Introduction to Artificial Intelligence.
Assignment Two.
Polina Turishcheva. Group 3.

Algorithm idea

1. Upload an input picture
2. Create first random generation
   - For each picture randomly select the number of points to create
   - For each point randomly select its coordinates
   - Take the RGB of the pixel in the selected position from the original picture
   - Draw a circle of random radius centered at the selected position and with the colour from the original picture
3. Update generation
   - Randomly shuffle the generation and chunk it to groups
   - From each group select best parents and create two children from them by crossover. If the group length is one, than perform mutation from the single parent.
   - Mutate the children that appeared after crossover
   - Add children to population only if its "best" parameter is better that at least one of its parents.
   - After all children are added, decrease the size of population to the initial size
4. Select the best one after n updates of generations

Definition of best

- Best picture is defined by minimum MSE between generated ant target pictures. The error is sum of squared difference between two pictures. For each pixel we sum the differences for all colour channels. In our case there were 3 channels for RGB.

  $\Sigma_{x=0}^{511}\Sigma_{y=0}^{511}(blue\_targ - blue\_cur)^2 + (red\_tar - red\_cur)^2 + (green\_tar - green\_cur)^2$,

  where *cur* is pixel at (x, y) position for current picture and *tar* is pixel at (x, y) position for target picture

Novelties

- We choose the colour from the target picture to speed up and the mimic of it. This does not lead to copying the picture because the minimum size of circle if more than 1 pixel. Moreover, because of different orders of colours we do not draw the same colour. RGB → BGR
- The crossover returns one child not two. But crossover is played twice between each group, with the length more than 1. If the group size if 2, than create 2 children from the same parents, otherwise create one children from 1-st and 2-nd best parents and 1 children from 1-st and 3-rd best parents. This is good because it helps us to create two really different children, who are not the complements of each other from parents union

- We add all children in population and after all crossovers are finished we sort the population by their *"best"* parameter (how are they close to the target picture) and decrease the size of population to the hyper-parameter constant value, leaving only the best in the population. Note that this will not lead to changing the whole population if the group size is more than 2 because each group generates only 2 children.
- The number of circles is not fixed. The sequence of circles is a gene, therefore, we have non-fixed number of genes.
- Mutation rate is not fixed as well. Moreover, during mutation we can not only change the circle but also change their number, both increase or decrease.

Algorithm hyper-parameters

- Range of numbers of circles for a picture
- Range of radii of circles
- Range of mutation rates
- Number of generations
- Size of each generation
- Group size

Results

Link to git repo with python source (click)
Here are examples of evolution for 3 different pictures:

