Reproducing Images using Genetic Algorithm

Arif Chaudhary[1], Harsh Singh[2], Hrutuja Saswade[3], Dheeraj Jain[4], Shiwani Gupta[5]

[1,2,3,4] Student, Dept. of Computer Engineering, Thakur College of Engineering & Technology, Maharashtra, India
[5]Asst. Professor, Dept. of Computer Engineering, Thakur College of Engineering & Technology, Maharashtra, India

---***---

Abstract - *This paper presents a Genetic Algorithm for reproducing images. The main motto of this image enhancement is to process the images so that the result is more suitable than the original image of a specific application. Image enhancement is used to improve the visual effect and quality of an image. Input images can have one or more channels (i.e., the image could be binary, gray, or color, such as RGB). The Genetic Algorithm (GA) starts from a casual generated image of the exact shape as the image input. This casually generated image is developed, using crossover and alternation, using GA until it produces an image which is similar to the original image. The same original images might not be accurately produced, but at least a same image will be generated. Image enhancement is one of the try and tough techniques in digital image processing. We willl be performing this process on four different types of images low quality RGB & B/W, high quality RGB & B/W & At last comparing the accuracy of different images by Structural Similarity Index Measure (SSIM). This paper presents an overview of Genetic Algorithms for reproducing images.*

***Key Words*: GARI, SSIM, Chromosome, Performance Comparision, 1D Row Vector, Value Encoding, PyGAD.**

1. INTRODUCTION

It is the implementation of Genetic Algorithm functions which is responsible for reproducing the images. It first reads a coloured image file named "cherries.jpg", there in the project, and then passes them through the function of GA. We display that image and how it is reproduced by Genetic Algorithm for Reproducing Images (GARI) after a 50,000-generation gap. Genetic algorithms find the output space of a function via simulated evolution, i.e., the survival of the fittest strategy. Genetic algorithms are shown to decode linear and nonlinear problems too by searching all regions of the state space and highly exploiting the promising areas through different combinations, crossover, and selection operations applied to the individual in the population, which is the individual's solution (analogous to chromosomes) of the state area. These operators, which are dependent on probability rules, are put up to the population, and successive generations are reproduced. Basically, the start of the search for an optimal solution begins with any randomly generated population of chromosomes. Each set of generation will have a newer set of chromosomes founded from the application of the operators. A fitness, or objective function, is to be defined related to the problems. The parent selection process makes sure that the fittest and the bestest member of the population have the highest probability ratio of becoming parents, in the thought that their offspring will come together to desirable features, and have higher fitness, to both. The algorithms terminate either when the set of generation numbers is reached, or the fitness has reached a "good" level. The main use of a Genetic Algorithm to solve these six major issues: (i) chromosome representation, (ii) selection function, (iii) creation of the initial population, (iv) genetic operators making up the reproduction function, (v) fitness function, and (vi) termination criteria.

2. LITERATURE SURVEY

The related work to reproducing images using Genetic Algorithm is discussed here, K.D Gupta & Sajib Sen proposed the method for multi-objective Genetic Algorithm. Their method will first resize the image using normal image resizing option provided by operating system or standard library and attach the extra 2 array of data which contains number of 1 in main image in each rows and columns. Also, the total number of 1 in the image will be there too. Now they will take the help of genetic algorithm which will use this informations to reproduce the image in original size. From their result analysis, it seems that if they make a smaller block, they can produce image faster and more accurate to original. Also, its seldom that they hit the local optima and stuck with not good enough version. Changing mutation rate that time gives them better result. It seems they need to apply dynamic penalty method and mutation rate to tackle this issue[1]. Shivangini & Arvind proposed their algorithm reduce noise reduction of different contrast value images is 98%, with mean=2.60, this can be improved. It is implemented with noise (different contrast value). De-noising was carried out with the mean of each noise reduction[2]. Aravind & Co proposed the image segmentation system developed by them under the constraint condition, the foremost being that it performs operation only for colorless images, there exists room for development in extending the system to applications in environments conforming to colored images. Recently, researchers have investigated the application of Genetic Algorithms into the color image segmentation problem[3]. M. Yu & Co proposed knowledge of the object shape, it is difficult to determine the optimum size of the structuring element. One possible way to select the suitable structuring element size is to use the edge characteristic-scale analysis. More extensive studies for automatic selection of the parameters can be carried out[4]. Sara Hashemi proposed method uses a simple chromosome structure, Fitness function: In the proposed method, the number of edges and their overall intensity are used as fitness value for each chromosome because a gray image with good visual contrast includes many intensive edges, Selection algorithm: Selection of the individuals is done based on the fitness value of the solutions, Crossover and different operators: Because of developing individual chromosomes based on an easy structure[5].

