The aim of the project is to build a model that is capable of identifying the class of a given animal. The interesting part is that we have to build the model without using neural network. Neural network has several hidden layers and with the non-linear activation functions, it can learn almost anything provided we supply suitable training dataset to start with.
In this project we focus on iterative method based approach to identify animals. We have developed a prototype for the same. Our aim is to construct the model without using any machine learning based inbuilt library (like sklearn etc). The algorithm used is genetic algorithm.

1 Overview of the code

There are 3 python files :

- **Parameters.py** : It contains all the parameters required for the execution of the Genetic Algorithm (GA). It creates a folder with the name of current date and time **(d-m-y_h-m-s)** in the directory **"Results"**. All the results will be stored in that folder.
  The following variables are defined in this file:
  - save_path : The path of the folder "Project" in the system
  - Generations : Number of iterations up to which the GA should be executed
  - times : To calculate the Population size.
    Population size is calculated as
    Population size = times $\times$ Number of animals in the dataset
    Currently, there are 6 classes of animals in the dataset.
    So, Population size = 6 $\times$ times
  - Cross_over_prob : Probability of crossing over
  - Mutation_prob : Probability of mutation
- **Cost_calculation.py** : It contains the functions to calculate the fitness function of the population, cross over and mutation in the population.
- **Optimization.py** : It has a code which calls the parameters from the "Parameters.py" and functions from the "Cost_calculation.py" and does the optimization. It outputs the files which are used for the post-optimization analysis.
  It gives the following outputs -

– A pdf file having the plot of maximum fitness vs generations, fitness vs population size, and test image.

– A csv file having the data of maximum fitness in each generation.

– Bunch of csv files having the data of fitness for each individual in the population at the generation gap of 10.

– A csv file that gives the information of probability of predicted class labels.

2 Materials and Methods

The entire procedure is divided roughly into three steps :

1. Pre-optimization processing
2. Optimization step
3. Post-optimization analysis

2.1 Pre-optimization methods

This step involves preparing the dataset of animals with labels, performing operations and transformations on the collected images and converting them to grey-scale.

2.1.1 Preparing the dataset

The dataset is taken from **Kaggle**. Currently, 6 classes of animals are being used in classification.

Class	Number of images taken
Elephant	371
Fox	245
Horse	372
Lion	286
Squirrel	399
Tiger	265

2.1.2 Cropping the image

Here, we are using Fourier transform of the intensity of the images along each row and column to crop the collected images so as to remove the background.

An image is the spatial distribution of intensity of light. This spatial locations are not continuous but are discrete and known as **pixels**.

In Fourier domain, we go from spatial domain to frequency domain.

So, basically we assume that background has somewhat lesser intensity then those pixels which convey the information of body of the animal. There is less intensity variation within the background pixels. When we go from background to the body of the animal, the intensity increases suddenly due to the contrast in the image.

The intensity profile of background and the pixels carrying the information of the animal is shown in Fig-1. We can see that there is little fluctuation in the the intensity profile of background. In Fig-1 (b), it can easily be seen that the intensity changes dramatically around the edge of the animal shape.