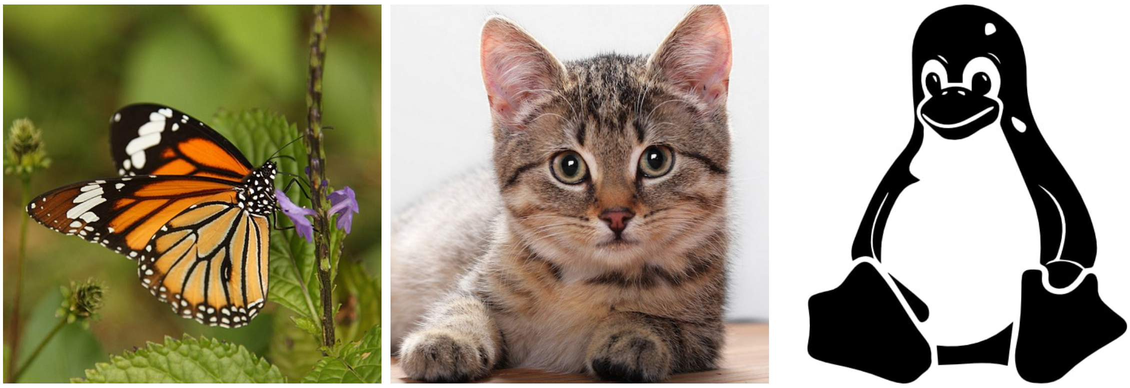

Figure 1: Target pictures

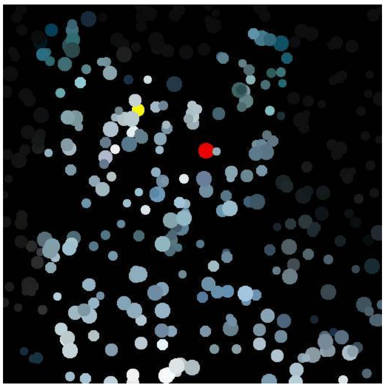

Figure 2: Parent1

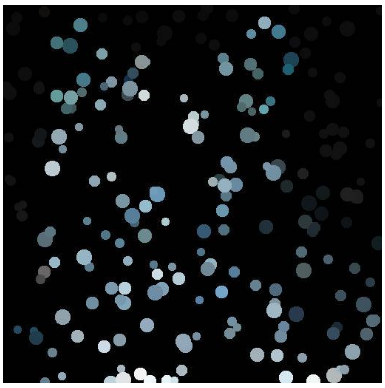

Figure 3: Parent2

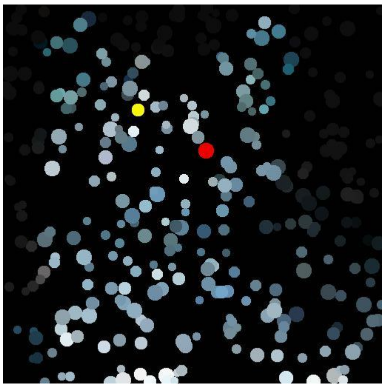

Figure 4: Child

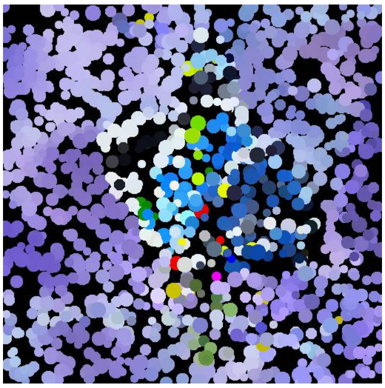

Figure 5: Before Mutation

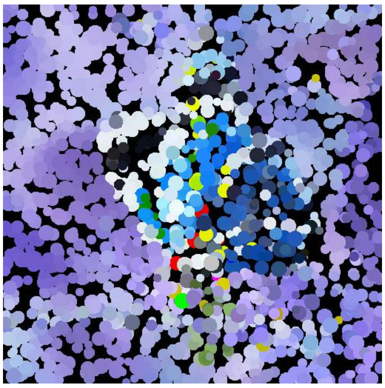

Figure 6: After Mutation

Example of adding bubbles after mutation

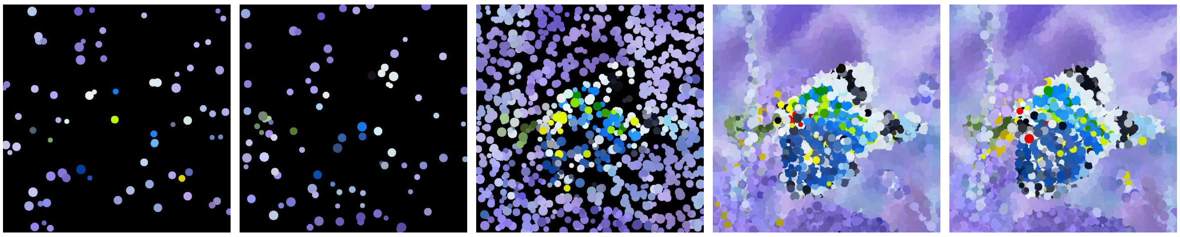

Figure 7: Evolution for butterfly. From 6th- to 10-th generations

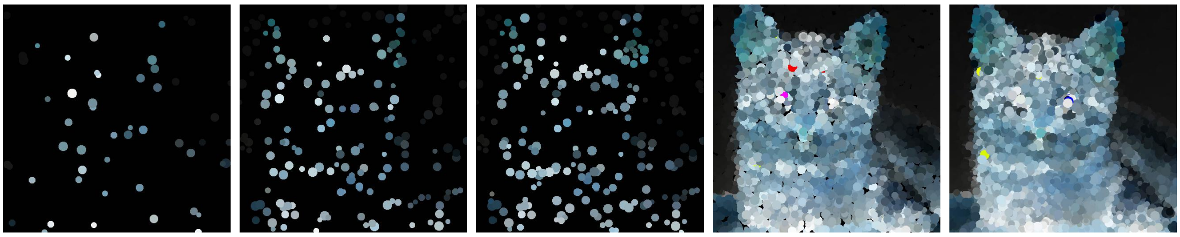

Figure 8: Evolution for cat. From 6th- to 10-th generations

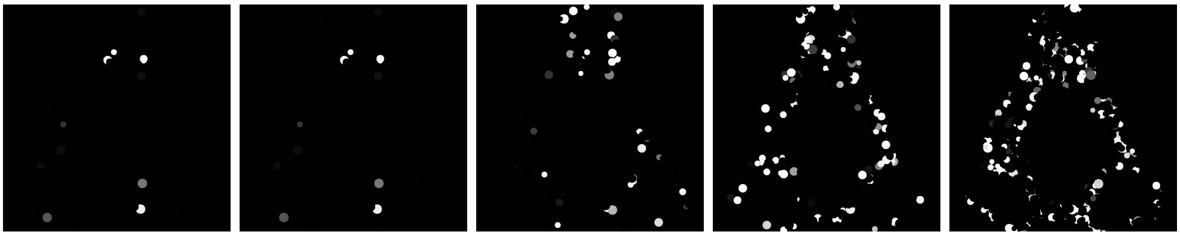

Figure 9: Evolution for the penguin. Generations 1-5, for computer drawn pictures, not photos it learns faster.The colours are inversed.

