Effectively Diagnosing Malaria by Optimizing the Hyperparameters of CNN using Genetic Algorithm on the Multi core GPU

Manjit Jaiswal, Aditya Sahu, Md Tausif Zafar

***Abstract**: Image classification is an important task in computer vision involving a large area of applications such as object detection, localization and image segmentation. When it comes to image classification, the most adopted methods are based on deep neural network and especially convolutional Neural Networks(CNN). Selection of hyperparameters plays a crucial role in performance of model and it comes by experience. So, in this paper, we will use the genetic algorithm(GA) to automate and build the CNN model for higher accuracy on GPU which is provided by Google Collaboratory cloud. The best architecture of CNN after several generations of the genetic algorithm is then compared to the state-of-the-art CNN. We have used the malaria cell images dataset to find out whether the person is normal or if they are suffering from malaria. We trained two types of malaria cells, which are uninfected and parasitized on Tesla P100 multi core GPU. We got a high training accuracy of 97% and got a testing accuracy of about 95% on the multicore GPU that boosted the speed of execution of training time period and testing time period.*

***Keywords**: CNN, genetic algorithm, GPU, Image classification, Neural Network Architecture optimization, Tesla, NVIDIA.*

I. INTRODUCTION

Image Classification using Convolutional Neural Networks (a branch of deep learning) is very popular and efficient. It has a very impressive record in classifying images. The Convolutional Neural Networks have gathered a lot of attention in recent years due to its performance in various visual recognition problems. Convolutional Neural Network (CNN) is a term used to describe an architecture based on spatially localized neural input for applying neural networks to two-dimensional arrays (usually images) [1]. CNN is used even in manufacturing semi-conductors, for the detection of faults and their classification.

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.

* Correspondence Author
Manjit Jaiswal*, Assistant professor, Department Computer Science and engineering, Institute of technology, Guru Ghasidas Vishwavidyalaya, Bilaspur.
Aditya Sahu, Bachelor of Technology, Department of Computer Science and Engineering from School of Studies in Engineering and Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India
Md Tausif Zafar, Bachelor of Technology,Department of Computer Science and Engineering, School of Studies in Engineering and Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur,

Here, we have used CNN for effectively diagnosing malaria**.** Not only we have used it to diagnose malaria, but we have also tried to increase the classification accuracy of CNN alone by combining it with the genetic algorithm. We have implemented our model in the GPU provided by google. GPU is highly anticipated in the medical field on big data like DNA alignment, image processing, image classification, social network recommendations, etc. These applications involve heavy computations on large datasets. To run the training process for such large datasets in a feasible time, parallel computing is considered and the graphics processing unit (GPU) are massively parallel devices candidates to perform such a parallel task [2].

A. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) all in all are an assortment of trainable mathematical units (neurons), that collectively learn non-linear functions. Practically they are used in various manufacturing application domains such as process planning or production control [3]. The use of artificial neural network had been proven a very effective method for pattern recognition and so they are also useful for diagnosis of cancer at early stages [4].

B. Convolutional Neural Network (CNN)

Unlike fully connected ANN, where each layer's (L) neuron is connected to all the neurons in the previous layer (L-1), CNN is a specific type of feed-forward ANN and neurons are connected locally. CNNs are used to recognize visual patterns directly from pixel images with variability. The architecture is composed of multiple feature extraction stages. Each stage consists mainly of three basic operations: convolution, non-linear neuron activation, and feature pooling. First of all, in order to get an output, an image is taken and made to pass over a series of convolutional, nonlinear(activation), pooling, and fully connected layers [5]. This output which we will get can describe a probability of classes or a single class. Here we have used the combination of genetic algorithm and CNN to diagnose malaria from the given cell images. By considering a cell image, we can predict whether a cell is parasitized or it is uninfected by malaria.

II. LITERATURE REVIEW

By using genetic algorithms (GA), they aimed to enhance the accuracy of artificial neural networks (ANNs) by finding out the optimal set of initial weights [6]. The purpose is to find the optimal initial weights of neural networks via genetic algorithm to improve predictability.

Here, the use of simple GA is proved to be effective for improving the accuracy of artificial neural networks.

Genetic algorithm (GA) and deep convolutional neural networks (CNN) were used for human action recognition [7]. The classification error is demonstrated to be minimized by initializing the weights of a convolutional neural network (CNN) classifier based on solutions generated by genetic algorithm (GA).