3. PROPOSED ARCHITECTURE

Following steps to follow in order to reproducing an image, are as follows:

- Read an image
- Prepare the fitness function
- Create an instance of the pygad.GA class with the appropriate parameters
- Run PyGAD
- Plot results
- Calculate some statistics

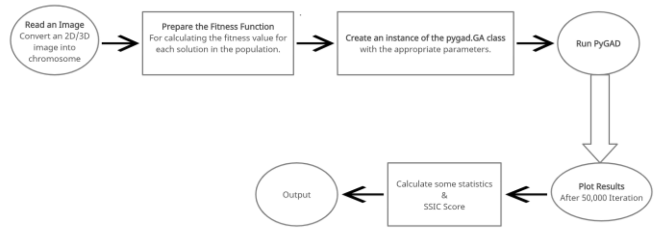
Read an Image
Convert an 2D/3D image into chromosome
Prepare the Fitness Function
For calculating the fitness value for each solution in the population.
Create an instance of the pygad.GA class with the appropriate parameters.
Run PyGAD
Plot Results
After 50,000 Iteration
Calculate some statistics & SSIC Score
Output

Fig -1: Block Diagram

It works with both color and gray images without any modifications. Just give the image path. Using three parameters, we can design our own to satisfy our needs. The parameters are: 1) Population size. i.e., number of individuals per population. 2) Mating pool size. i.e., Number of the chosen parents in the mating pool. 3) Mutation percentage. i.e., number of genes to change their values. Value encoding used for representing the input. Crossover is applied by giving half of genes from the two parents. The changes in the image are applied by randomly changing the values of randomly selected predefined percent of genes from the parent's chromosome. First step in GA is to represent/encode the input as a sequence of characters. The encoding used is value encoding by giving each gene in the chromosome its actual value in the image. Image is converted into a chromosome by reshaping it as a single row vector. Creating an initial population randomly. First step in GA is to represent the input in a sequence of characters. The encoding used is value encoding by giving each gene in the chromosome its actual value.

Calculating the fitness of a single solution. The fitness is basically calculated using the sum of absolute difference between genes values in the original and reproduced chromosomes. Then population fitness calculates the fitness of all solutions in the population.

selects the best individuals in the current generation, according to the number of parents specified, for mating and generating a new better population. Then applying crossover operation to the set of currently selected parents to create a new generation. Applying mutation by selecting a predefined percent of genes randomly. Values of the randomly selected genes are changed randomly.

Saving the best solution in a given generation as an image in the specified directory. Image is saved according to the stop point to make sure the stop saves images from all generations as saving many images will make the algorithm work slow. Show all individuals as images in a single graph.

4. PROPOSED MODEL

4.1 Implementation

Genetic algorithms are a random-based enhancement technique that has a number of generic processes that are normally followed to solve any enhanced problem. These processes are then made by choice to the problem being solved[7]. This paper discusses these processes in deep but focuses on how to modify them according to this model. The summary of these steps is as follows:

4.1.1 Data Representation (Read an Image)

The first task for an image problem using Genetic algorithm is to ponder about the finest way to showcase the data. Genetic algorithm trusts the chromosome (i.e., solution) as a 1D row vector. The input image will not be 1D. The image might be in 2D if it's a binary or a gray image.

There might be 2 or more than 2 dimensions if the input image is colour. If it's RGB, for eg, then there are three dimensions, one for every channel. Any data of greater than 1 dimension must be represented in a 1D vector. Is it possible that we convert the MD data to a 1D vector ?

Going with the easy case in which the input is a 2D image (i.e., 2D matrix), changing it into a 1D vector needs us to mix the 2 dimensions into a one dimension. The matrix has n number of rows, and its compulsory to mix all of these rows into one's row[8]. This can be done by stacking the multiple rows together. This is illustrated in the figure 3 & 4. The figure shows how an image/matrix of 3 rows and 3 columns. That's a total of 3x3=9 elements.

