Noise Removal in an Image Using a Neuro-fuzzy Technique and Evolutionary Method

Report submitted for the partial fulfillment of the requirements for the degree of
Bachelor of Technology in
Information Technology

Submitted by

ARIJIT SHIT
11700214020

SUBROTO ROY
11700214073

SWARNAVA CHATTOPADHYAY
11700214081

Under the Guidance of **MR. SOUMYADIP DHAR**
Assistant Professor, Department of Information Technology
RCC Institute of Information Technology, Kolkata

RCC Institute of Information Technology
Canal South Road, Beliaghata, Kolkata – 700 015
[Affiliated to West Bengal University of Technology]

RCC INSTITUTE OF INFORMATION TECHNOLOGY

KOLKATA–
7OOO15, INDIA

CERTIFICATE

The report of the Project titled Noise removal in an image using a neuro fuzzy technique and evolutionary method submitted Arijit Shit (11700214020), Swarnava Chattopadhyay (11700214081), Subroto Roy (11700214073) of B.Tech.(IT) 8th Semester of 2018 has been prepared under our supervision for the partial fulfillment of the requirements for B Tech (IT) degree in Maulana Abul Kalam Azad University of Technology.
The report is here by forwarded

....
Mr. Soumyadip Dhar,
Assistant Professor,
Dept. of Information Technology,
RCCIIT,Kolkata

Counter signed by

…………………………………………..
Dr. Abhijit Das,
Associate Professor & Head,
Dept of Information Technology
RCC Institute of Information Technology,
Kolkata – 700015,
India

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **MR. SOUMYADIP DHAR** of the department of Information Technology, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

Place: RCCIIT, Kolkata

Date:

………………………………
Arijit Shit

………………………………
Swarnava Chattopadhyay

………………………………
Subroto Roy

RCC INSITUTE OF INFORMATION TECHNOLOGY

KOLKATA– 7OOO15, INDIA

श्रमम् बिना न किमपि साध्यम्

CERTIFICATE of ACCEPTANCE

The report of the Project titled Noise removal in an image using a neuro fuzzy technique and evolutionary method submitted Arijit Shit (11700214020), Swarnava Chattopadhyay (11700214081), Subroto Roy (11700214073) of B.Tech.(IT) 8th Semester of 2018 has been prepared under our supervision for the partial fulfillment of the requirements for B Tech (IT) degree in Maulana Abul Kalam Azad University of Technology.

Name of the Examiner	**Signature with Date**
1. ……………………………………	……………………………
2. ……………………………………	…………………………….
3. ……………………………………	…………………………….
4. ……………………………………	…………………………….
5.……………………………………	…………………………….
6.…………………………………….	…………………………….

INDEX

1. Introduction

Image noise is a random variation of brightness and color information in an image and is usually an aspect of electronic noise. Image noise can range from almost imperceptible specks on a digital photograph taken in good light, to optical and radio-astronomical images that are almost entirely noise, from which a small amount of information can be derived by sophisticated processing. Such a noise level would be unacceptable in a photograph since it would be impossible even to determine the subject. Noise can seriously affect quality of digital images. Noise removal is one of the most important areas within digital image processing. One type of noise that can appear in images is impulse noise, which can be produced during image acquisition, storage, or transmission and can affect later stages of processing if not removed properly while preserving the details . This problem arises in many fields, from medical imaging to the analysis of satellite images. Impulse noise appears when some of the pixels in the image are replaced by outliers while the rest remain unchanged. Outliers can have a fixed minimum or maximum gray-scale value or may vary within that range. The first type is known as "salt and pepper" and is the one analyzed in this paper. Fat-tail distributed or "impulsive" noise, sometimes called salt-and-pepper noise or spike noise is very common form of corruption. An image containing salt-and-pepper noise will have dark pixels in bright regions and bright pixels in dark regions. This type of noise can be caused by analog-to-digital-convertor errors, bit errors in transmission, etc. Very different from Gaussian noise, the impulse noise is not evenly distributed throughout an image but it is randomly arranged, meaning all the image pixels are not corrupted but the corruption is scattered all throughout the image. This uncorrelated nature of impulse noise makes it a challenging task for identification and de-noising. Fuzzy models can represent a very appropriate solution to this problem because uncertainty generally affects the process of extracting information from corrupted data.

Fuzzy logic offers us a powerful tool to represent and process human knowledge in form of fuzzy if-then-else rules. Fuzzy sets are a generalization of the classical set theory and its ability to capture imperfections has been utilized in various fields over the last few decades. Fuzzification of the image makes it easier to identify the correctness of an image pixel by comparing it with its neighboring pixels. This mimic of human knowledge through fuzzy logic can be utilized more efficiently if human learning could also have been utilized.

Artificial Neural Networks (ANN) based approaches are best suited for this representation of human learning and processing. ANN's architecture is completely dependent on the goal that is intended to achieve. Massive connectivity amongst the neurons makes it more fault tolerant. Performance of an ANN is heavily dependent on the selection of appropriate parameters that governs the architecture. Parameter selection depends on the efficient search algorithms for complex search spaces. Nature inspired Bat Algorithm (BA) introduced in the year 2010 by Xin-She Yang is a very efficient algorithm for this searching.

This project aims to implement fuzzy logic, artificial neural network and Bat Algorithm to efficiently remove impulse noise from images.

Applications of impulse noise-removal from an image:

Visual information is the most important type of information perceived, processed and interpreted by the human brain. One third of the cortical area of the human brain is dedicated to visual information processing. Image filtering as a computer-based technology, carries out manipulation and interpretation of such visual information, and it plays an increasingly important role in many aspects of our daily life, as well as in a wide variety of disciplines and fields in science and technology, with applications such as television, photography, medical diagnosis and industrial inspection.

• Computerized photography (e.g., Photoshop)
• Space image filtering (e.g., Hubble space telescope images, interplanetary probe images)
• Clear detection of broken limbs through noise removal from X-ray plates
• Clear detection of Cancer through noise removal from mammographic plates
• Recovery of ancient and historical images through noise removal
• Recovery of distorted images which is required in crime investigation
• Noise removal of highly corrupted data

2. Problem Analysis:

Aim: To implement fuzzy logic, artificial neural network and Bat Algorithm to efficiently remove impulse noise from images.

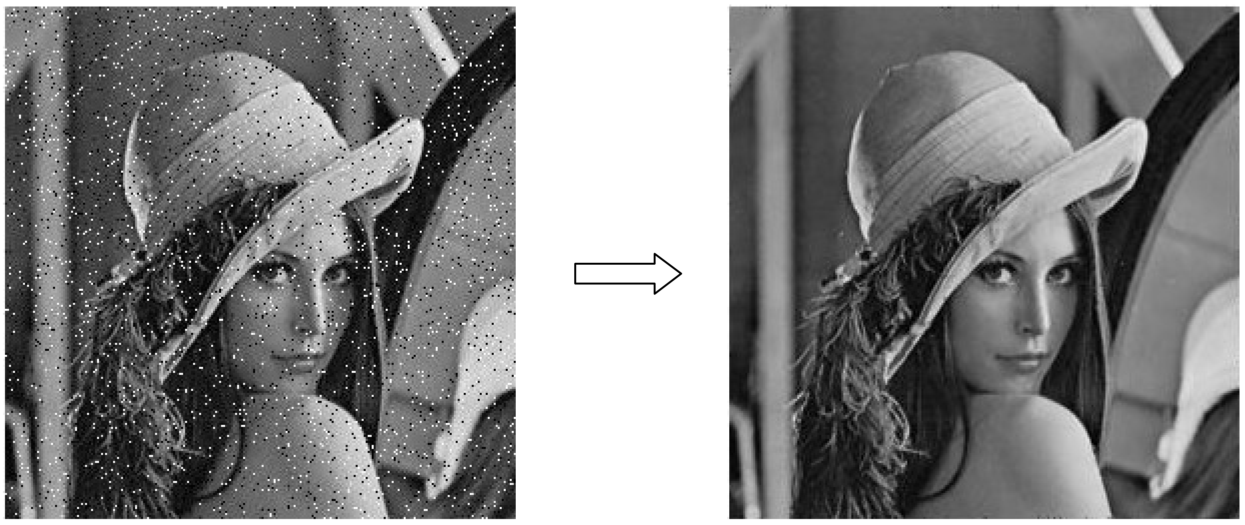
Problem definition:

The impulse noise removal of an image can be formulated as follows:

Given an input image of an entity, how can we determine a collection of optimized parameters as a result of bat algorithm and how it can be fed to the neural network which will remove impulse noise from that image and produce a noise-free image as an output of the neural network.

Broad Classification:

- Study and analysis of fuzzy logic, artificial neural network and Bat Algorithm
- Implementation of all the methods
- Training the neural network and feeding the optimal parameters of the neural network through which satisfactory results can be achieved.
- Analyzing the results in the form of:



(Noisy image) (Noise free image)

Fig. 1

3. Review of Literature

- Impulse noise corruption is very common in digital images. Impulse noise is always independent and uncorrelated to the image pixels and is randomly distributed over the image. Hence unlike Gaussian noise, for an impulse noise corrupted image all the image pixels are not noisy, a number of image pixels will be noisy and the rest of pixels will be noise free. There are different types of impulse noise namely salt and pepper type of noise and random valued impulse noise.[1]

- Definition and Applications of a Fuzzy Image Processing Scheme.[3]

- In this paper, a novel two-stage noise removal algorithm to deal with impulse noise is proposed. In the first stage, an adaptive two-level feed forward neural network (NN) with a back propagation training algorithm was applied to remove the noise cleanly and keep the uncorrupted information well. In the second stage, the fuzzy decision rules inspired by the human visual system (HVS) are proposed to classify the image pixels into human perception sensitive class and non-sensitive class, and to compensate the blur of the edge and the destruction caused by the median filter.[11]

- Image enhancement plays a vital role in various applications. There are many techniques to remove the noise from the image and produce the clear visual of the image. Moreover, there are several filters and image smoothing techniques available in the literature. All these available techniques have certain limitations. Recently, neural networks are found to be a very efficient tool for image enhancement. A novel two-stage noise removal technique for image enhancement and noise removal is proposed in this paper. In noise removal stage, Adaptive Neuro-Fuzzy Inference System (ANFIS) with a Modified Levenberg-Marquardt training algorithm was used to eliminate the impulse noise.[6]

- A new method for de-noising remote-sensing images based on partial differential equations (PDEs) is proposed. The method employs the similarity between the different band images in a multi-component image. Initially, one of the noise-free images in multi-component remote-sensing images as a prior is introduced into the PDE de-noising method. To make use of the priors of the noise-free image in de-noising, we construct a new smoothing term for the PDE so as to compute the total variation.[7]

- Evolutionary neural fuzzy filters are a new class of nonlinear filters for image processing. The original network structure of these filters adopts fuzzy reasoning in order to cancel noise without destroying fine details and textures. The learning method based on the Genetic Algorithms yields very satisfactory results within a few generations. Experimental results have shown that evolutionary neural fuzzy filters are very effective in removing impulse noise from highly corrupted images and significantly outperform conventional techniques. This chapter aims at providing a detailed description of the network architecture of these filters focusing on fuzzy set-based operations, encoding schemes and training procedures.[1]