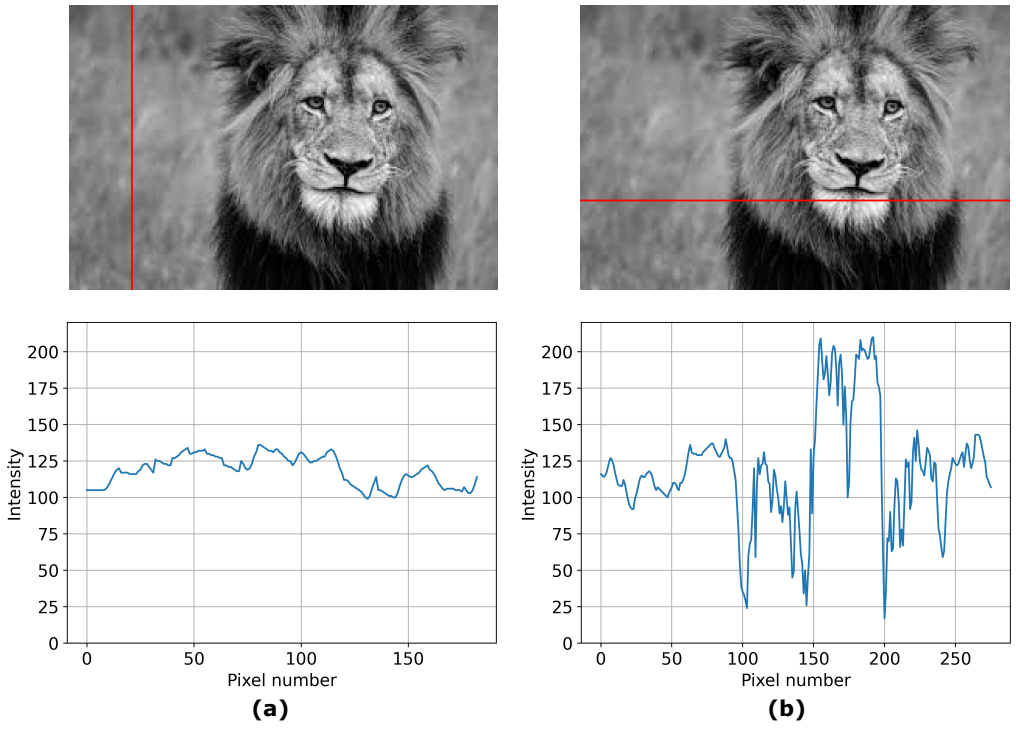


Fig-1 Intensity profile about the (a) vertical and (b) horizontal line

We employ this property to crop the image.

By Fourier series expansion, we know that any function $f(x)$ can be written as the linear combination of sines and cosines or the complex exponentials.

$f(x) = \sum_k c_k e^{ikx}$

k : the spatial frequency

$c_k s$: the amplitude (coefficient) corresponding to the spatial frequency k.

Basically, we are trying to fit the function f(x) by the superposition of sine and cosine functions of various frequencies. The contribution of various frequencies to the linear combination is the amplitude of that frequency in the Fourier domain.

In Fig-1(a), we can see that there is very little fluctuation of the intensity across pixels. So, low frequency sinusoidal functions are enough to represent the intensity profile. Higher frequencies have very little contribution.

On the contrary, in Fig-1(b), the intensity profile changes very rapidly and by larger amount, so higher frequencies have significant contribution to the Fourier spectrum.

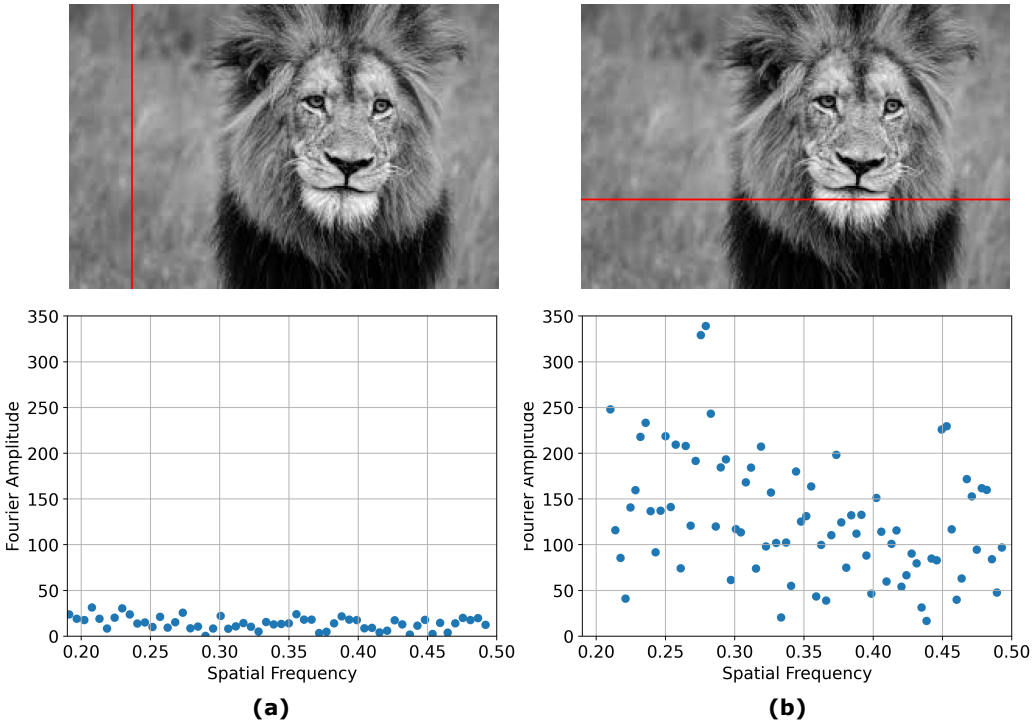


Fig-2 Fourier Spectrum of the intensity about (a) vertical and (b) horizontal line, showing the amplitude for higher frequencies

So, we are scanning through each row and columns and retaining only those rows and columns in

which higher frequencies have significant amount of contribution and rejecting the others. In this way, we are able to crop the image.