Fig -2: Input Image

When converting to a 1D vector, the vector length will obviously be 9. The first three figures of this vector will be chosen from the first row in the image. The next three elements of the vector will be chosen from the second row, and lastly the last three elements in the vector will be taken from the last row.

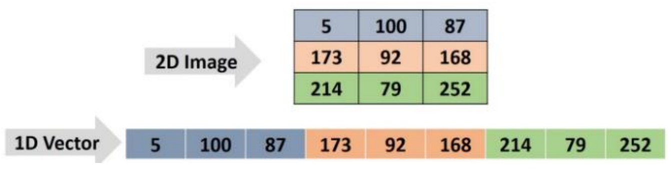

Fig -3: 2D Image into 1D Vector[6]

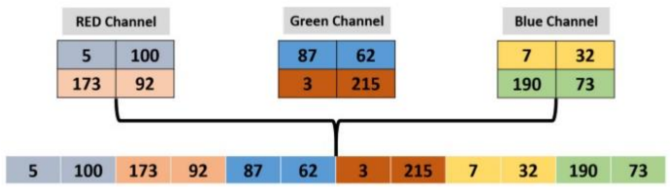

Fig -4: 3D Image into 1D Vector[6]

While converting the MD data to a 1D vector is the last step for representing the data of this model, but it may not be the last tread for other similar problems. In this model, value encoding is used. The similar values in the image are used in the chromosome. In the other problems, it may prefer to encode the values in different ways. In this particular case, there is an encoding step between the original form of the data and the chromosome.

4.1.2 Initializing Population

GA starts an initial population, which may be a group of solutions (chromosomes) to the given problem. These solutions are randomly generated. A function named initial_population() is created to return such a population.[9]

Without even looking at the function arguments, let's think about what arguments are expected to be passed to it. In beginning, a chromosome (1D vector) is to be created. The vector length is almost same to the number of objects in the image. Thus, there must be an argument to assist in calculating this length. This is why the function accepts the form of the image because the first argument named img_shape.

4.1.3 Fitness Calculation

The next thing is to create a function which will be used as a fitness function for calculating the fitness value for every solution within the population. This function must be a maximization function that accepts 2 parameters representing an answer and its index. It returns a value representing the fitness value.

The fitness value is calculated using the sum of absolute difference between genes values within the original and reproduced chromosomes. The gari.img2chromosome() function is named before the fitness function to represent the image as a vector because the Genetic Algorithm can work with 1D chromosomes.

After calculating the fitness values for all the possible solutions, the very next step is to select the perfect one of them for making the next population. The best of these solutions is called parents. By mating these parents, the think for a return is good solutions (offspring).

4.1.4 Parent Selection

The parents selected from a given population are the simplest solutions within it. once we say "best solutions", we're pertaining to the solutions with the very best fitness values.

Assume that the population has 6 solutions and their fitness values are as given within the figure below. Before selecting the simplest parents, we'd like to make a decision what percentage parents to pick . Assuming that half (3 out of 6) of the solutions are going to be selected, then the simplest 3 solutions are those with the very best fitness values. This is why we need the solutions with ID 4, 2, and 6 are selected[10].

4.1.5 Crossover

Mating two organisms has a meaning of creating a fresh offspring that shares the genes inside both of them. The crossover operation selects a number of genes from each parent and places them into their offspring.

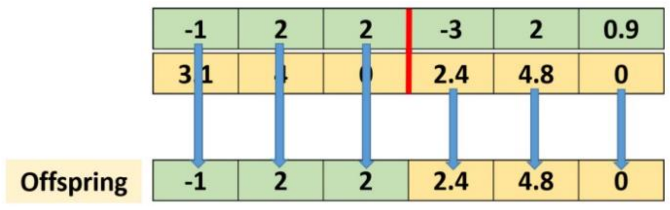

Fig -5: Crossover[6]

The crossover operation is applied within the model employing a function named crossover(). It accepts 3 arguments: the oldsters selected previously using the select_mating_pool() function, the input image shape (img_shape), and therefore the number of offspring to return (n_individuals), which defaults to eight[11].

The function goes through the ancestors, selecting 2 of them for mating and producing an offspring. Then it moves to a different 2 parents and repeats the method.

4.1.6 Mutation

The mutation operation selects some genes within the chromosome then randomly changes their values.

It's implemented consistent with the mutation() function listed below. It accepts the population returned by the crossover() function, number of oldsters , and therefore the percent of the genes to be changed. the no. of ancestors is passed so as to easily apply the mutation over the offspring and skip the oldsters[12].