- Metaheuristic algorithms such as particle swarm optimization, fire fly algorithm and harmony search are now becoming powerful methods for solving many tough optimization problems. In this paper, we propose a new met heuristic method, the Bat Algorithm, based on the echolocation behavior of bats. We also intend to combine the advantages of existing algorithms into the new bat algorithm. After a detailed formulation and explanation of its implementation, we will then compare the proposed algorithm with other existing algorithms, including genetic algorithms and particle swarm optimization. Simulations show that the proposed algorithm seems much superior to other algorithms, and further studies are also discussed.[4]

4. Network Structure for impulse noise removal:

Suppose we deal with images having L gray levels. Let x(n) be the pixel luminance at location n=[n1,n2] in the noisy image and let x1(n), x2(n),…,x8(n) be the luminance values of eight neighboring pixels as shown below

X_1(n)	**X_2(n)**	**X_3(n)**
X_8(n)	**X(n)**	**X_4(n)**
X_7(n)	**X_6(n)**	**X_5(n)**

Fig 2. Pixels belonging to a 3x3 neighborhood

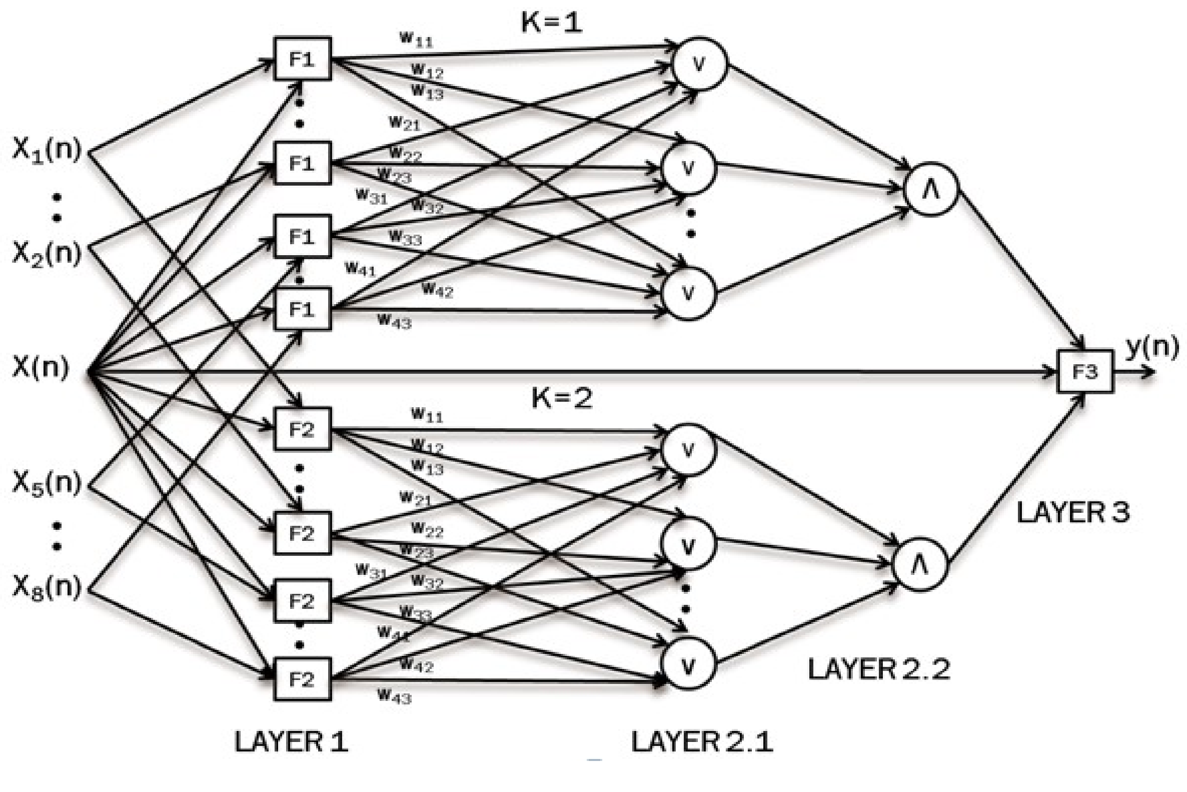


Fig 3. Network Architecture

Figure 3 denotes the neural structure of the evolutionary fuzzy filter. Square denotes the nodes that perform fuzzy set-based operations. Circles denote minimum and maximum operators. The filter is formed by two symmetrical sub-networks that aim at detecting positive and negative noise pulses respectively.

Layer 1(fuzzification layer): This layer is named as fuzzification layer because it simply performs fuzzification of the input window and passes the result on to the next layer. The node expression is represented as follows:

$O_{k,j}^{1} = m_{F_k}(x_i, x);\ k = 1,2;\ i = 1, \dots \dots .., 8$ ……………..(1)

where m$_{F1}$ and m$_{F2}$ are two membership functions.

$m_{F_1}(x_i, x) = \frac{(x_i - x + L - 1)}{2(L-1)}$ …………….(2)

$m_{F_2}(x_i, x) = \frac{(x - x_i + L - 1)}{2(L-1)}$ …………….(3)

$where\ 0 < x_i < L - 1\ and\ 0 < x < L - 1.$

Layer 2.1: Let $O_{k,j}^{2}$ be the output of the j-th node in the k-th sub-network. The node function is yielded by:

$O_{k,j}^{(2)} = MAX_{i=1,\dots,8}\{w_{i,j}, O_{k,i}^{(1)}\};\ k = 1,2; j = 1, \dots, M$ .....………(4)

Where M is the number of nodes in each sub-network. Value of M is to be chosen according to requirement. $w_{i,j}$ represent the strength of the connection between the nodes of layer 1 and layer 2. Now each connection from layer 1 is directed towards a node of layer 2 whose weight is going to be either 0 or 1, implying it is a binary weight. More about the binary weight and M is described later in this paper.

Layer 2.2: This layer receives as input, the output of each of the node of layer 2 and the minimum of all the inputs is given as output. The node function is represented by the following equation:

$O_{K}^{(3)} = MIN\left\{O_{k,j}^{(2)}\right\};\ k = 1,2. j = l, \dots, M$ ……..…….(5)

Layer 3: This layer is the final and the output layer. This layer simply evaluates the correction term and adds it to the original pixel.

$$y(n) = x(n) + (L - l)mLA(|\Delta y0|, x)\ if\ \Delta y0 > 0$$

$$= x(n) - (L - l)mLA(|\Delta y0|, x)\ if\ \Delta y0 < 0$$

$$Where\ \Delta y0 = (L - 1)(O_1^{(3)} - O_2^{(3)})$$

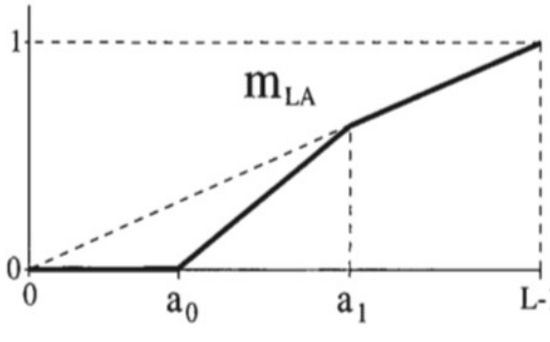


Fig. 4 - Fuzzy set LA.

The shape of fuzzy set LA plays a key role in preserving the image details during the process of noise removal. Indeed, the smoothing action gradually ranges from full correction to no correction, depending on the estimated amplitude $|\Delta y_0|$ of a noise pulse. When this amplitude is large ($m_{LA}(|\Delta y_0|)== 1$), full correction is produced. When this amplitude is small ($m_{LA}(|\Delta y_0| << 1$), the filtering action is further reduced. In fact, the smaller the value of $|\Delta y_0|$ the more uncertain we can be that the processed pixel really represents a noise pulse. The cancellation of noise can be improved by enabling the removal of small-amplitude noise pulses when the pixel luminance x(n) is not medium. This effect can be easily achieved by varying the slope of fuzzy set LA as follows: $a_0=a'_1\ m_{MD}(x)$, where MD (medium) is a trapezoid-shaped fuzzy set centered on L/2 (Fig.5) and a'l =a1

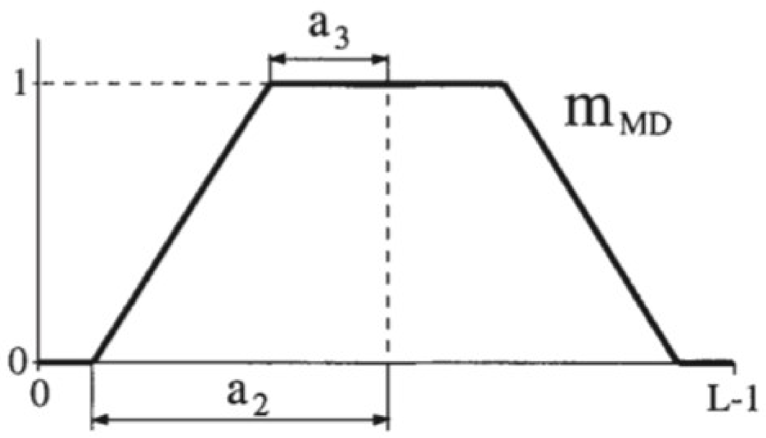


Fig.5 - Fuzzy set MD.

According to the network structure described in the previous section, the filtering behavior depends on the choice of 8M weights $\{W_{i,j}\}$ and three fuzzy set parameters a1, a2 and a3.

The correct choice of the weights of the connections between layer1 and layer2 will determine the efficiency of the network. So, the weights are determined with help of Bat Algorithm. Also the number of weights is dependent on the number of nodes in layer 2 i.e. M. As evident from equation 4, each output from layer 1 is multiplied with its respective connectionsweightinlayer2.Forexamplesupposethenumber of nodes in layer 2.1 is 4. This implies that the number of connection from layer 1 to 2 will be 8*4=32 thus there will be 32 binary weights required. The first 8 weights are for the connections from layer 1 to the first node of layer 2.1 similarly the second 8 weights are intended for the second node of layer 2.1 and so on.

BAT ALGORITHM: Interesting echolocation behavior of bats has been observed by Xin-She Yang and developed into an efficient optimization algorithm in huge search spaces. Bat algorithm has three generalized rules; they are:

1. All bats use echolocation to sense distance, and they also know the difference between food/ prey and background barriers in some magical way.

2. Bats fly randomly with velocity vi at position xi with a fixed frequency f_{min} , varying wavelength and loudness A_0 to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse emission r [01] depending on the proximity of the target.

3. Although the loudness can vary in many ways, here it is assumed that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}. Main steps of the algorithm are given below: 1. Initialization; Repeat 2. Generation of new solutions 3. Local searching; 4. Generation of a new solution by flying randomly; 5. Finding the current best solution; until (requirements are met). In BA algorithm, initialization of the bat population is performed randomly. Generating new solutions is performed by moving virtual bats according to the following equations:

$$f_i = f_{min} + (f_{max} - f_{min}) \qquad \text{..........(6)}$$

$$v_t^i = v_t^i - 1 + (x_t^i - x *)f_i \qquad \text{...........(7)}$$

where β€[01] is a random vector drawn from a uniform distribution. Here x* is the current global best location (solution) which is located after comparing all the solutions among all the bats. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{max}, f_{min}]$.

A random walk with direct exploitation is used for the local search that modifies the current best solution according to the equation:

$$x_{new} = x_{old} + A_t \qquad \text{….......(8)}$$

where [01] a random number, while A_t is the average loudness of all the bats at this time step.

Bat Algorithm requires an objective function which it is going to optimize. For our problem it is clear that we are reducing noise. So lower the mean square error the better. Thus,

$$F = \sum_n (y(n) - s(n))^2 \qquad \text{............(9)}$$

Here y(n) is the processed pixel and s(n) is the original, noise free pixel.