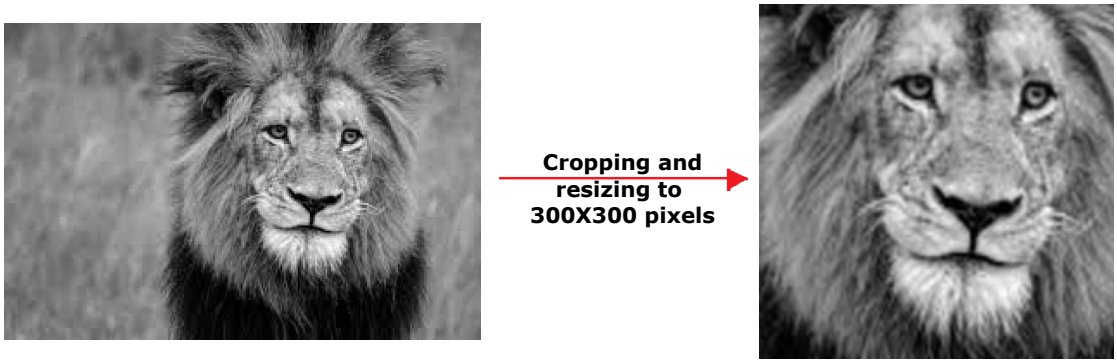


Fig-3 Cropping of image by selecting the rows and columns having higher frequency components in Fourier spectrum

The method which we are using for cropping suffers from limitations. It assumes that background is constant and the contrast in image is good. If the background also has some structures like houses, people, etc, then it won't be able to remove that. Because they also contribute to the higher frequencies in Fourier spectrum.

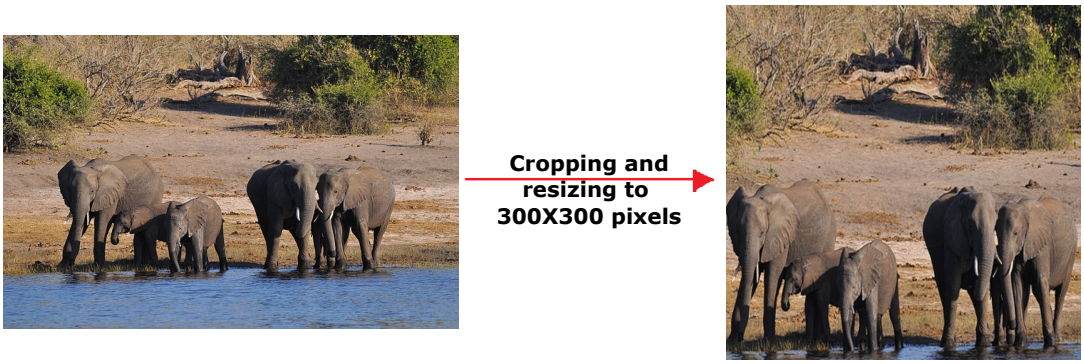


Fig-4 The background of the image is not uniform so it remains in the cropped image

2.2 Preparing the dataset

So after cropping and converting each image to grey-scale, we have resized the images to same size (300,300). We have converted the images of each class of animals into numpy array of size (300,300,Number of pics in that class) and save it in .npy format in the folder "Data".
Ex - In **Elephant.npy**, Elephant$[:,:,5]$ is the (300×300) array for the 6th image of the Elephant. As we have to predict the class of animals, it might happen that the test image is close to the mirror

image of one of the images in our dataset. So while creating array of images we are also adding the mirror image of the existing dataset images