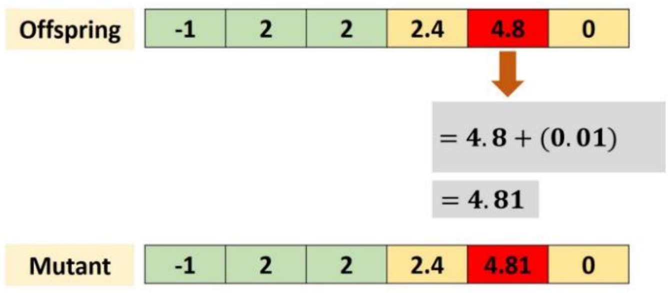

Fig -6: Mutation[6]

4.1.7 Creating an Instance of PyGAD.GA Class

It is vital to use random mutation and set the mutation_by_replacement to True. supported the range of pixel values, the values assigned to the init_range_low, init_range_high, random_mutation_min_val, and random_mutation_max_val parameters should be changed[13].

If the image pixel values range from 0 to 255, then set init_range_low and random_mutation_min_val to 0 as they're but change init_range_high and random_mutation_max_val to 255. Then, Simply, call the run() method to run PyGAD.

5. RESULT & DISCUSSION

5.1 Performed on Low Quality RGB Image

First, we create and shows a plot that summarizes how the fitness value evolved by generation on low quality RGB Image. It is called after completing at least 1 generation.

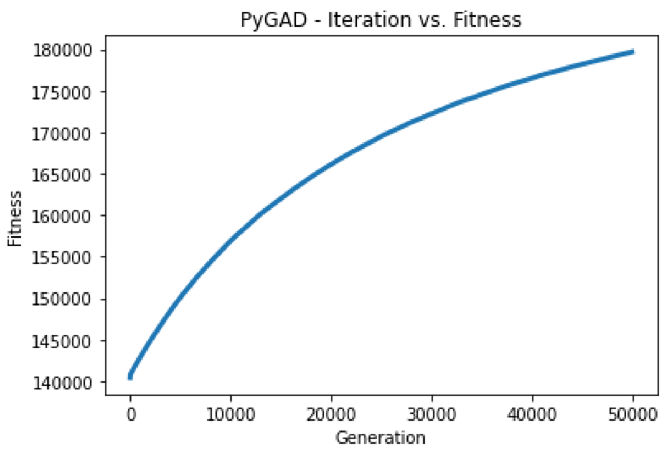

Chart -1: Iteration vs Fitness(Low Quality RGB Image)

As we can see that the fitness value of the best solution, we got is 179655.4851110709 & best fitness value reached after 49997 generations out of 50000 generations.

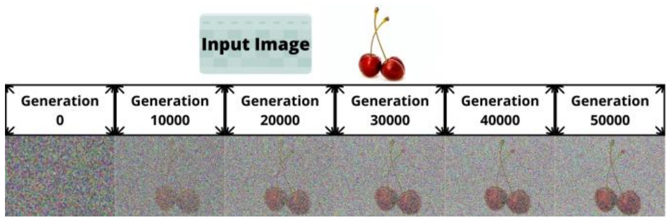

Fig -7: Generations Comparison(Low Quality RGB Image)

First, we save the solutions (images) on the memory in order to track the progress of the algorithm. The progress of the algorithm when applied to a Low Quality RGB image is given in the above figure.

Table -1: SSIM Performance over Generations(Low RGB)

SSIM Performance Metric	
SSIM after 10K Generations	0.015210803663310029
SSIM after 20K Generations	0.019835823501690747
SSIM after 30K Generations	0.023477062997603257
SSIM after 40K Generations	0.02347706299760325
SSIM after 50K Generations	0.02730847980812102

As we can see that the Structural Similarity Index Measure (SSIM) is increasing over the generations for low quality RGB Image in the above table.

5.2 Performed on High Quality RGB Image

Second, we create and shows a plot that summarizes how the fitness value evolved by generation on high quality RGB Image. It is called after completing at least 1 generation.