5. Problem Discussion:

Image noise filtering presents a challenging problem in the field of image analysis and as such has received a great deal of attention over the last few years because of its many applications in various domains. Experimental results have shown that evolutionary neural fuzzy filters are very effective in removing impulse noise from highly corrupted images and significantly outperform conventional techniques. This paper aims at providing a detailed description of the network architecture of these filters focusing on fuzzy set-based operations, encoding schemes and training procedures. It also mentions some of the algorithms related to this purpose, especially the bat algorithm proposed by Xin-She Yang . A detailed architecture of a neuro-fuzzy technique for image filtering that uses Bat Algorithm (BA) for the parameter optimization of the neural network and its learning has been mentioned in a lucid manner.

Impulse noise removal using the proposed method:

Since reducing image noise presents a classical image processing problem, a great number of methods to reduce it have been proposed. In some methods every pixel in the image is restored regardless of whether it was originally noisy or not. The general scheme of this type of filtering is the use of a mask which is normally centered around the pixel of interest for computational reasons. The mask is used to sweep the image and perform some operations with the pixels inside it in order to obtain the reconstruction value. The main advantage of our project is its simplicity, especially when the probability of noise is high. None of the methods proposed in the literature presents a better performance in all scenarios, that is, for different added noise ratios or for all the images evaluated.

6. Implementation of Network Structure Using Bat Algorithm:

The number of nodes in layer 2.1 is fixed to be M. Thus we will be requiring M*8+3 parameters for a1, a2, a3 to be optimized using BA. For training of the neural network we need to follow these steps.

Step 1: Input the noisy image and the original image.

Step 2: Repeat step-3 to step 7 till no change in objective function is observed or number of iterations completes.

Step 3: Generate new solutions by the Eq. (5), Eq. (6) and Eq. (7).

Step 4: if rand > r_i then random walk around one of best solutions.

Step 5: Generate a new solution by flying randomly and calculate F. Calculation of F will require calling of the neural network.

Step 6: If rand < A_i and $F(x_i) < F(x_i*)$then accept new solutions.

Step 7: Find the current best x^*.

Step 8: Store the best x^* this implies the best and optimized parameters for the neural network.

Now, to remove noise from an image using a trained neural network we need to follow these steps:

Step 1: Input the noisy image.

Step 2: For each 3x3 window of the noisy image as shown in figure 1, repeat step 3 step 5

Step 3: Fuzzify the window using equation (1).

Step 4: Evaluate the maximum and minimum operations of layer 2.

Step 5: Evaluate the correction term and add it to the noisy pixel.

Step 6: Display the noise free image as output.

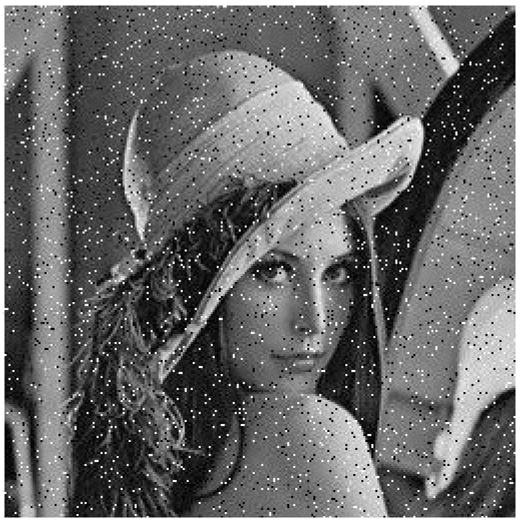
7. **Results & Discussion**:

Here, we have trained the neural network for 40 generations, each generation having a population size of 40. The network structure had 8 nodes in level 2.1 meaning M=8. It is evident from the network architecture shown in figure 3 that it will require 8*M that is 8*8=64 weights to be optimized. Now having with the 64 weights and additional 3, that is a total of 67 parameters would be required to be optimized by the bat algorithm. To avoid such huge number of parameter what we tried to do is we optimized only two set of 8 weights that is a total of 16 weights and remaining 3 parameter for a_1,a_2,a_3, reducing it to a total of 19 parameter.

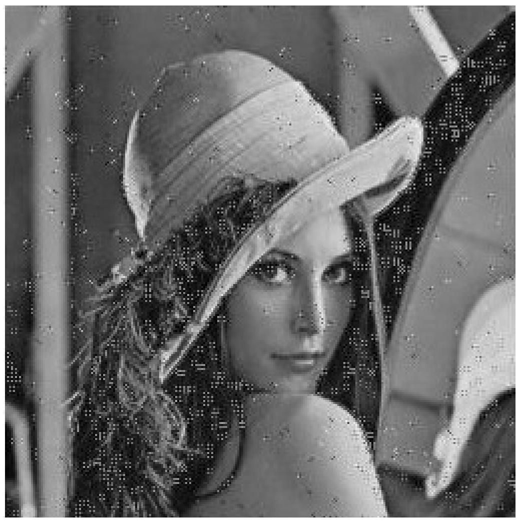
Now we take the first 8 parameters. These form the connection weights between level 1 and first node of level 2.1. Now we rotate these eight weights by 90 degree to get another set of weights that form a connection between level 1 and 2md node of level2.1, similarly we again rotate is by 180 degree and 270 degree to form the connection between layer 1 and 2nd and 3rd node of layer 2.1. Same is repeated with the next set of 8 parameters to get the total of 64 weights and 3 parameter for a_1,a_2,a_3.

At the end of each generation for all the 40 generations, the best result was shown as output. We are feeding lena.jpg and cameraman.tif as input and among the series of images, we are giving the best result of each generation.

Lena.jpg



(1)



(2)

⇨

(3)

(4)

⇨

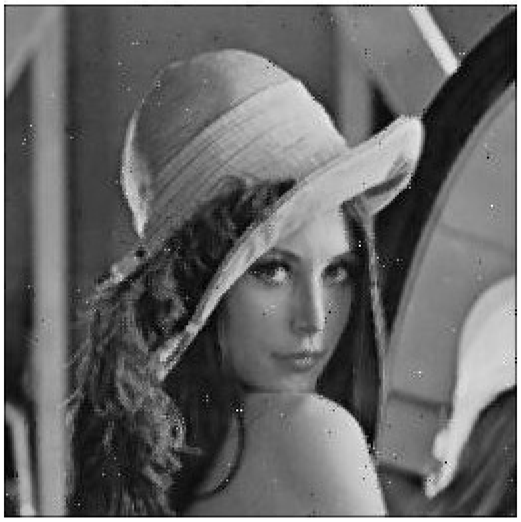
(5)

(6)

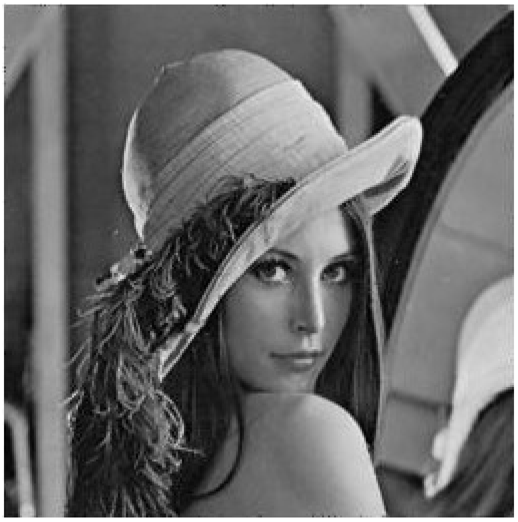
⇨

(7)

(8)



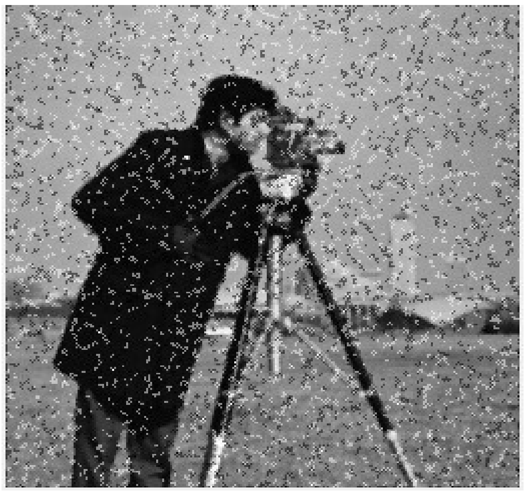
(9)



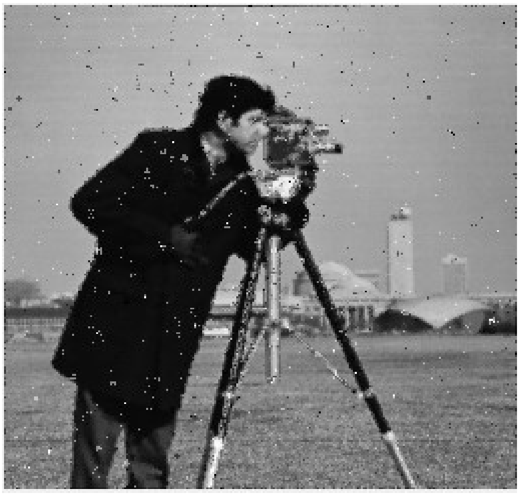
(10)

Fig6. Transition of the image through various generations of the training

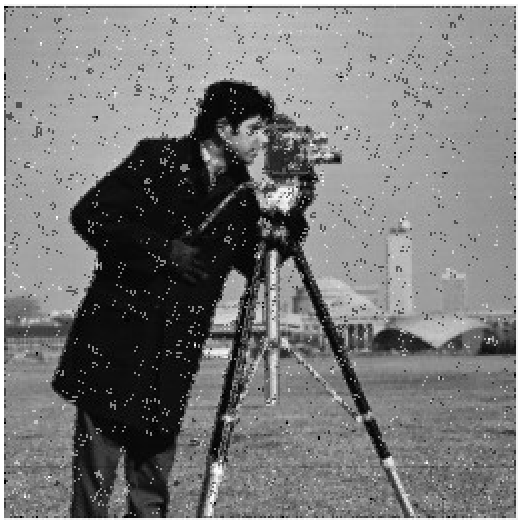
Training with Cameraman.tif



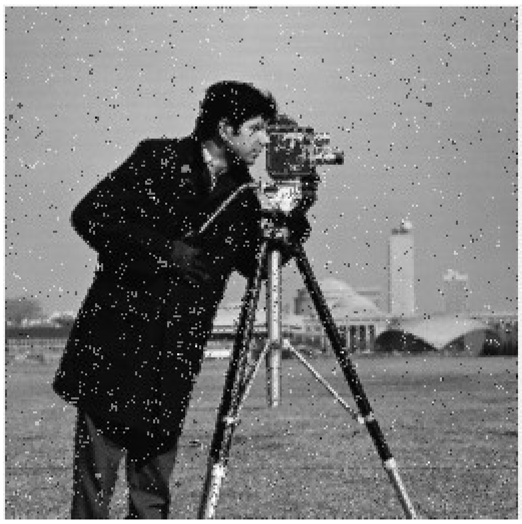
(1)



