Artificial Intelligence

Assignment 2

Submitted By-

Name - Shivam Singh

Class - AI\_CS-18

Roll no. - 21051732

GITHUB LINK :

https://github.com/shivam777singh/21051732\_AI

**Task 1:** You are required to implement a program that takes an image as its input and generates the same image

using N number of squares.

You need to implement this using Genetic Algorithm.

# Introduction:-

A genetic algorithm is a search heuristic that is inspired by Charles Darwin’s theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

# Scope of work:-

Five phases are considered in a genetic algorithm.

1. Population

Population is a subset of solutions in the current generation. It can also be defined as a set of chromosomes. There are several things to be kept in mind when dealing with GA population −

* The diversity of the population should be maintained otherwise it might lead to premature convergence.
* The population size should not be kept very large as it can cause a GA to slow down, while a smaller population might not be enough for a good mating pool. Therefore, an optimal population size needs to be decided by trial and error.

1. Fitness function

The fitness function simply defined is a function which takes a candidate solution to the problem as input and produces as output how “fit” our how “good” the solution is with respect to the problem in consideration.

Calculation of fitness value is done repeatedly in a GA and therefore it should be sufficiently fast. A slow computation of the fitness value can adversely affect a GA and make it exceptionally slow.

In most cases the fitness function and the objective function are the same as the objective is to either maximize or minimize the given objective function. However, for more complex problems with multiple objectives and constraints,

an Algorithm Designer might choose to have a different fitness function.

1. Selection

Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.

Parent selection is very crucial to the convergence rate of the GA as good parents drive individuals to a better and fitter solutions.

However, care should be taken to prevent one extremely fit solution from taking over the entire population in a few generations, as this leads to the solutions being close to one another in the solution space thereby leading to a loss of diversity. Maintaining good diversity in the population is extremely crucial for the success of a GA. This taking up of the entire population by one extremely fit solution is known as premature convergence and is an undesirable condition in a GA.

1. Crossover

The crossover operator is analogous to reproduction and biological crossover. In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents. Crossover is usually applied in a GA with a high probability –

1. Mutation

In simple terms, mutation may be defined as a small random tweak in the chromosome, to get a new solution. It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – pm. If the probability is very high, the GA gets reduced to a random search.

Mutation is the part of the GA which is related to the “exploration” of the search space. It has been observed that mutation is essential to the convergence of the GA while crossover is not.

# Population:-

Here the population is basically the different shapes present in the canvas to generate the image. The population signifies all the possible combinations that the generated image may have. Among the possible combinations we need to select the best option to move ahead from the randomly generated solutions. This is done to get the optimal solution no matter how much time it takes.

# Fitness function:-

This fitness function used here are to select the best shapes which can be used to generate the next generation of the image.

I used the similarity to determine the best shapes which are added to the generated image and this determining its “fitness”. It is applied in the evaluatePool() in the class GeneticAlgorithm which is overridden in the class GeneticShape.

I converted the target image and the generated images to pixel array used the similarity between the two pixel array to measure the similarity between them to check how fit the generated image is.

# Selection function:-

I used the selectBestFromGen() to select the best fitted shapes from the generated shapes. This is done using a threshold found using

let threshold = Math.ceil(this.parent.POP\_SIZE \* 0.25);

This allows us to make a new array for the genBest from the pool and utilize the same to use in the generated image by selecting the best (threshold) number of the possible images from the present pool.

# Crossover:-

The genBest which is created in the selection function, is now used to generate a crossover i.e. at random we mate any two of the items from the genBest, where any of the attributes are selected at random to make the offspring.

createFromBest() {

this.pool = new Array(this.parent.POP\_SIZE); let i = 0;

for (i = 0; i < this.genBest.length; i++) { this.pool[i] = Model.clone(this.genBest[i]);

}

while (i < this.parent.POP\_SIZE) {

this.pool[i] = (this.mate(random(this.genBest), random(this.genBest)));

i++;

}

}

As analogy to the biological reproduction the offspring gets random characteristics from both of the parents.It is applied using the random function, taking min and max value and taking a random value between them to get the length of the child. And then the child of that length is returned. This child is then added to the main pool.