To find a solution that is closer to global-optimum, the global search capabilities of genetic algorithms and the local search ability of the gradient descent algorithm are utilized. A method was proposed for classifying different grades of glioma brain tumor MR images [8].

Instead of using existing methods of selecting a deep neural network architecture normally based on trial and error, or by following some specific structures which are predefined, here CNN's architecture (structure) is built using GA. Genetic algorithm is utilized to search for a CNN structure that produces better results. They got 90.9 % accuracy for classification in one case study and 94.2% in another. A system was proposed where an original MRI image is used for training a classification network [9]. To classify the image as a tumorous and non-tumorous sample, CNN is used. For segmentation, the watershed algorithm is used. In the existing system, segmentation was done manually and data was used for training and testing. Then, CNN was used for comparing the given image with the dataset. The proposed system improves the accuracy to 89% on 2D brain tumor classification from MRI images. A web based handwritten digit recognition system was developed by using the convolutional neural network [10]. MNIST handwritten database was used for training and testing. The comparison based on accuracy and time spent is done between CPU and GPU on both Neural Network(NN) and CNN architectures. A processing based approach was presented for the corrosion grade.

III. METHODOLOGY

Convolutional Neural Network (CNN)

The Convolutional Neural Network comprises of a sequence of layers. The main use of CNN is to recognize patterns within images. There are 4 main layers which form the basic building block of every CNN as shown in Fig. 1.

A. Convolution Layer

The first layer of CNN architecture is always a Convolution layer. It can be called the building block of any convolutional network. Here, we aim at extracting features from the image. Three-dimensional object (e.g. N x N x D) is the input of this layer here. To clearly understand this process, we can assume that we take a window, of size k x k x D which is much smaller in size when compared to the dimensions of the actual object. This window can be called as a kernel or filter. To calculate the output, after the process of convolution is carried out at one location, we multiply the numbers on the filter and the input image elementwise. The results which we get after multiplication across the height, width, and depth are summed up in order to get a single value [5]. Then, the average value is found out by averaging that single value over the total number of values present in the filter. Now, this filter is moved across all the areas of the input. We start from the top left corner of the given input, and each filter is moved from left to right one step at a time until it reaches the top-right corner. Then the filter is moved one step in the downward direction and again it is moved from left to right until the filter reaches the bottom right corner. Each step is called stride. A number is produced by every unique location on the input object. We get a 2-D array after we complete sliding the filter over all the locations on the object, and this 2-D array is called a feature map.segmentation of corrosion in different grades utilizing unsupervised learning [11]. The CNN outperformed the preceding corrosion detection approach where wavelet characteristics were implemented. The proposed approach resulted in higher accuracy for the identification of

Activation Function Layer

The Activation function is applied to all the values in the filtered image to the output of the convolution layer in order