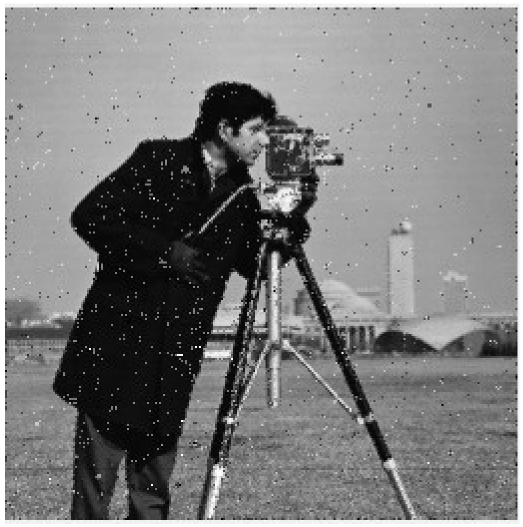
(2)

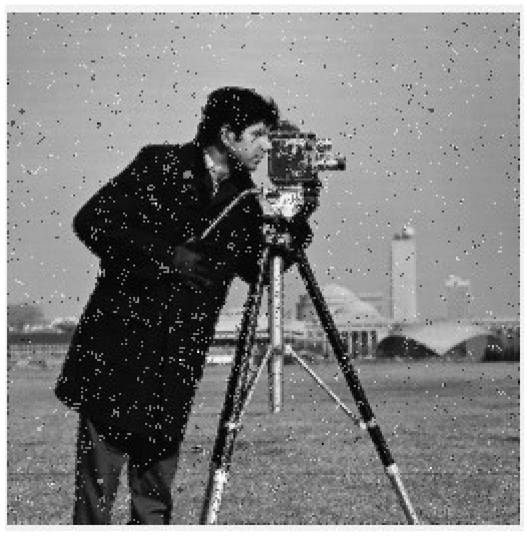


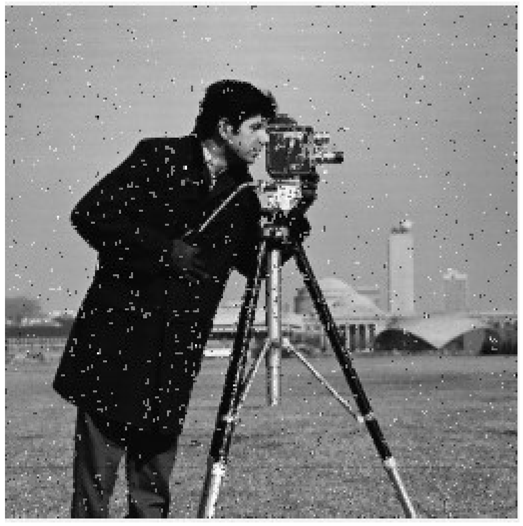
(3)

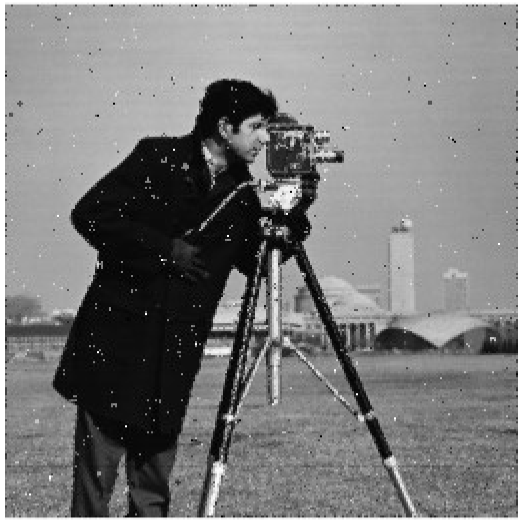


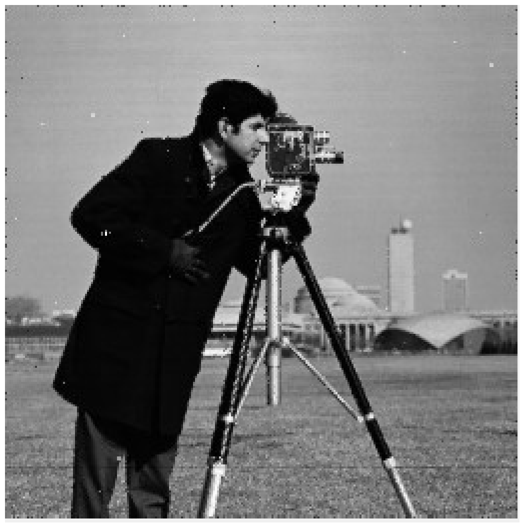
(4)


(5)


(6)


(7)


(8)


(9)

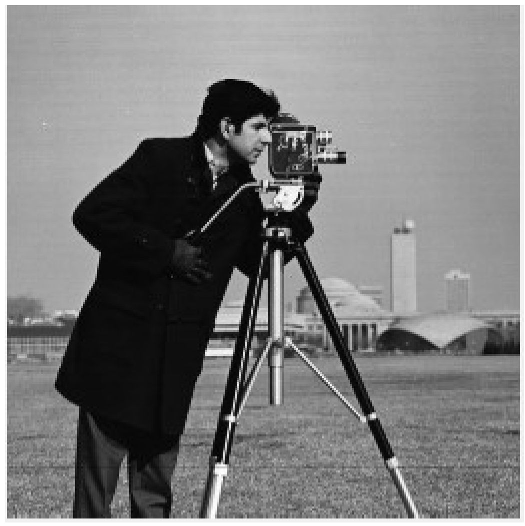

(10)

Fig7. Transition of the image through various generations of the training

The number of images shown here are very less and for only an few generation. It is because after a few generations the image showed no further improvement, indicating that the optimized parameters have been acquired and no more optimization is possible. Although stopping at this instance is not feasible because often the algorithm gets stuck for a few generations at a local minima and only after a few more generations it again regains its property of minimizing the optimization function. The graph shown below shows one of such cases.

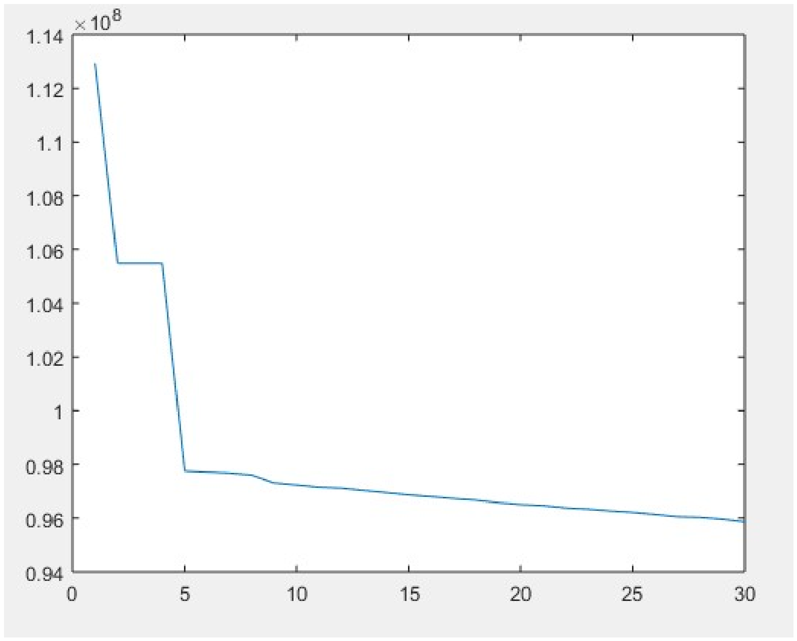


Fig 8. Optimization function stuck at local minima

It is evident from the graph that the algorithm was stuck at a local minima soon before generation 5 and only after generation 5 it began its optimization operation again. Thus to avoid such a situation the optimization was not stopped before completing its complete generation count.

8. Output With Screenshots:

The neural network shown above in figure 3 has been trained on the following noisy images.

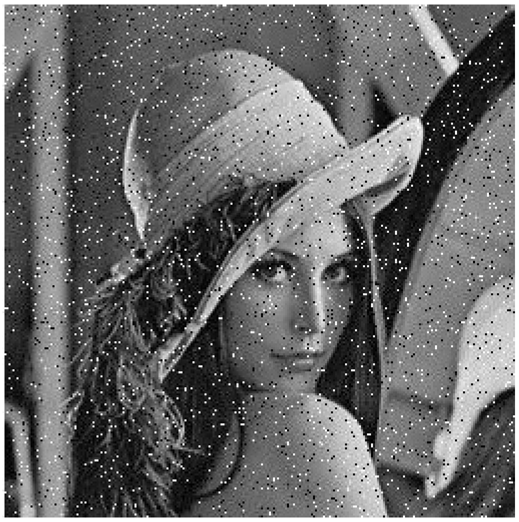


Fig. 9(a) Training Image 'lena.jpg' having salt and pepper noise of probability 0.1

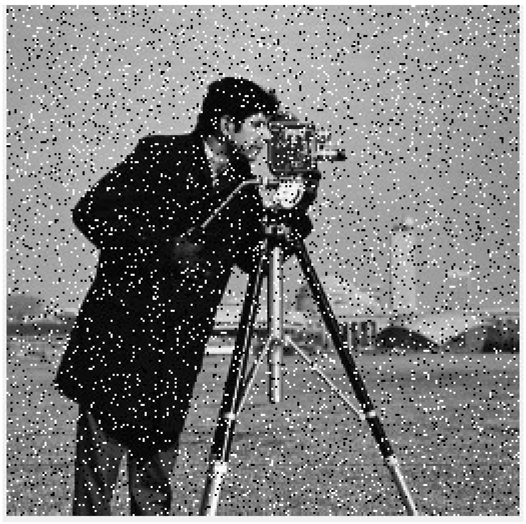


Fig. 9(b) Training Image 'Cameraman.tif' having salt and pepper noise of probability 0.1

During the training a graph of fitness in equation (9) against the number of generation was plotted as shown in figure 7.

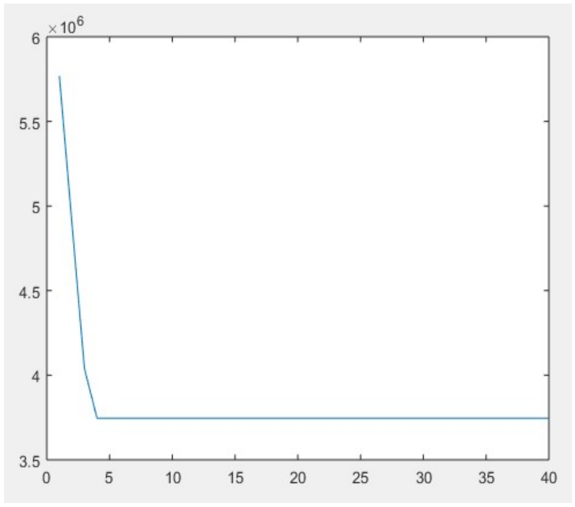


Fig. 10 Fitness versus number of generations

It is evident from the graph that the optimal value of the parameters were reached by nearly the 5th and the 10th generation although the searching was continued through the rest of the generation and not stopped to avoid local minima.

