenz claims.

I myself have attempted to define what lays behind the term Art. So, several so-called components or properties of true Art were derived.

1. **Need in input.**
   Input is any information or signals from the outside world which somehow motivates or triggers the process of creation of art. This component is common for both sides: humans and computers. At least, during the stage of learning, the basics about the world/environment are needed.

2. **Innovativeness.**
   As it was already said, AI has achieved such a level of development that it is capable of inventing new techniques of producing pictures, music or even sculptures. So, speaking about innovative methods of performance, humans and AI have abilities to make up new ones. But what is about creativity in terms of ideas which are expressed using these techniques? Exactly this aspect is unavailable for machines.

3. **Meaningfulness.**
   Art without a meaning is nothing. If there is no idea that an artist wants to show, then there is no spirit inside the creation, and it's worthless. Consequently, the question arises: "Can computers put any meaningful thought into their products?" Actually, the answer is yes. AI is already possible to make conclusions, perform logical reasoning, deduce dependencies and patterns. Nevertheless, this meaningfulness is dictated by pure logic, rationality and deduction.

4. **Emotional part.**
   By this collocation, I consider: any kinds of feelings, inner motivations and desires, inspiration born inside. And this is the major difference between a human and an AI. Computers are not able to have a sudden urge to draw, to sing, to dance - to express themselves. One component is missing. We call it a soul. This is our unique psycho-emotional part of the consciousness. And this particular thing makes us human beings. This particular thing turns artistic creations into Art.

To sum up, I think true piece of Art can not be created without feelings. Technique, performance, implementation, visual component - all these things may be perfect and eye-catching, but if the process of creating itself didn't include emotions and inner disturbances, the result is only a physical shell without a spirit. Nevertheless, I would like to note that the level of advances in the sphere of AI amazes me, and different technologies invented for artistic purposes are impressive. It is especially incredible to observe collaboration of artists and AI for creating something even more ground-breaking and beautiful. So, I would rather offer to design a brand new term to designate "Computer's Art", in order to treat AI and people who created it with proper respect and admiration.

Furthermore, my algorithm in particular is no more than a sequence of instructions needed to be executed to get the output. All output images are performed in the same fixed technique which is also programmed beforehand. So, I don't evaluate these images as art according to mentioned reasoning.

List of references:
• "Could a computer ever create better art than a human?" Article by Eleanor Lawrie.
• "Can Computers Create Art?" Article by Aaron Hertzmann
• Research papers by Romy Lorenz