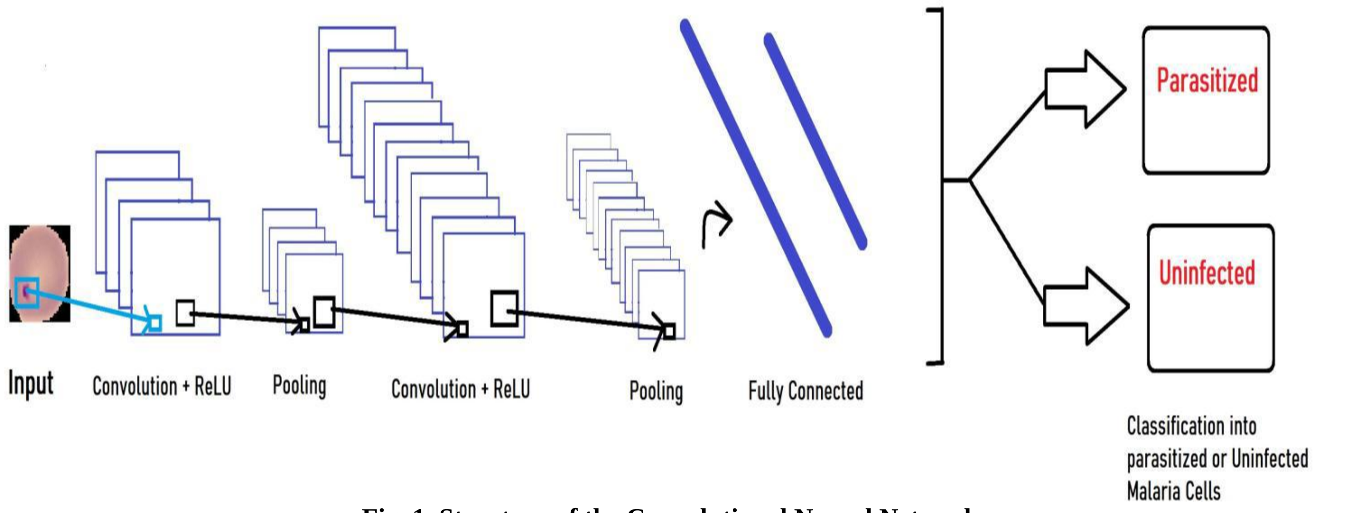


Fig. 1. Structure of the Convolutional Neural Network

to extract the nonlinear features. Some common activation functions used are Sigmoid, ReLU: max(0, x), tanh etc.

Pooling Layer

Pooling is used to reduce or shrink down the size of the image while preserving the most important characteristics in them. Therefore, it is also known as down sampling or
sub sampling. Many times, pooling layers are periodically inserted in-between successive convolutional layers in CNN architecture. Two of the most common types of pooling layers are max pooling and average pooling. As the name suggests, max-pooling consists of moving a small filter across the image and taking the maximum value of filter at each step. In the same way, average pooling involves calculating the average as the filter moves across the image. Pooling has various advantages associated with it. It makes computation faster and reduces memory also. Since, it reduces the number of parameters or weights in the network, it also prevents overfitting.

D. Fully Connected Layer

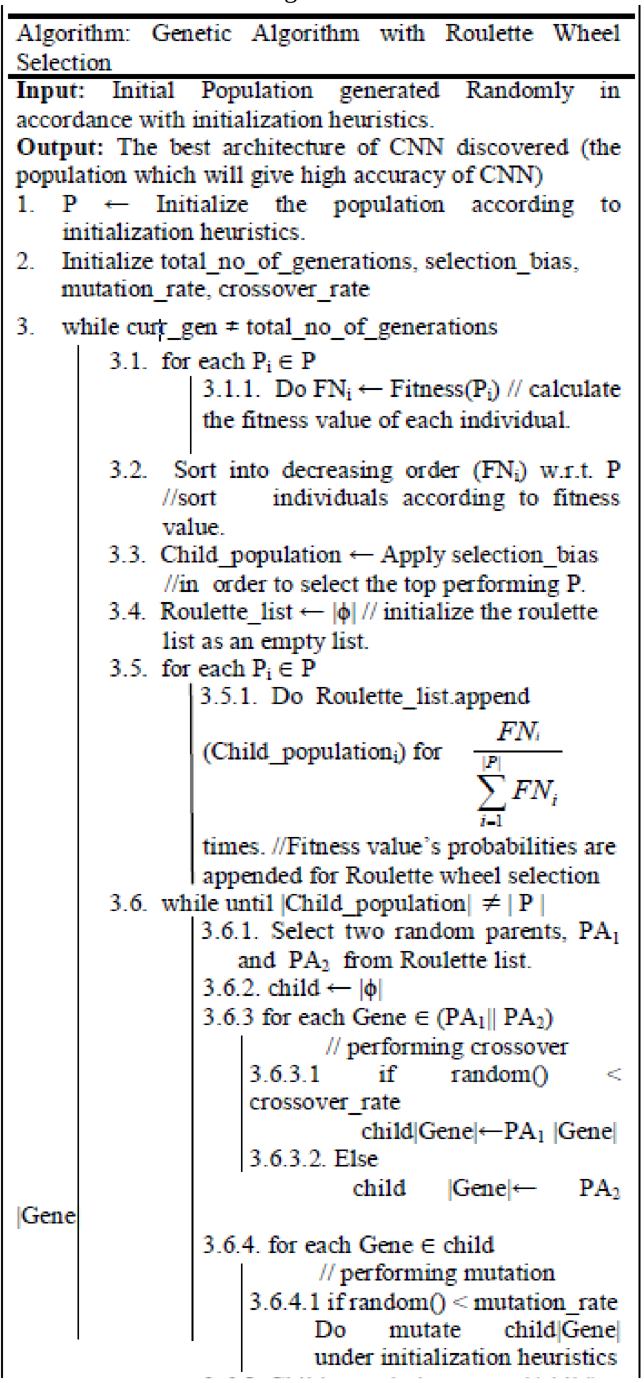
These layers are often used as final layers of the Convolutional Neural Network. These are regular traditional deep neural network layers which take input from the previous layers of the CNN. These are used to do classification based on the input provided by the previous layers. Typically, they may contain a softmax activation function, which outputs a probability (0-1) for the classes which are being predicted by the model.