When the parameters of the neural network after optimization were as follows,

w1=0.246086652000000

w2=0.782806842000000

w3=0.453340060000000

w4=0.00585256500000000

w5=0.608857973000000

w6=0.0358862720000000

w7=0.202246905000000

w8=0.998073431000000

w9=0.0186071980000000

w10=0.337406363000000

w11=0.932139487000000

w12=0.879809250000000

w13=0.0129585350000000

w14=0.448374089000000

w15=0.992009397000000

w16=0.00818476300000000

a1=254.988174800000

a2=107.859023300000

a3=127.236886521634

These when fed into the neural network gave the following results with the test images;

Image1. lena.jpg

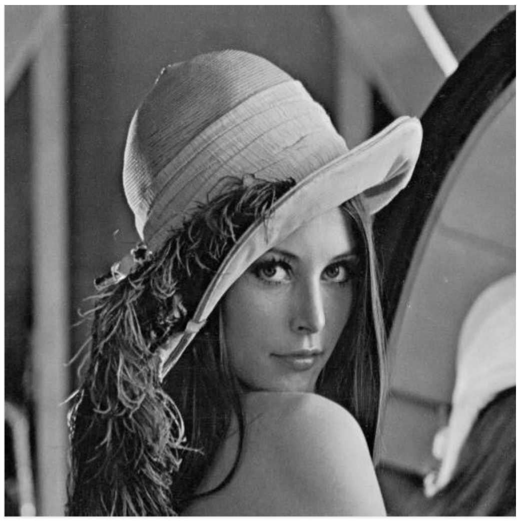


Fig.11 (a) Original image

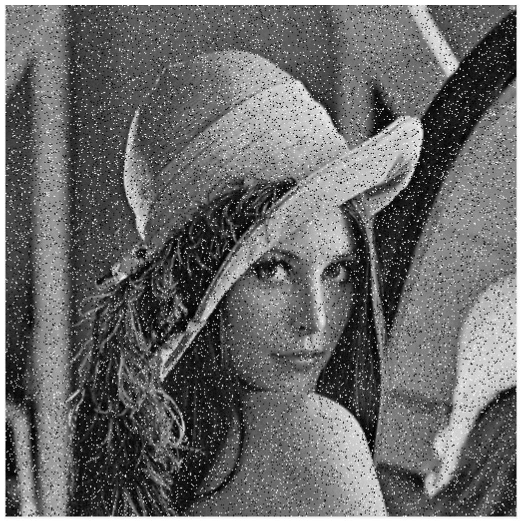


Fig.11 (b) Salt and pepper noise of probability 0.1 added to the original image

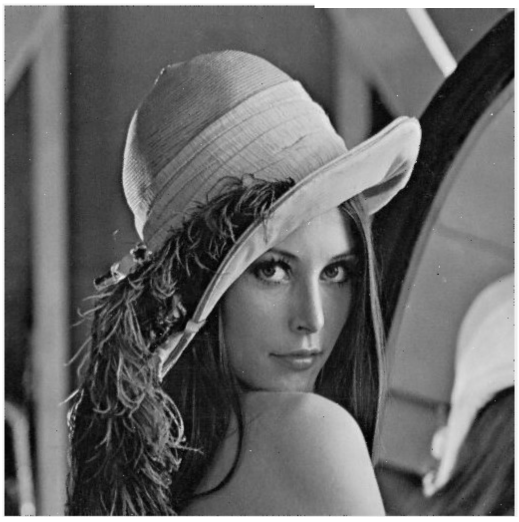


Fig 11 (c) Image after noise reduction

Image 2. rice.png

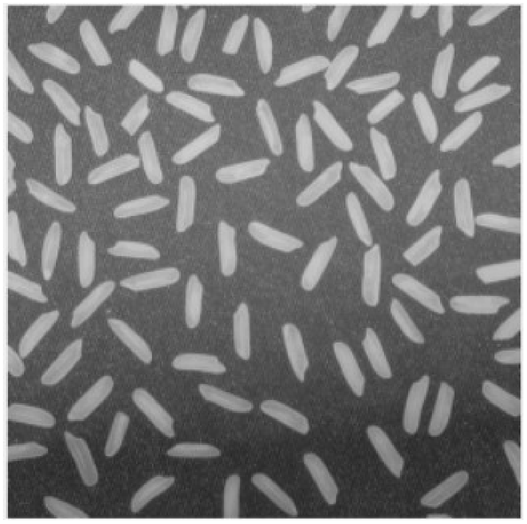


Fig 12 (a).Original image

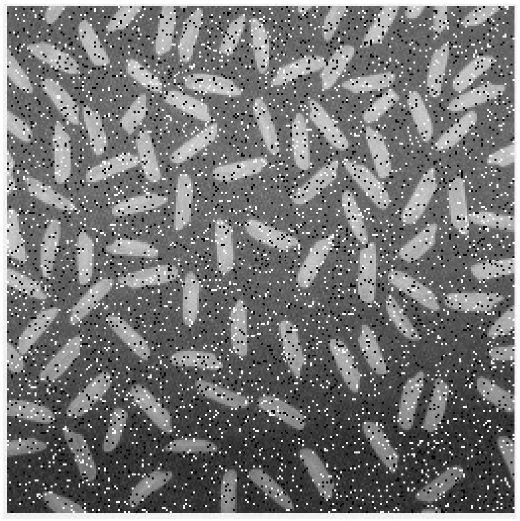


Fig 12 (b) Salt and pepper noise having probability of 0.1

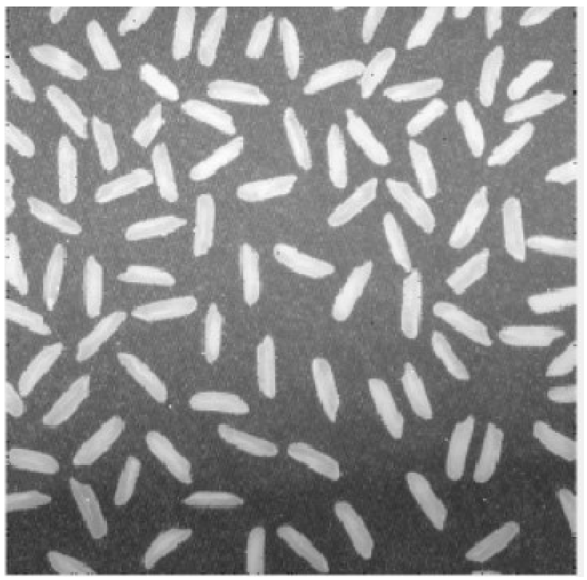


Fig 12 (c) Image after noise reduction

Image 3. Cameraman.tif

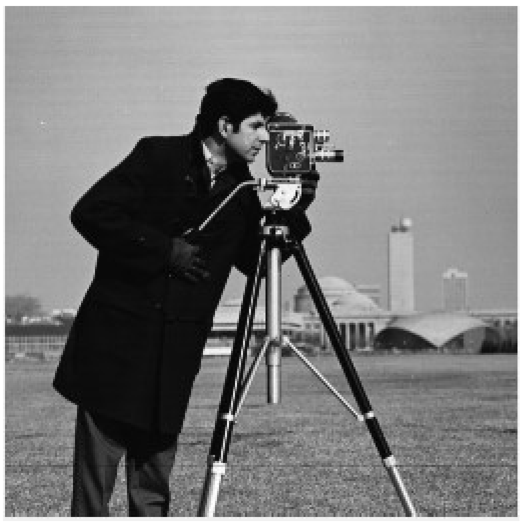


Fig 13 (a). Original image

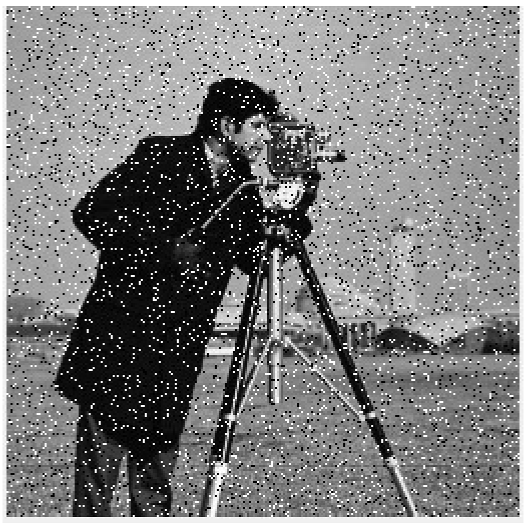


Fig 13(b). Salt and pepper noise of probability 0.1 added to the original image

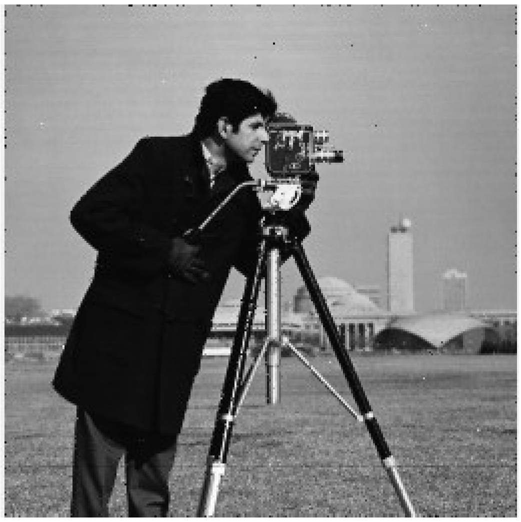


Fig 13(c) Image after noise reduction

Image 4 pout.tif

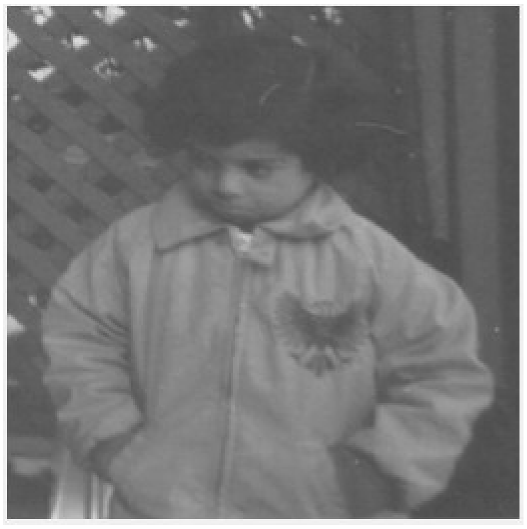


Fig 14(a). Original image

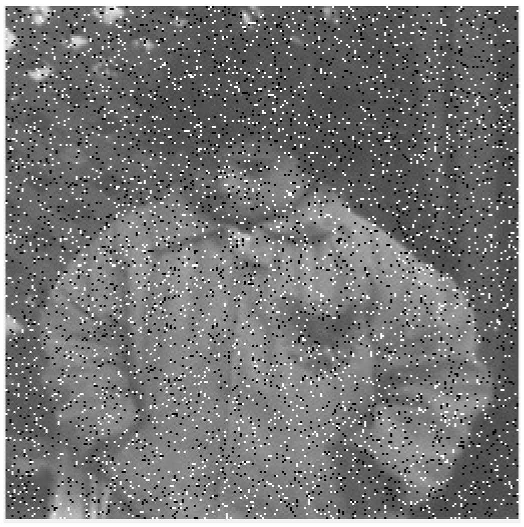


Fig 14(b). Salt and pepper noise of probability 0.1 added to the original image

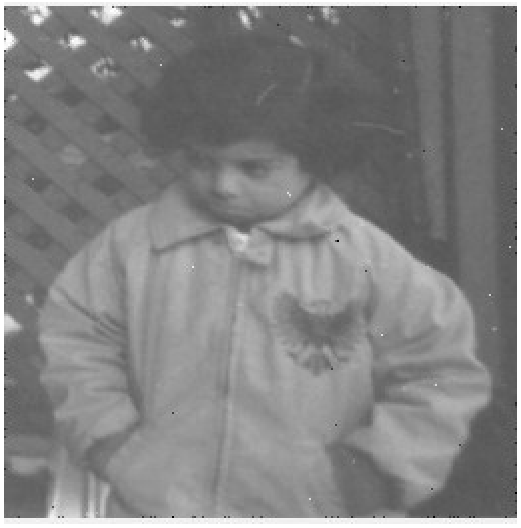


Fig 14(c) Image after noise reduction

Screenshots:

We have performed the filtering operation with a few other images and good the following result:

Rice.png:

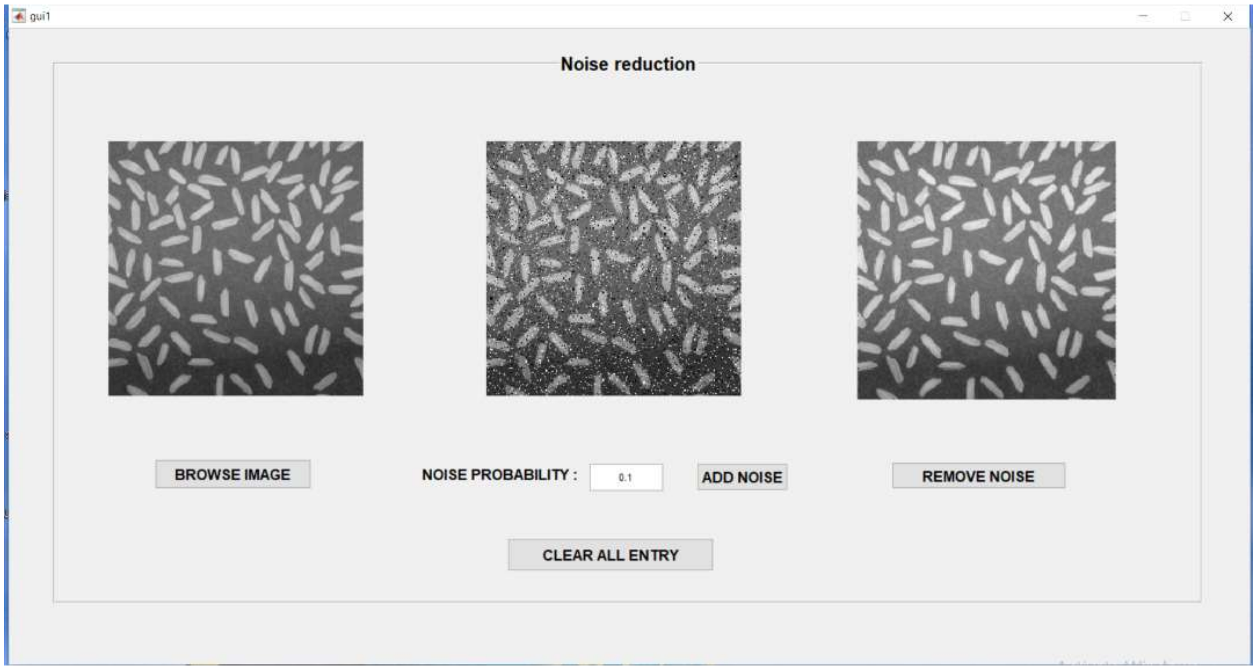


Fig. 14. Output Screenshot 1

Coin.png:

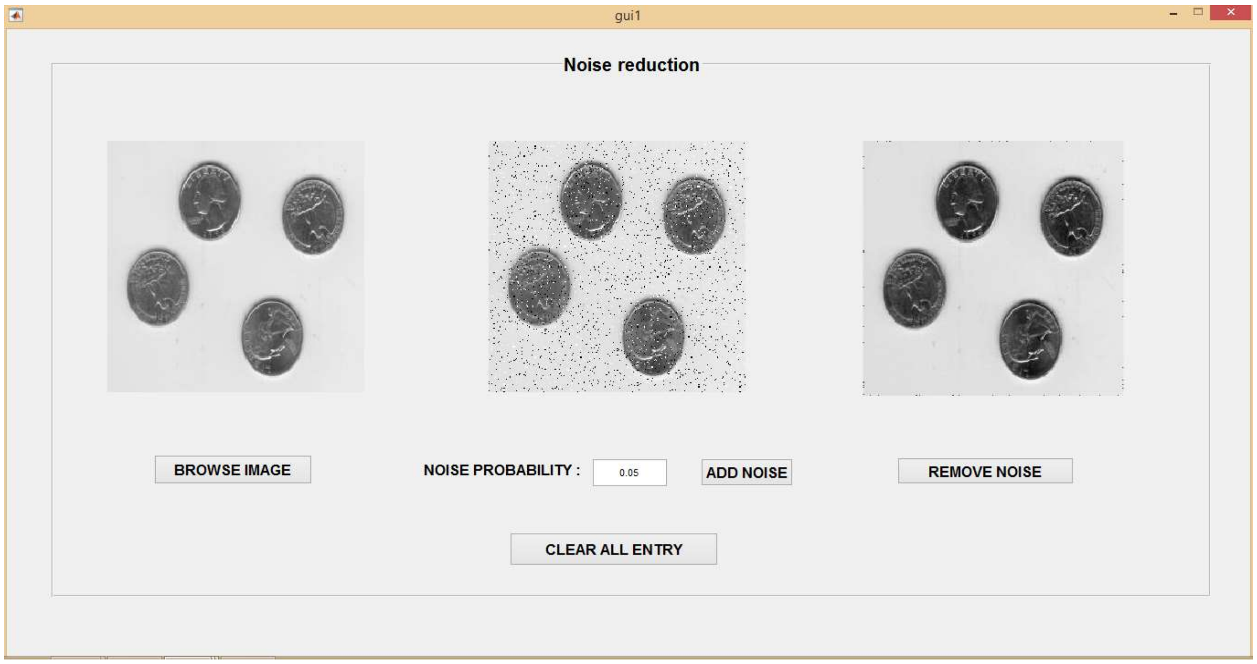


Fig .15 . Output Screenshot 2

Moon.jpg:

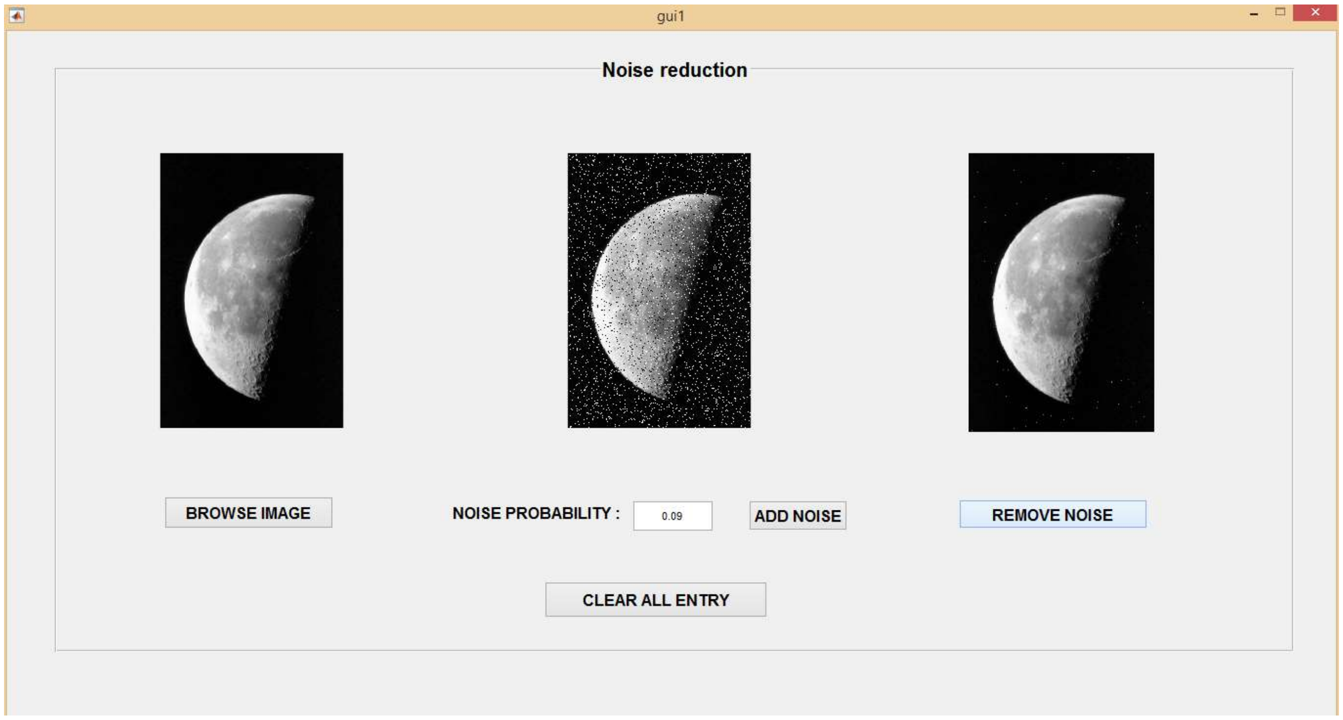


Fig.16. Output Screenshot 3

Pout.tif:

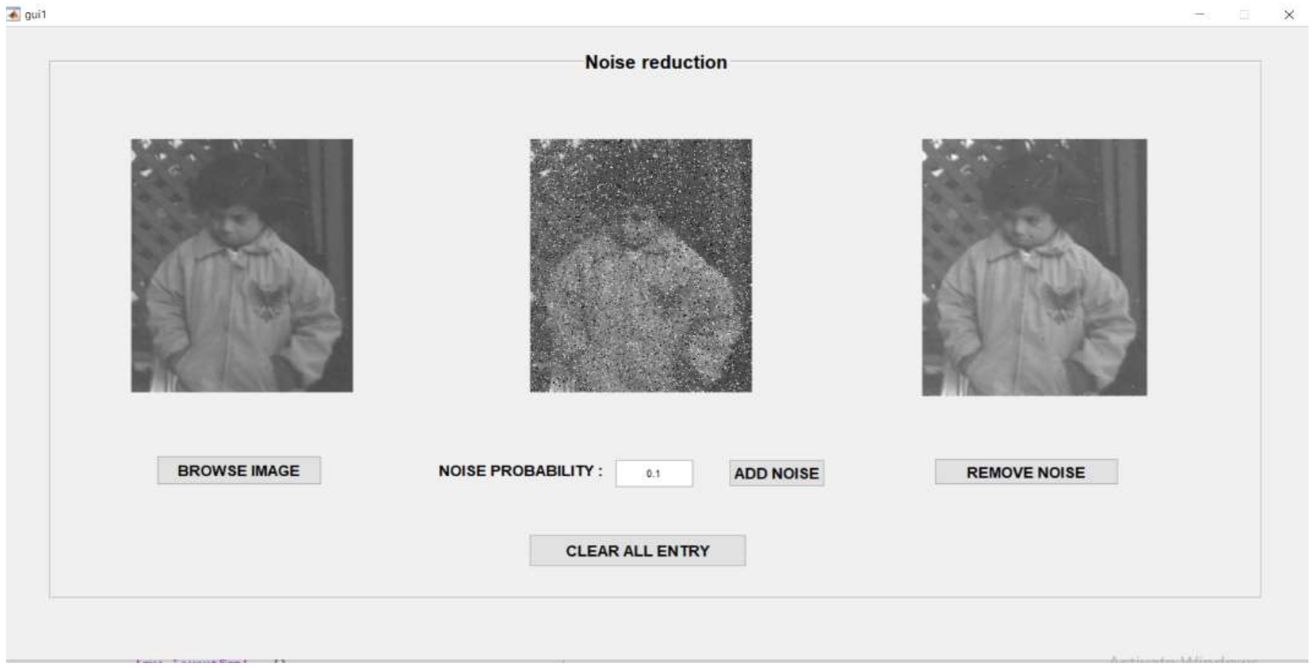


Fig.17

9. Conclusion / Future Scope of Work

This is to conclude that this project provides an evolutionary neural fuzzy filter for impulse noise removal. The approach is based on a closed integration of soft-computing techniques. It exploits the effective representation of knowledge that is a key feature of fuzzy models and is able to acquire this knowledge from a set of training data. The efficiency of the project is used to minimize the noise in an image to a great extent without affecting the original information of the image. Use of neural network gives the algorithm a scope to learn the noise patterns from already input images and Bat algorithm is able to significantly optimize the parameters. The network structure of a filter for highly corrupted data has also been described. Experimental results have shown that evolutionary neural fuzzy filters are able to significantly outperform classical operators which are based on old conventional techniques.