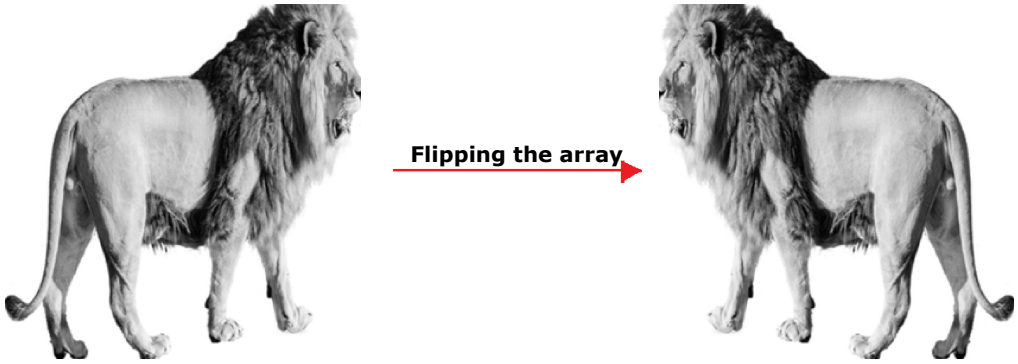


Fig-5 Creating mirror image of the existing image by flipping the corresponding array from right to left.

The content of folder "data" is shown below -

Name	Shape of array
Elephant.npy	(300,300,742)
Fox.npy	(300,300,490)
Horse.npy	(300,300,744)
Lion.npy	(300,300,572)
Squirrel.npy	(300,300,798)
Tiger.npy	(300,300,530)

2.3 Optimization step

2.3.1 Representing images as chromosome

Each image of a particular class is represented using a column vector as explained below.

Class	Label given
Elephant	0
Fox	1
Horse	2
Lion	3
Squirrel	4
Tiger	5

Fig-4 Representing each image of a particular class as a chromosome with each entries be regarded as a gene.

2.3.2 Genetic Algorithm

It is an optimization method developed by taking inspiration from the evolutionary pathway in the nature. The nature works on the simple rule **"Survival of the fittest"**. Those individuals which can cope with the changing environmental conditions gets survived and can pass their genes to the next generations which carry the information of fitness with them. In this way, their traits are being selected by nature, and pass onto the next generations.
Total number of generations = M

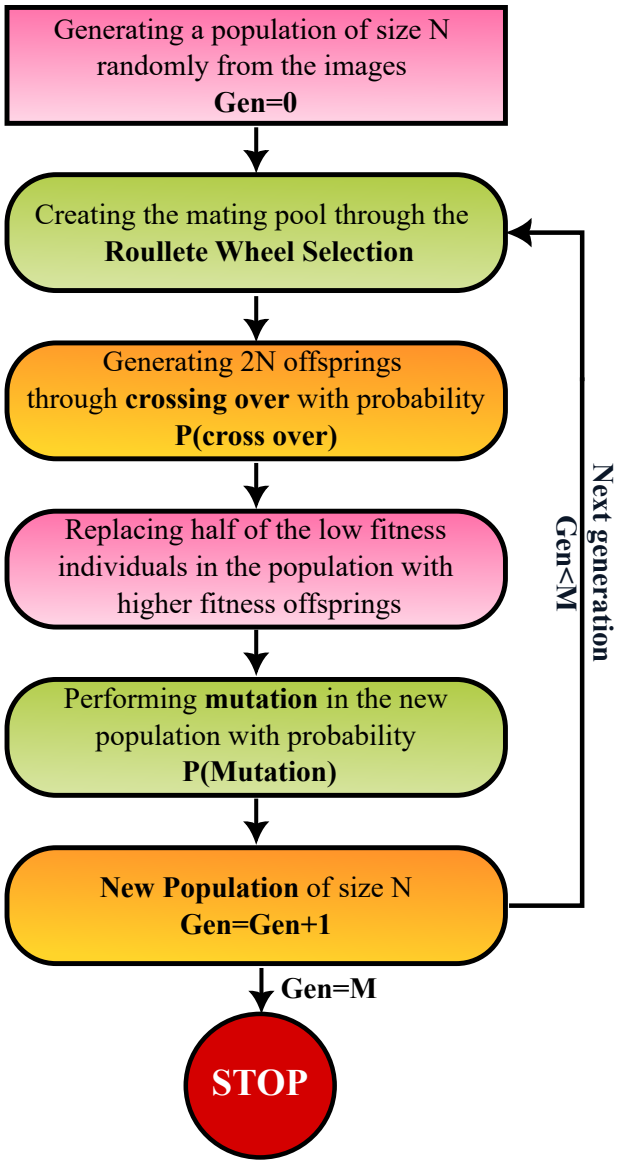


Fig-6 Schematic of the steps followed in the Genetic algorithm.

1. **Population Size**

   Initially we have to generate individuals randomly to create the population.

   Total number of classes of animals are 6.