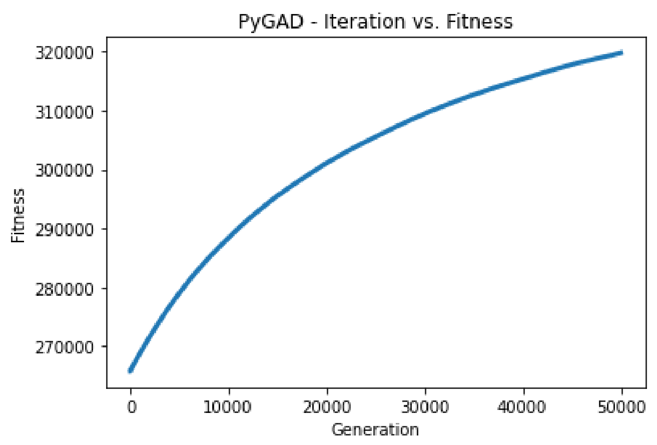

Chart -2: Iteration vs Fitness(High Quality RGB Image)

As we can see that the fitness value of the best solution, we got is 319791.33394319925 & best fitness value reached after 49999 generations out of 50000 generations.

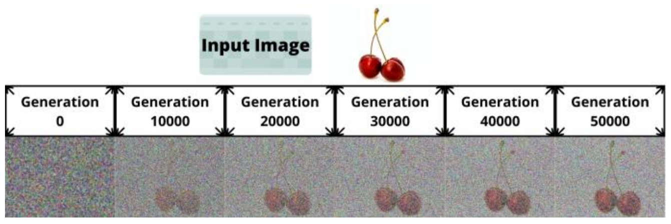

Fig -8: Generations Comparison(High Quality RGB Image)

Second, we save the solutions (images) on the memory in order to track the progress of the algorithm. The progress of the algorithm when applied to a High Quality RGB image is given in the above figure.

Table -2: SSIM Performance over Generations(High RGB)

SSIM Performance Metric	
SSIM after 10K Generations	0.012446354098952955
SSIM after 20K Generations	0.014665676774561496
SSIM after 30K Generations	0.016536368946887887
SSIM after 40K Generations	0.017748078384999255
SSIM after 50K Generations	0.018785332391945426

As we can see that the Structural Similarity Index Measure (SSIM) is increasing over the generations for high quality RGB Image in the above table.

5.3 Performed on Low Quality B/W Image

Third, we create and shows a plot that summarizes how the fitness value evolved by generation on low quality B/W Image. It is called after completing at least 1 generation.

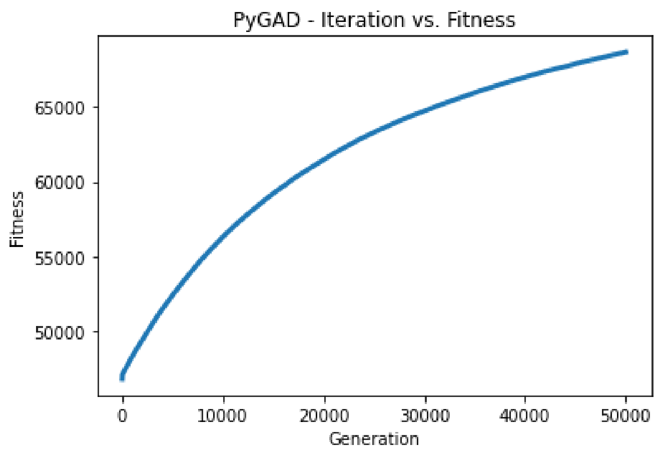

Chart -3: Iteration vs Fitness(Low Quality B/W Image)

As we can see that the fitness value of the best solution, we got is 68639.63029713475 & best fitness value reached after 49999 generations out of 50000 generations.

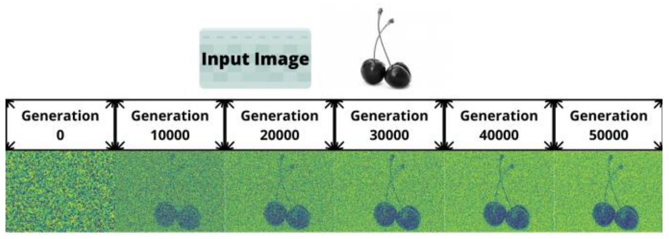

Fig -9: Generations Comparison(Low Quality B/W Image)

Third, we save the solutions (images) on the memory in order to track the progress of the algorithm. The progress of the algorithm when applied to a Low-Quality B/W Image is given in the above figure.