Future scope:

We intend to extend the application of the project to the following fields.

- Removal of noise from Highly corrupted data i.e. an image with high probability of noise. Highly corrupted image can be look like this,

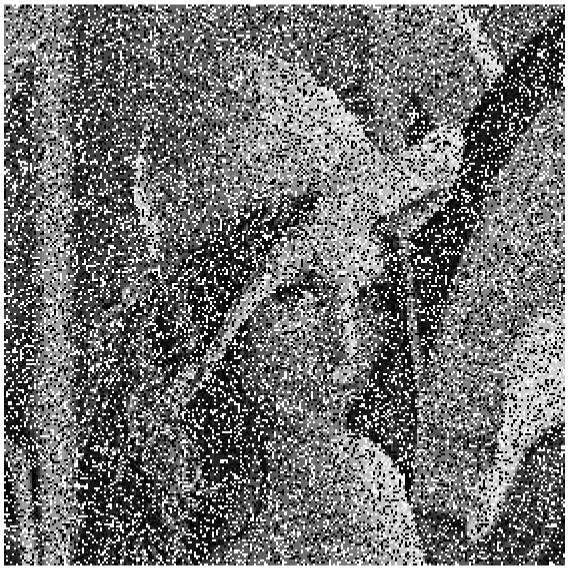


Fig 18. Highly corrupted image.

- Other than impulse noise removal, this project can provide assurance to remove any kind of noise with higher probabilities.
- Some applications related to medical field such as mammography ,i.e. detection of breast cancer can implement this to get a clear and noise free detection plate
- This project can also be implemented in the field of underwater photography
- This project can also be implemented to recover old and distorted images.
- This project can also be implemented for color image.

10. <u>References</u>

[1] Russo, F. (2000),Image Filtering Using Evolutionary Neural Fuzzy Systems, DEEI - Universita degli Studi di Trieste Via A. Valerio 10,1-34127 Trieste, Italy.

[2] Harikiran, J., Saichandana,B. and Divakar,B. (2010). Impulse Noise Removal in Digital Images. International Journal of Computer Applications (0975 8887) Volume 10 No.8.

[3] Chacon, Mario. (2007). Fuzzy Logic for Image Processing: Definition and Applications of a Fuzzy Image Processing Scheme. Advances in Industrial Control. 101-113. 10.1007/978-1-84628-469- 4 7.

[4] Yang, X.S. (2011). A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization. (NICSO). 284, 6574. doi:10.1007/978-3- 642-12538- 6 6.

[5] Sheng-Fu Liang, Shih-Mao Lu, Jyh-Yeong Chang, Member, IEEE, and Chin-Teng (CT) Lin, Fellow, IEEE. A Novel Two-Stage Impulse Noise Removal Technique Based on Neural Networks and Fuzzy Decision.

[6] V.Saradhadevi, Research Scholar, Karpagam University, Coimbatore, India. Dr.V.Sundaram, Director of MCA, Karpagam Engineering College, Coimbatore, India. A Novel Two-Stage Impulse Noise Removal Technique based on Neural Networks and Fuzzy Decision.

[7] Peng Liu Fang Huang, Guoqing Li Spatial Data Center, Center for Earth Observation and Digital Earth, Chinese Academy of Sciences , Beijing, China. Remote-Sensing Image Denoising Using Partial Differential Equations and Auxiliary Images as Priors.

[8] Rafael C.Gonzalez &Richard Woods, Digital Image Processing.

[9] B.Yegnanarayana, Artificial Neural Networks.

[10] Prof. Sankar K. Pal Dr. Ashish Ghosh Prof. Malay K. Kundu Machine Intelligence Unit Indian Statistical Institute 203 B.T. Road Calcutta 700035 India, Soft Computing for Image Processing.

[11] Sheng-FuLiang, Shih-MaoLu, Jyh-Yeong Chang and Chin-TengLin. A novel two-stage impulse noise removal technique based on neural networks and fuzzy decision

11. Appendix

Program Code

Start.m

```
close all;
clear;
clc;
i=imread('lena.jpg');
OI=imresize(i,[125 125]);
J = imnoise(OI,'salt & pepper',0.2);
figure,imshow(J);
I=double(J);
OI=double(OI);
[F, min, iter]=albat(I,OI);
```

albat.m

```
function [best,fmin,N_iter]=albat(Im,OI)

% Default parameters
%if nargin<1,  para=[20 5000 0.5 0.5];  end
para=[10 40 0.5 0.5];
n=para(1);      % Population size, typically 10 to 40
N_gen=para(2);  % Number of generations
A=para(3);      % Loudness  (constant or decreasing)
r=para(4);      % Pulse rate (constant or decreasing)
% This frequency range determines the scalings
% You should change these values if necessary
Qmin=0;         % Frequency minimum
Qmax=2;         % Frequency maximum
% Iteration parameters
N_iter=0;       % Total number of function evaluations
% Dimension of the search variables
d=19;           % Number of dimensions
% Lower limit/bounds/ a vector
Lb=zeros(1,d);
% Upper limit/bounds/ a vector
Ub=ones(1,16);
Ub(17)=255;
Ub(18)=(255/2);
Ub(19)=(255/2);
% Initializing arrays
Q=zeros(n,1);   % Frequency
v=zeros(n,d);   % Velocities

% Initialize the population/solutions
```

```
for i=1:n,
  Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
  Fitness(i)=Fun_mod(Sol(i,:),Im,OI);
end
% Find the initial best solution
[fmin,I]=min(Fitness);
%disp(['fmin:',num2str(fmin),'I:',num2str(I)]);
best=Sol(I,:)

% Start the iterations -- Bat Algorithm (essential part)  %
for t=1:N_gen,
    t
% Loop over all bats/solutions
        for i=1:n
          Q(i)=Qmin+(Qmin-Qmax)*rand;
          v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);
          S(i,:)=Sol(i,:)+v(i,:);
          % Apply simple bounds/limits
         Sol(i,:)=simplebounds(S(i,:),Lb,Ub);

%           Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub)
          % Pulse rate
          if rand>r
          % The factor 0.001 limits the step sizes of random walks
              S(i,:)=best+0.001*randn(1,d);
          end
          S(i,:)=simplebounds(S(i,:),Lb,Ub);
%           S(i,:);

     % Evaluate new solutions
           Fnew=Fun_mod(S(i,:),Im,OI);

     % Update if the solution improves, or not too loud
           if (Fnew<=Fitness(i)) & (rand<A) ,
                Sol(i,:)=S(i,:);
                Fitness(i)=Fnew;
           end

          % Update the current best solution
          if Fnew<=fmin,
                best=S(i,:);
                fmin=Fnew;
          end
        end
        fmin1(t)=fmin;
        N_iter=N_iter+n;
        fmin
end
ct=1:N_gen;
```

```
fmin1;
figure,plot(ct,fmin1);
% Output/display
%disp(['Number of evaluations: ',num2str(N_iter)]);
%disp(['Best =',num2str(best),' fmin=',num2str(fmin)]);

% Application of simple limits/bounds
function s=simplebounds(s,Lb,Ub)
 % Apply the lower bound vector
 ns_tmp=s;
 I=ns_tmp<Lb;
 ns_tmp(I)=Lb(I);

 % Apply the upper bound vector
 J=ns_tmp>Ub;
 ns_tmp(J)=Ub(J);
 % Update this new move
 s=ns_tmp;
```

<u>Fun_mod.m</u>

```
function z=Fun_mod(u,Im,OI)
I=Im;
OI=double(OI);
w1=u(1);w2=u(2);w3=u(3);w4=u(4);w5=u(5);w6=u(6);w7=u(7);w8=u(8);a1=u(17);a2=u(18)
;a3=u(19);
w9=u(9);w10=u(10);w11=u(11);w12=u(12);w13=u(13);w14=u(14);w15=u(15);w16=u(16);
main;
sizes=size(OI);
z=sum(sum((y-OI).^2));
```

<u>layer1_2_Net1.m</u>

```
function
c=layer1_2_Net1(a,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16)
p(1)=(a(1,1)-a(2,2)+256-1)/(2*(256-1));
p(2)=(a(1,2)-a(2,2)+256-1)/(2*(256-1));
p(3)=(a(1,3)-a(2,2)+256-1)/(2*(256-1));
p(4)=(a(2,3)-a(2,2)+256-1)/(2*(256-1));
p(5)=(a(3,3)-a(2,2)+256-1)/(2*(256-1));
p(6)=(a(3,2)-a(2,2)+256-1)/(2*(256-1));
p(7)=(a(3,1)-a(2,2)+256-1)/(2*(256-1));
p(8)=(a(2,1)-a(2,2)+256-1)/(2*(256-1));

d(1)=round(w1)*p(1);
d(2)=round(w2)*p(2);
d(3)=round(w3)*p(3);
d(4)=round(w4)*p(4);
d(5)=round(w5)*p(5);
d(6)=round(w6)*p(6);
```

```
d(7)=round(w7)*p(7);
d(8)=round(w8)*p(8);

m(1)=max(d);


d(1)=round(w7)*p(1);
d(2)=round(w8)*p(2);
d(3)=round(w1)*p(3);
d(4)=round(w2)*p(4);
d(5)=round(w3)*p(5);
d(6)=round(w4)*p(6);
d(7)=round(w5)*p(7);
d(8)=round(w6)*p(8);

m(2)=max(d);


d(1)=round(w5)*p(1);
d(2)=round(w6)*p(2);
d(3)=round(w7)*p(3);
d(4)=round(w8)*p(4);
d(5)=round(w1)*p(5);
d(6)=round(w2)*p(6);
d(7)=round(w3)*p(7);
d(8)=round(w4)*p(8);

m(3)=max(d);

d(1)=round(w3)*p(1);
d(2)=round(w4)*p(2);
d(3)=round(w5)*p(3);
d(4)=round(w6)*p(4);
d(5)=round(w7)*p(5);
d(6)=round(w8)*p(6);
d(7)=round(w1)*p(7);
d(8)=round(w2)*p(8);

m(4)=max(d);

d(1)=round(w9)*p(1);
d(2)=round(w10)*p(2);
d(3)=round(w11)*p(3);
d(4)=round(w12)*p(4);
d(5)=round(w13)*p(5);
d(6)=round(w14)*p(6);
d(7)=round(w15)*p(7);
d(8)=round(w16)*p(8);
```

```
m(5)=max(d);


d(1)=round(w15)*p(1);
d(2)=round(w16)*p(2);
d(3)=round(w9)*p(3);
d(4)=round(w10)*p(4);
d(5)=round(w11)*p(5);
d(6)=round(w12)*p(6);
d(7)=round(w13)*p(7);
d(8)=round(w14)*p(8);

m(6)=max(d);

 d(1)=round(w13)*p(1);
d(2)=round(w14)*p(2);
d(3)=round(w15)*p(3);
d(4)=round(w16)*p(4);
d(5)=round(w9)*p(5);
d(6)=round(w10)*p(6);
d(7)=round(w11)*p(7);
d(8)=round(w12)*p(8);

m(7)=max(d);

d(1)=round(w11)*p(1);
d(2)=round(w12)*p(2);
d(3)=round(w13)*p(3);
d(4)=round(w14)*p(4);
d(5)=round(w15)*p(5);
d(6)=round(w16)*p(6);
d(7)=round(w9)*p(7);
d(8)=round(w10)*p(8);

m(8)=max(d);
c=min(m);
```