Drawbacks

The drawback of the algorithm is that is we want to create close to original shape, we have to use rather small size of circles. It was empirically discovered that after 10 generations the shape of the original picture is usually reached (not always but in most of the cases). However, this implies that we already have the big number of circles and in the majority of population and this slows evolution crucially (from $\approx$ 500 it starts to calculate long). The sample of logs with timestamps is provided further. The solution for this maybe to use *cython* library to speedup the calculations.
The other weak point is that due to much randomness it is extremely difficult to calculate the complexity of the algorithm analytically.

```
in increase_population  23:35:54.591371
in crossover  23:35:54.591486
in crossover  23:35:54.606743
in crossover  23:35:54.621138
in crossover  23:35:54.640373
in crossover  23:35:54.662646
in crossover  23:35:54.685163
in decrease_population  23:35:54.700023
this is generation 0
in increase_population  23:35:54.705841
in crossover  23:35:54.705885
in crossover  23:35:54.722691
in crossover  23:35:54.732698
in crossover  23:35:54.750920
in crossover  23:35:54.763151
in crossover  23:35:54.779571
in decrease_population  23:35:54.796950
this is generation 1
in increase_population  23:35:54.804236
in crossover  23:35:54.804301
in crossover  23:35:54.810019
in crossover  23:35:54.816843
in crossover  23:35:54.837342
in crossover  23:35:54.844903
in crossover  23:35:54.853685
in decrease_population  23:35:54.871975
this is generation 2
```

```
in increase_population  23:35:54.879918
in crossover  23:35:54.879967
in crossover  23:35:54.905875
in crossover  23:35:54.913262
in crossover  23:35:54.922817
in crossover  23:35:54.942642
in crossover  23:35:54.972166
in decrease_population  23:35:55.005627
this is generation 3
in increase_population  23:35:55.019793
in crossover  23:35:55.019946
in crossover  23:35:55.069615
in crossover  23:35:55.087388
in crossover  23:35:55.145251
in crossover  23:35:55.215262
in crossover  23:35:55.302251
in decrease_population  23:35:55.354285
this is generation 4
in increase_population  23:35:55.363103
in crossover  23:35:55.363171
in crossover  23:35:55.416921
in crossover  23:35:55.505352
in crossover  23:35:55.617166
in crossover  23:35:55.730923
in crossover  23:35:55.879519
in decrease_population  23:35:56.041312
this is generation 5
in increase_population  23:35:56.048841
in crossover  23:35:56.048896
in crossover  23:35:56.330021
in crossover  23:35:56.618671
in crossover  23:35:57.014967
in crossover  23:35:57.570363
in crossover  23:35:58.257388
in decrease_population  23:35:59.218006
this is generation 6
in increase_population  23:35:59.225091
in crossover  23:35:59.225136
in crossover  23:36:00.699694
in crossover  23:36:02.490311
in crossover  23:36:04.587317
in crossover  23:36:07.917831
in crossover  23:36:10.623846
in decrease_population  23:36:15.743073
this is generation 7
in increase_population  23:36:15.750316
in crossover  23:36:15.750363
in crossover  23:36:19.848430
in crossover  23:36:19.858075
in crossover  23:36:26.135946
in crossover  23:36:33.653738
in crossover  23:36:40.787790
in decrease_population  23:36:51.415734
this is generation 8
in increase_population  23:36:54.879158
in crossover  23:36:54.879212
in crossover  23:37:09.069387
in crossover  23:37:28.482683
in crossover  23:37:53.698563
in crossover  23:38:25.462723
in crossover  23:39:06.671587
in decrease_population  23:39:53.479184
this is generation 9
in increase_population  23:39:56.970714
in crossover  23:39:56.970791
in crossover  23:40:44.216072
in crossover  23:42:06.790754
in crossover  23:44:03.800053
in crossover  23:55:50.874402
in crossover  00:01:16.763762
in decrease_population  00:04:50.229001
this is generation 10
in increase_population  00:06:00.533750
in crossover  00:06:00.533803
in crossover  00:10:43.668231
in crossover  00:32:43.135719
in crossover  00:38:42.524655
in crossover  01:16:07.982991
in crossover  01:43:50.214400
in decrease_population  02:38:59.026929
this is generation 11
in increase_population  02:44:14.864957
in crossover  02:44:14.865023
in crossover  04:28:45.410687
Keyboard interrupt
```