   We have defined a new variable **"times"** such that the Population size = times×Number of class

   We have set times = 100

   Population size = $100 \times 6 = 600$

   Initially 100 random images is taken from each class to create the population.

2. **Crossing over** $P(\text{cross over}) = 0.6$

As shown in Fig-4, there are two genes associated with each chromosome.

If cross over happens, either of the two genes can be exchanged.

So we generated a random number 0 or 1 with each having probability 0.5. If the random number is 0, then the bottom gene will be exchanged otherwise the top gene. We have associated a column vector with each of these random number. This is explained below.

Random number : 0

Column vector : $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Parent - 1 : $\begin{pmatrix} a \\ b \end{pmatrix}$ Parent - 2 : $\begin{pmatrix} c \\ d \end{pmatrix}$

Parent : $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$

Offspring[2nd row of column vector]=Offspring[1] = Second row of Offspring

Offspring[2nd row of column vector] = Transpose$\Big(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\big(\text{Transpose(Parent).Column vector}\big)\Big)$

$\implies$ Offspring[first row] $= \Big(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}\Big)^T = \begin{pmatrix} d & b \end{pmatrix}$

Offspring[1st row of column vector]=Offspring[0] = First row of Offspring

Offspring[1st row of column vector] = do nothing = $\begin{pmatrix} a & c \end{pmatrix}$

Offspring = $\begin{pmatrix} \text{First row} \\ \text{Second row} \end{pmatrix} = \begin{pmatrix} a & c \\ d & b \end{pmatrix}$

$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \rightarrow \begin{pmatrix} a & c \\ d & b \end{pmatrix}$

Random number : 1

Column vector : $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Parent - 1 : $\begin{pmatrix} a \\ b \end{pmatrix}$ Parent - 2 : $\begin{pmatrix} c \\ d \end{pmatrix}$

Parent : $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$

Offspring[2nd row of column vector]=Offspring[0] = First row of Offspring

Offspring[2nd row of column vector] = Transpose$\Big(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\big(\text{Transpose(Parent).Column vector}\big)\Big)$

$\implies$ Offspring[first row] = $\Big(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\Big)^T = \begin{pmatrix} c & a \end{pmatrix}$

Offspring[1st row of column vector]=Offspring[1] = Second row of Offspring

Offspring[1st row of column vector] = do nothing = $\begin{pmatrix} b & d \end{pmatrix}$

Offspring = $\begin{pmatrix} \text{First row} \\ \text{Second row} \end{pmatrix} = \begin{pmatrix} c & a \\ b & d \end{pmatrix}$

$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \rightarrow \begin{pmatrix} c & a \\ b & d \end{pmatrix}$

Caution : In our dataset, different classes of animals have different number of images. It might happen that after crossing over, the number in the second row exceeds the total number of images available for the class corresponding to first row. So to avoid this error, we are also performing "modulo" operation in crossing over.
Ex - If after crossing over, second row has number 550 while the total number of available images corresponding to that class is just 300. Then 550 modulo 300 $=$ 250 $<$ 300.

3. **Mutation**
   Probability(Mutation) = 0.05
   We are generating a random 0 or 1 for each individual in the population depending on the probability of mutation.

4. **Fitness Function** The Fitness function used here is **Pearson Correlation (PC)**.
   We have each image which is basically a matrix of size (300×300). During fitness calculation, we are making it a row vector (1×90000). The range of PC lies between -1 to +1.
   Dataset and test images are represented as D and T respectively.

$$PC(D,T) = \frac{\sum_{i=1}^{90000}(D_i - \overline{D})(T_i - \overline{T})}{\sum_{i=1}^{90000}(\sqrt{(D_i - \overline{D})^2})\sum_{i=1}^{90000}(\sqrt{(T_i - \overline{T})^2})}$$