2. Genetic Algorithm (GA)

Genetic algorithm is a heuristic solution-search or optimization technique that is inspired by Charles Darwin's theory of natural evolution and principles of natural selection where only the fittest individuals are selected out of a population to reproduce and produce offspring for the next generation. It is generally used to find optimal solutions to difficult problems. GA is particularly useful for complex optimization problems, where there is a large number of parameters and the analytical solutions are difficult to get [12]. In GA, we have a population of possible solutions to the given problem. Each individual in the population represents a possible solution to a given problem. The genetic algorithm (GA) transforms a population of individual objects, each with an associated fitness value, into a new generation of the population and these solutions then undergo recombination and mutation (as in natural genetics), producing new children [13]. The process is repeated over various generations. We keep "evolving" better individuals or solutions over generations, until we reach a stopping criterion. In genetic algorithms, a potential solution to a particular problem is encoded on a simple chromosome-like data structure. Then various recombination operators are then applied to these structures in such a way as to preserve basic information [14]. When Crossover is done between two individuals, it produces two new individuals. In case if two individuals are binary coded, some bits of the two parents are exchanged. The mutation is performed for a few offspring: for such offspring, one variable is altered by a small perturbation, for instance, the change of one bit in the binary coding case [15].
Five phases are considered in a genetic algorithm.

1. Initial population generation
2. Fitness evaluation
3. Roulette wheel selection
4. Crossover
5. Mutation

In the genetic algorithm, we need to select chromosomes from the population so that they will become parents for crossover. As we need to select the best chromosomes for this purpose, there are various selection methods for selecting the best of the chromosomes. Here, we have used the Roulette wheel selection for this purpose. The algorithm which we have followed is given below.



```
Algorithm: Genetic Algorithm with Roulette Wheel Selection
Input: Initial Population generated Randomly in accordance with initialization heuristics.
Output: The best architecture of CNN discovered (the population which will give high accuracy of CNN)
1. P ← Initialize the population according to initialization heuristics.
2. Initialize total_no_of_generations, selection_bias, mutation_rate, crossover_rate
3. while curr_gen ≠ total_no_of_generations
    3.1. for each $P_i \in P$
        3.1.1. Do $FN_i$ ← Fitness($P_i$) // calculate the fitness value of each individual.
    3.2. Sort into decreasing order ($FN_i$) w.r.t. P //sort individuals according to fitness value.
    3.3. Child_population ← Apply selection_bias //in order to select the top performing P.
    3.4. Roulette_list ← |ϕ| // initialize the roulette list as an empty list.
    3.5. for each $P_i \in P$
        3.5.1. Do Roulette_list.append (Child_population_i) for $\frac{FN_i}{\sum_{i=1}^{|P|} FN_i}$ times. //Fitness value's probabilities are appended for Roulette wheel selection
    3.6. while until |Child_population| ≠ | P |
        3.6.1. Select two random parents, $PA_1$ and $PA_2$ from Roulette list.
        3.6.2. child ← |ϕ|
        3.6.3 for each Gene ∈ ($PA_1$|| $PA_2$)
                // performing crossover
            3.6.3.1 if random() < crossover_rate
                    child|Gene|←$PA_1$ |Gene|
            3.6.3.2. Else
                    child |Gene|← $PA_2$ |Gene|
        3.6.4. for each Gene ∈ child
                // performing mutation
            3.6.4.1 if random() < mutation_rate
                    Do mutate child|Gene| under initialization heuristics
```

Fig. 2. Genetic algorithm with Roulette Wheel selection

```
        3.6.5. Child_population.append(child)
    3.7. P ← Child_population
```

4. **Return** P // Return the population with best fitness value

In the above algorithm, we are using the heuristic approach of population initialization where we are initializing our population by restricting the search space.