layer1_2_Net2.m

```
function
c=layer1_2_Net2(a,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16)
p(1)=(a(2,2)-a(1,1)+256-1)/(2*(256-1));
p(2)=(a(2,2)-a(1,2)+256-1)/(2*(256-1));
p(3)=(a(2,2)-a(1,3)+256-1)/(2*(256-1));
p(4)=(a(2,2)-a(2,3)+256-1)/(2*(256-1));
p(5)=(a(2,2)-a(3,3)+256-1)/(2*(256-1));
p(6)=(a(2,2)-a(3,2)+256-1)/(2*(256-1));
p(7)=(a(2,2)-a(3,1)+256-1)/(2*(256-1));
p(8)=(a(2,2)-a(2,1)+256-1)/(2*(256-1));
```

```
d(1)=round(w1)*p(1);
d(2)=round(w2)*p(2);
d(3)=round(w3)*p(3);
d(4)=round(w4)*p(4);
d(5)=round(w5)*p(5);
d(6)=round(w6)*p(6);
d(7)=round(w7)*p(7);
d(8)=round(w8)*p(8);

m(1)=max(d);

 d(1)=round(w7)*p(1);
d(2)=round(w8)*p(2);
d(3)=round(w1)*p(3);
d(4)=round(w2)*p(4);
d(5)=round(w3)*p(5);
d(6)=round(w4)*p(6);
d(7)=round(w5)*p(7);
d(8)=round(w6)*p(8);

m(2)=max(d);

 d(1)=round(w5)*p(1);
d(2)=round(w6)*p(2);
d(3)=round(w7)*p(3);
d(4)=round(w8)*p(4);
d(5)=round(w1)*p(5);
d(6)=round(w2)*p(6);
d(7)=round(w3)*p(7);
d(8)=round(w4)*p(8);

m(3)=max(d);

d(1)=round(w3)*p(1);
d(2)=round(w4)*p(2);
d(3)=round(w5)*p(3);
d(4)=round(w6)*p(4);
d(5)=round(w7)*p(5);
d(6)=round(w8)*p(6);
d(7)=round(w1)*p(7);
d(8)=round(w2)*p(8);

m(4)=max(d);

d(1)=round(w9)*p(1);
d(2)=round(w10)*p(2);
d(3)=round(w11)*p(3);
d(4)=round(w12)*p(4);
d(5)=round(w13)*p(5);
```

```
d(6)=round(w14)*p(6);
d(7)=round(w15)*p(7);
d(8)=round(w16)*p(8);

m(5)=max(d);

 d(1)=round(w15)*p(1);
d(2)=round(w16)*p(2);
d(3)=round(w9)*p(3);
d(4)=round(w10)*p(4);
d(5)=round(w11)*p(5);
d(6)=round(w12)*p(6);
d(7)=round(w13)*p(7);
d(8)=round(w14)*p(8);

m(6)=max(d);

 d(1)=round(w13)*p(1);
d(2)=round(w14)*p(2);
d(3)=round(w15)*p(3);
d(4)=round(w16)*p(4);
d(5)=round(w9)*p(5);
d(6)=round(w10)*p(6);
d(7)=round(w11)*p(7);
d(8)=round(w12)*p(8);

m(7)=max(d);

d(1)=round(w11)*p(1);
d(2)=round(w12)*p(2);
d(3)=round(w13)*p(3);
d(4)=round(w14)*p(4);
d(5)=round(w15)*p(5);
d(6)=round(w16)*p(6);
d(7)=round(w9)*p(7);
d(8)=round(w10)*p(8);

m(8)=max(d);
c=min(m);
```

layer4_mod.m

```
function [Op4,del_y0,t1]=layer4_mod(O1,O2,I,a1,a2,a3)
L=256;
del_y0=(L-1)*(O1-O2);
sizes=size(O1);
X=zeros(sizes(1),sizes(2));
X1=reshape(X,1,sizes(1)*sizes(2));
I1=reshape(I,1,sizes(1)*sizes(2));
del_y01=reshape(del_y0,1,sizes(1)*sizes(2));
```

```
t1=(L-1)*(m_la(abs(del_y01),I,a1,a2,a3));
for i=1:sizes(1)*sizes(2)
        if del_y01(i)>=0
            X1(i)=I1(i)+t1(i);
        else
            X1(i)=I1(i)-t1(i);
        end
end
Op4=reshape(X1,sizes(1),sizes(2));
end
```

<u>m_la.m</u>

```
function mla=m_la(delY0,I,a1,a2,a3)
L=256;
sizes=size(I);
mla=zeros(sizes(1),sizes(2));
len=sizes(1)*sizes(2);
I=reshape(I,1,len);
mla=reshape(mla,1,len);
mmd=m_md(I,a2,a3);
a0=a1*mmd;

for i=1:len
if delY0(i)<=a0(i)
    mla(i)=0;
elseif delY0(i)<=a1
    mla(i)=(a1*(delY0(i)-a0(i))/((L-1)*(a1-a0(i))));
else
    mla(i)=delY0(i)/(L-1);
end
end
```

<u>m_md.m</u>

```
function mmd=m_md(I,a2,a3)
L=256;
sizes=size(I);
mmd=zeros(1,sizes(2));
if a2<a3
        t=a3;
        a3=a2;
        a2=t;
end
for i=1:sizes(1)*sizes(2)

    if I(i)>(L/2)
        I(i)=L-I(i);
    end
    if I(i)<(L/2)-a2
        mmd(i)=0;
    elseif I(i)<(L/2)-a3
        mmd(i)=(I(i)-L/2+a2)/(a2-a3);
    else
        mmd(i)=1;
    end
end
```

<u>main1.m</u>

```
I=imread('pout.tif');
I=imresize(I,[256 256]);
figure, imshow(I);
I=imnoise(I,'salt & pepper',0.1);
figure, imshow(I);
I=double(I);
sizes=size(I);
w1=F(1);
w2=F(2);
w3=F(3);
w4=F(4);
w5=F(5);
w6=F(6);
w7=F(7);
w8=F(8);
w9=F(9);
w10=F(10);
w11=F(11);
w12=F(12);
w13=F(13);
w14=F(14);
w15=F(15);
w16=F(16);
a1=F(17);
a2=F(18);
a3=F(19);
Run_layer1_2_Net1;
Run_layer1_2_Net2;
 [y,del_y0,t1]=(layer4_mod(O1,O2,I,a1,a2,a3));
y1=reshape(y,sizes(1),sizes(2));
figure,imshow(y1,[]);
```

Matlab functions used:

[Platform: Matlab 2015a
Version: MATLAB 8.5]

1. imread:
   Read image from graphics file.
   Syntax:
   A=imread(filename)
   Description:
   A=imread(filename) reads the image from the file specified by filename, inferring the format of the file from its contents. If filename is a multi-image file, then imread reads the first image in the file.

2. imresize:
   Resize image.
   Syntax:
   B = imresize(A,scale)
   Description:
   B =imresize(A,scale) returns image B that is scale times the size of A. The input image A can be a grayscale, RGB, or binary image. If A has more than two dimensions, imresize only resizes the first two dimensions. If scale is in the range [0, 1], B is smaller than A. If scale is greater than 1, B is larger than A. By default, imresize uses bicubic interpolation.

3. imnoise:
   Add noise to image.
   Syntax:
   J = imnoise(I, 'salt & pepper', d)
   Description:
   J = imnoise(I, 'salt & pepper', d) adds salt and pepper noise, where d is the noise density. This affects approximately d*numel(I) pixels.

4. imshow:
   Display image
   Syntax:
   imshow(I)
   Description:
   imshow(I) displays the grayscale image I in a figure. imshow optimizes figure, axes, and image object properties for image display

5. <u>reshape</u>:
   reshape array
   <u>Syntax</u>:
   B = reshape(A,sz1,...,szN)
   <u>Description</u>:
   B = reshape(A,Sz) reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).

6. <u>rand</u>:
   Uniformly distributed random numbers.
   <u>Syntax</u>:
   X = rand(sz1,...,szN)
   <u>Description</u>:
   X = rand(sz1,...,szN) returns an sz1-by-...-by-szN array of random numbers where sz1,...,szN indicate the size of each dimension. For example, rand(3,4) returns a 3-by-4 matrix.

7. <u>plot</u>:
   2-D line plot.
   <u>Syntax</u>:
   plot(X,Y)
   <u>Description</u>:
   plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

- If X and Y are both vectors, then they must have equal length. The plot function plots Y versus X.
- If X and Y are both matrices, then they must have equal size. The plot function plots columns of Y versus columns of X.
- If one of X or Y is a vector and the other is a matrix, then the matrix must have dimensions such that one of its dimensions equals the vector length. If the number of matrix rows equals the vector length, then the plot function plots each matrix column versus the vector. If the number of matrix columns equals the vector length, then the function plots each matrix row versus the vector. If the matrix is square, then the function plots each column versus the vector.
- If one of X or Y is a scalar and the other is either a scalar or a vector, then the plot function plots discrete points. However, to see the points you must specify a marker symbol, for example, plot(X, Y, 'o').

8. <u>size</u>:
   Array size.
   <u>Syntax</u>:

sz = size(A)

Description:

sz = size(A) returns a row vector whose elements contain the length of the corresponding dimension of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4].

9. zeros:

   Create array of all zeros.

   Syntax:

   X = zeros(n)

   Description:

   X = zeros(n) returns an n-by-n matrix of zeros.

10. ones:

    Create array of all ones.

    Syntax:

    X = ones(n)

    Description:

    X = ones(n) returns an n-by-n matrix of ones.

11. min:

    Minimum elements of an array.

    Syntax:

    M = min(A)

    Description:

    M = min(A) returns the minimum elements of an array.

- If A is a vector, then min(A) returns the minimum of A.
- If A is a matrix, then min(A) is a row vector containing the minimum value of each column.
- If A is a multidimensional array, then min(A) operates along the first array dimension whose size does not equal 1, treating the elements as vectors. The size of this dimension becomes 1 while the sizes of all other dimensions remain the same. If A is an empty array with first dimension 0, then min(A) returns an empty array with the same size as A.

12. max

    Maximum elements of an array.

    Syntax:

    M = max(A)

    Description: M = max(A) returns the maximum elements of an array.

- If A is a vector, then max(A) returns the maximum of A.
- If A is a matrix, then max(A) is a row vector containing the maximum value of each column.
- If A is a multidimensional array, then max(A) operates along the first array dimension whose size does not equal 1, treating the elements as vectors. The size of this dimension

becomes 1 while the sizes of all other dimensions remain the same. If A is an empty array whose first dimension has zero length, then max(A) returns an empty array with the same size as A.

13. abs:
Syntax:
Y = abs(X)
Description:
Y=abs(X) returns the absolute value of each element in array X. If X is complex, abs(X) returns the complex magnitude.