$\overline{D}$ = Mean of pixel intensities of D

$\overline{T}$ = Mean of pixel intensities of T

$$\boxed{PC = \frac{Covariance(T, D)}{\sqrt{Variance(T)}\sqrt{Variance(D)}}}$$

2.4 Post-Optimization Analysis

2.4.1 Analysing the mutation rate on the prediction

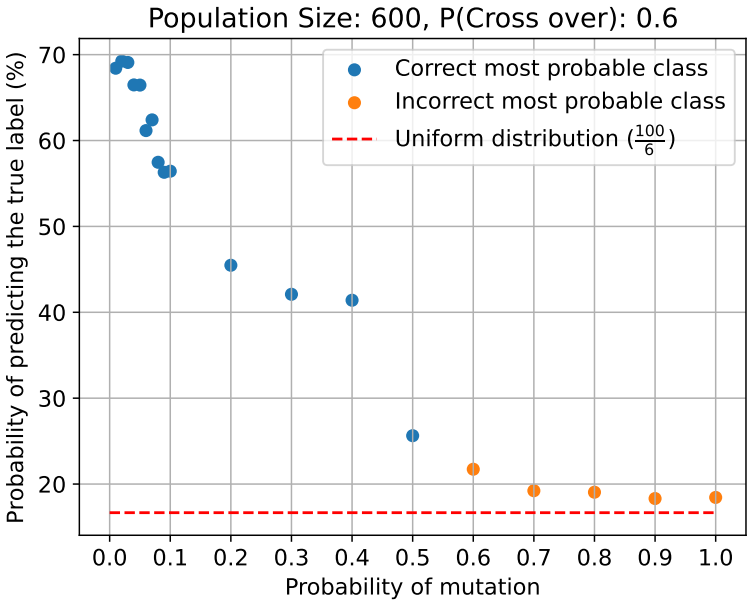
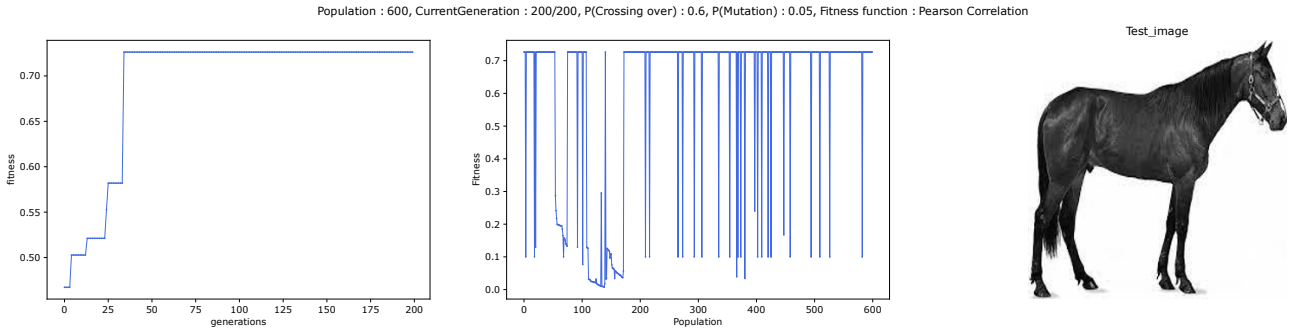


Fig-7 Schematic of the steps followed in the Genetic algorithm.

So, from the above graph we get a rough idea that mutation probability between between 0.01 and 0.1 can prove to be good.

2.4.2 Output from the code



(a)

	A	B	C	D
1	Animals	Probability of prediction (%)		
2	Fox	3.827007		
3	Elephant	6.526136		
4	Tiger	6.699855		
5	Lion	7.278054		
6	Squirrel	9.214893		
7	Horse	66.45406		
8				

(b)

Fig-8 (a) Output from the pdf file : First plot is Fitness vs Generation, Second plot is for Fitness within the population at the present generation, Third image is the test image (b) Output from the csv file showing prediction probability

2.4.3 Efficiency of the algorithm

We have calculated efficiency of the algorithm for the two values of mutation probability 0.05 and 0.1.

We are using the following formula for calculating the efficiency.

$$\text{Efficiency} = \frac{\text{Number of correct most probable class}}{\text{Number of test data images}}$$

Mutation Probability = 0.05

Number of correct classifications = 11

Total number of test images = 30

Efficiency = $\frac{11}{30} = 0.367$

Efficiency = 36.7%

Mutation Probability = 0.1

Number of correct classifications = 18

Total number of test images = 30

Efficiency = $\frac{18}{30} = 0.64$

Efficiency = 60.0%

3 Significance

With GA, we can do the real time optimization. We can also use the same model to recognise handwritten digits and other classification problems. We can integrate computer vision and other image processing techniques for tracking animal behaviour for scientific research or surveillance.

4 Conclusion

We have developed a prototype for the animal detection using GA. We have analysed the effect of mutation probability on the efficiency of the algorithm. The efficiency of the algorithm can be improved further so that the fitness function does not saturate quickly. Implementing image filters and convolutions which can perform edge detection can prove to be good.