We have constrained the model up to three layers and the values of the number of filters, the size of filters are chosen between predefined values and activation function is also restricted. The selection bias denotes the random trade of choosing which top-performing parents are to be chosen for crossover. In the steady-state, we tend to try and retain the fittest individuals within the population. In the roulette wheel selection approach, we selected the fittest individuals (best performing parents) based on the fitness value for each of the parents. Then, the crossover and mutation are carried out in order to get a child population with better fitness.

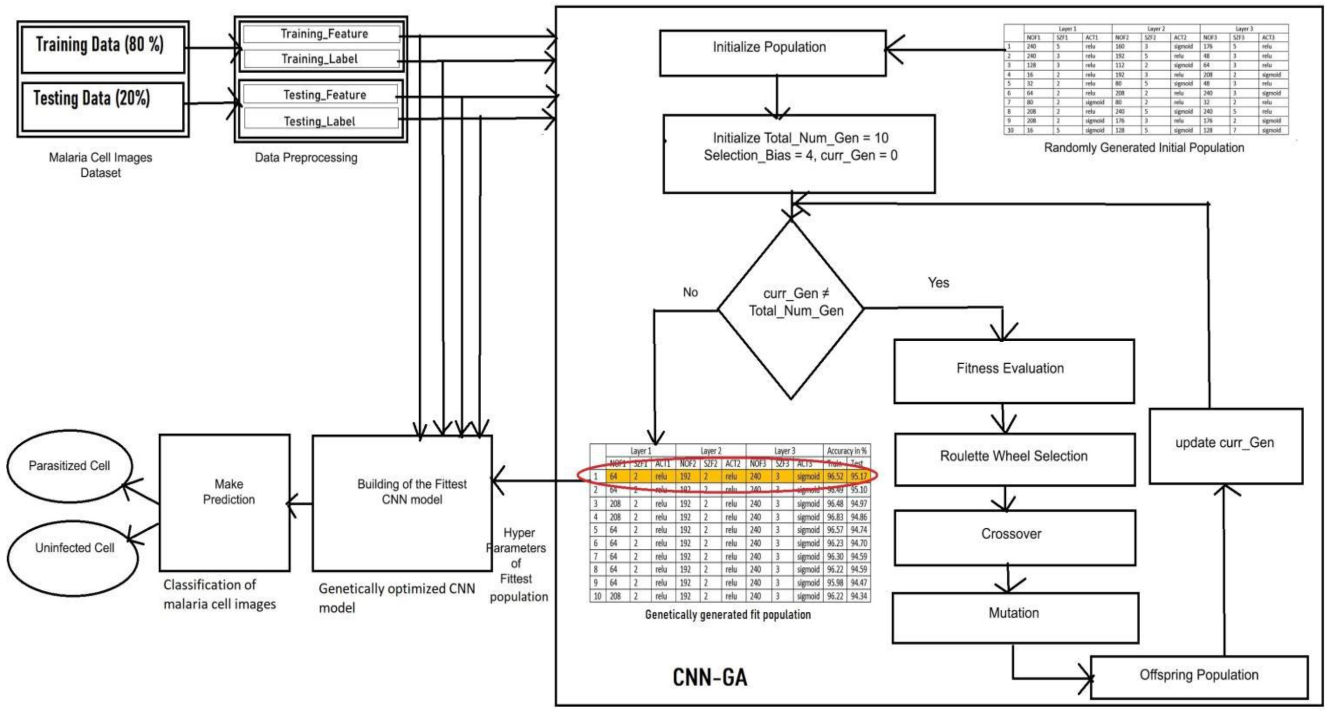
genetic algorithm automatically selects the best set of hyperparamaters which give us high accuracy.

IV. PROPOSED WORK

A. Integration of genetic algorithm with CNN

As in Fig. 3 Integration of the GA with the CNN involves the following steps:

1) Initialization of each chromosome with random values
2) Substituting the weights of the CNN with the values of a selected chromosome.
3) Selecting the parents of the next generation through a roulette process.
4) performing the Crossover between the parents to make new children.
5) Mutating the child generated.
6) Repeating steps 2 to 5 until the evaluation criteria have been met.
7) We get the best set of values of hyperparameters of CNN which will give the highest accuracy.