Table -3: SSIM Performance over Generations(Low B/W)

SSIM Performance Metric	
SSIM after 10K Generations	0.018704007571078884
SSIM after 20K Generations	0.021406325260735045
SSIM after 30K Generations	0.023941404878391997

SSIM after 40K Generations	0.02580501003686182
SSIM after 50K Generations	0.027722954118421596

As we can see that the Structural Similarity Index Measure (SSIM) is increasing over the generations for low quality B/W Image in the above table.

5.4 Performed on High Quality B/W Image

Fourth, we create and shows a plot that summarizes how the fitness value evolved by generation on high quality B/W Image. It is called after completing at least 1 generation.

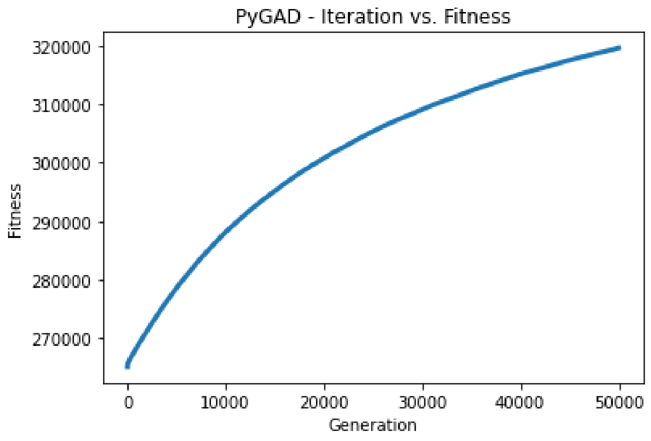

Chart -4: Iteration vs Fitness(High Quality B/W Image)

As we can see that the fitness value of the best solution, we got is 319595.73380138964 & best fitness value reached after 50000 generations out of 50000 generations.

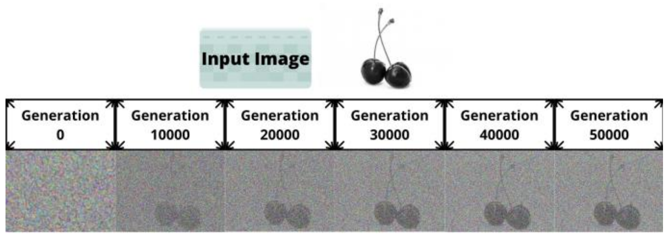

Fig -10: Generation Comparison(High Quality B/W Image)

Fourth, we save the solutions (images) on the memory in order to track the progress of the algorithm. The progress of the algorithm when applied to a High-Quality B/W image is given in the above figure.

Table -4: SSIM Performance over Generations(High B/W)

SSIM Performance Metric	
SSIM after 10K Generations	0.012357200191323874
SSIM after 20K Generations	0.014826052247667526
SSIM after 30K Generations	0.016706441573328106
SSIM after 40K Generations	0.017847103834102082
SSIM after 50K Generations	0.020847889452754736

As we can see that the Structural Similarity Index Measure (SSIM) is increasing over the generations for high quality B/W Image in the above table.

5.5 Comparison of SSIM Index

Here is the line graph showing comparision of Structural Similarity Index Measure (SSIM) on all four types of images we have iterated over 50,000 generations.

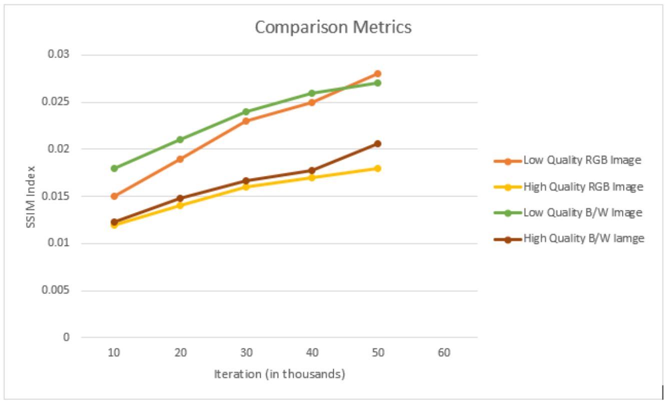

Chart -5: Comparison of SSIM Index

As we can see that, low quality (RGB & B/W) images are giving better SSIM Index value as compare to high quality (RGB & B/W) images. So, we can say that our model performs best on low quality images & RGB Images.