# Mutation:-

There can be multiple ways to mutate the items in the pool. I utilized multiple mutation methods to mutate the pool. The shape can be mutated in multiple ways.

The methods used here are - mutate\_deleteItem(data, itm), mutate\_addItem(data, itm), mutate\_changeItem(data, itm), mutate\_switchItems(data, itm), mutate\_generateRandomImage(data, itm), mutate\_moveItemPointX(data, itm), mutate\_moveItemPointY(data, itm), mutate\_changeItemColor(data, itm) and mutate\_changeItemColorAlpha(data, itm).

All the functions are self explanatory. Among these mutate functions, a function is randomly chosen to mutate in the mutate(). This is

done using the MUTATE\_RATE of the parent(I.e. the item). It is used to maintain and introduce diversity in the genetic population. It is done as it has been observed that mutation is essential to the convergence of the GA.

# Implementation and the output:-

So the code’s output provides the number of generation , percentage of similarity to the target image and the time elapsed from the start of the program. It start with producing random shapes in the canvas. As the draw function is a infinite loop it will go on until the code is terminated. During this time the algorithm evaluates and tries to reach the target image or something close to it.

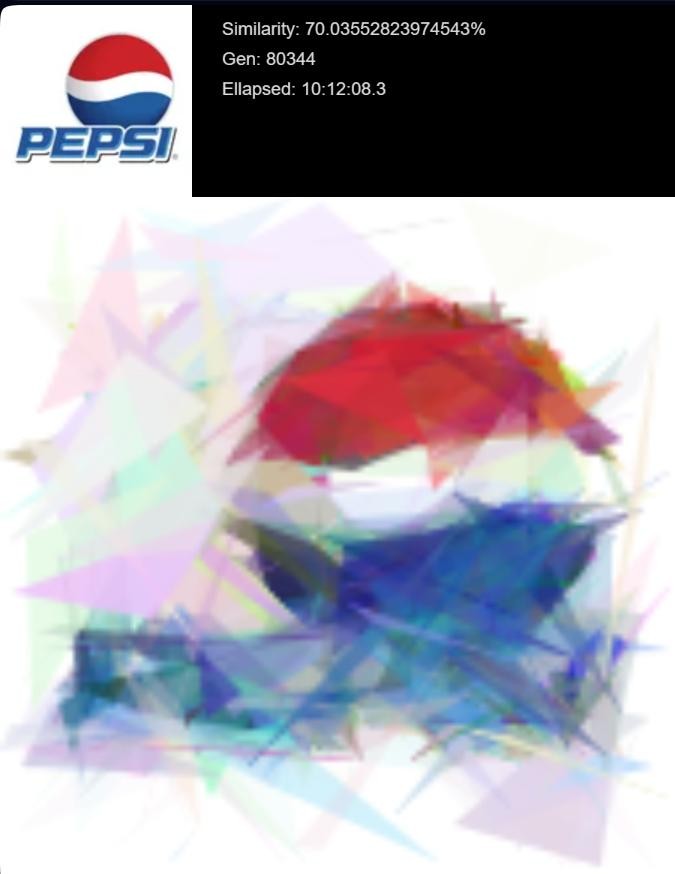
As the similarity reaches 10%, we get a random mishmash of shapes of random colors and sizes.



As the similarity reaches 60%, we get a semblance to the target image.



As the similarity reaches 70%, it starts showing as blurred image of the target. This is the closest we can get in a limited time as the program starts to take more and more time to improve the similarity as the algorithm reaches the convergence slowly.



# Conclusion:-

In conclusion, the implementation of a genetic algorithm for image generation has demonstrated both the potential and challenges in harnessing computational techniques to produce visual content. This shows how a concept of “the survival of the fittest” from biology, a subject way different from computer science manage to improve and provide a new angle to the making of the algorithm.

This implementation has underscored the significance of optimizing various parameters within the genetic algorithm framework. The careful adjustment of population size, mutation rates, crossover mechanisms, and fitness evaluation were pivotal in steering the algorithm towards producing images that exhibit desired target characteristics.

I managed to utilize this technique to make a simple image generation algorithm, which generates the images that resembles the target and and provides a complete solution.