6. FUTURE SCOPE

Future work must be mainly concentrated on the improvement and efficiency of the model. Like, Other methods to generate initial population. Instead of randomly generating the initial population, a certain probability may be used depending on the neighborhood of the corresponding pixels in the noisy image. An alternative choice is a threshold image. Other fitness function to measure the fitness of the chromosomes. More precise measurement may be used instead of just using the summation of pixel wise error. Different selection criteria to select individuals for the mating pool. After the fitter individuals are selected, a small number of randomly selected individuals should be selected to bring more diversity into the mating pool.

At last, Genetic Algorithm is still a developing method in digital image processing background. The results obtained are very promising and is therefore worth studying further. Many improvements can be introduced into the basic proposed algorithm for enhancing the computation time of the model and accuracy of the results.

7. CONCLUSION

The final goal of this to develop and study the feasibility of an Image Reproducing technique that is based on the Genetic Algorithm. In this approach, image reproduction is considered as an optimization problem and it is carried out using Genetic Algorithms. Genetic algorithms are based on the evolution theory also known as "survival of the fittest" principle. In Genetic Algorithm, an initial population of candidates is required to start the evolution. All these individuals are generated based on the problem to be solved. In the process of reproduction, the parent selection is used before crossover and mutation. After this operation, the fitness of all the individuals is evaluated and only the fitter candidates are allowed to produce offsprings. The offspring so created tend to inherit the "best feature" of their parents. Generation after generation, the overall fitness of the entire population is improved and finally each individual in the population is as good as others in the sense of being the fittest individual. And we have done this process on four different type of images to compare the performance of our model. An exhaustive performance study of the algorithm on four different type of images was performed. The parameters that affect the performance of the algorithm were studied and discussed.

REFERENCES

[1] K.D Gupta & Sajib Sen,(2018) "A genetic algorithm approach to regenerate image from a reduce scaled image using bit data count," ResearchGate.

[2] S. Shrivastava & A. Upadhyay, (2014) "Image enhancement using genetic algorithm," IJERT.

[3] I. Aravind, C. Chandra, M. Guruprasad, P.S. Dev, R.D.S. Samuel, (2003) "Implementation of image segmentation and reconstruction using genetic algorithms," IEEE.

[4] M. Yu, N. Eua-anant, A. Saudagar, L. Udpa, (2002) "Genetic algorithm approach to image segmentation using morphological operations," IEEE.

[5] Sara Hashemi, Soehila Kiani, Navid Noroozi, Mohsen Ebrahimi Moghaddam, (2009) "An image enhancement method based on genetic algorithm," IEEE.

[6] Ahmed Gad, (2019) "Reproducing images using a genetic algorithm with python," Heartbeat Article.

[7] Ahmed Gad, (2019) "Introduction to optimization with genetic algorithm," KDNuggets.

[8] MIT License, (2018) "GARI python library," Pypi GARI.

[9] GeneticAlgorithmPython, (2021) "PyGAD python library," Pypi PyGAD.

[10] Ahmed Fawzy Gad, (2020) "5 Genetic Algorithm Applications Using PyGAD," Paperspace.

[11] Dana, Machine learning optimization using genetic algorithm, Architect and Industrial Engineer: Udemy Instructor, 2017.

[12] Thomas M. Wolfley, Genetic programming for image compression, Master of Science: University of California, Los Angeles, 2010.

[13] Tao Wu, The Journal. Image-Guided rendering with an evolutionary algorithm based on cloud model: Volume 2018, doi:10.1155/2018/4518265.

BIOGRAPHIES

Arif Chaudhary is pursuing B.E. in Computer Engineering. He has done various projects in this area like movie recommendation system, real time face recognition. His area of interest includes Machine Learning, Artificial Intelligence.

Harsh Singh is pursuing B.E. in Computer Engineering. He has done various projects in this area like movie recommendation system, real time face recognition. His area of interest includes Machine Learning, Artificial Intelligence.

Hrutuja Saswade is pursuing B.E. in Computer Engineering. She has done various projects in this area like movie recommendation system, real time face recognition. Her area of interest includes Machine Learning, Artificial Intelligence.

Deeraj Jain is pursuing B.E. in Computer Engineering. He has done various projects in this area like intelligent face mask. His area of interest includes Artificial Intelligence.

Shiwani Gupta is pursuing Ph.D. in Technology and holds a M.Tech and B.Tech degree. She has around 70 publications in various journals and conferences. Her area of interest includes Machine Learning, Artificial Intelligence and Algorithms.