	Layer 1			Layer 2			Layer 3		
	NOF1	SZF1	ACT1	NOF2	SZF2	ACT2	NOF3	SZF3	ACT3
1	240	5	relu	160	3	sigmoid	176	5	relu
2	240	3	relu	192	5	relu	48	3	relu
3	128	3	relu	112	2	sigmoid	64	3	relu
4	16	2	relu	192	3	relu	208	2	sigmoid
5	32	2	relu	80	5	sigmoid	48	3	relu
6	64	2	relu	208	2	relu	240	3	sigmoid
7	80	2	sigmoid	80	2	relu	32	2	relu
8	208	2	relu	240	5	sigmoid	240	5	relu
9	208	2	sigmoid	176	3	relu	176	2	sigmoid
10	16	5	sigmoid	128	5	sigmoid	128	7	sigmoid

Randomly Generated Initial Population

	Layer 1			Layer 2			Layer 3			Accuracy in %	
	NOF1	SZF1	ACT1	NOF2	SZF2	ACT2	NOF3	SZF3	ACT3	Train	Test
1	64	2	relu	192	2	relu	240	3	sigmoid	96.52	95.17
2	64	2	relu	192	2	relu	240	3	sigmoid	96.49	95.10
3	208	2	relu	192	2	relu	240	3	sigmoid	96.48	94.97
4	208	2	relu	192	2	relu	240	3	sigmoid	96.83	94.86
5	64	2	relu	192	2	relu	240	3	sigmoid	96.57	94.74
6	64	2	relu	192	2	relu	240	3	sigmoid	96.23	94.70
7	64	2	relu	192	2	relu	240	3	sigmoid	96.30	94.59
8	64	2	relu	192	2	relu	240	3	sigmoid	96.22	94.59
9	64	2	relu	192	2	relu	240	3	sigmoid	95.98	94.47
10	208	2	relu	192	2	relu	240	3	sigmoid	96.22	94.34

Genetically generated fit population

Fig. 3. Working process of CNN with GA

We propose a model as in Fig. 3 comprising of both genetic algorithm and CNN that increases the percentage accuracy of the model in order to classify images when it is being trained with CNN. For the accurate design of the state of the art CNNs, a crucial step is the selection of the set of hyperparameters, as it increases the performance of the model. What we have done is, we have automated this process of selection of hyperparameters with the help of the genetic algorithm. The genetic algorithm gives us the best set of hyperparameters in order to give input to the convolutional neural network.

The CNN automated with genetic has several advantages: -

- We get better accuracy for classification as when compared with classification by CNN only.
- An experienced person is not needed for performing the selection of hyperparameters. The

Proposed Model

Here, we have described our network model which shows how our model of CNN and genetic algorithm produces the desired output and helps in classifying the malaria cell to groups of parasitized or uninfected cells. Our malaria cell image dataset is divided into training (80%) and testing data (20%). We changed the dimensions of images to 28 * 28 * 3. The training data was then divided into the training labels and training features. The value of the pixels of the image is stored in training feature and training label. The same process is repeated for testing data. Now, let us understand how our model uses this data. First of all, according to the genetic algorithm, we need an initial population. Therefore, we initialized a random population of chromosomes (which contains a random value of hyperparameters). We set the initial value to the total number of generations to 10 (as we want the best set of offspring population after 10 generations). The generation which is currently running is initialized value of 0 and the selection bias is set to 4. Here, we have assumed that the chromosome is represented as population and hyper-parameter is represented as genes.

Here, our objective is to generate an automatic architecture design algorithm for CNNs for image classification problems by using the GA which is capable of discovering the fittest CNN architecture. We have taken a total of 10 populations in 1 generation and there are 10 generations. We have randomly initialized the population of first generation. Then, as Fig. 3 shows, we have set a loop that checks whether we have reached the last generation or not. Until then, we try to find out the optimized set of hyperparameters which will give us the best accuracy. The first step after the condition is found to be true is the fitness evaluation. In the fitness evaluation, the classification accuracy of testing data on CNN is used as a fitness value. After the evaluation of fitness, Roulette wheel selection is used to find out the parent population which will be used for child population creation. In roulette wheel selection, each population is allocated a probability of being chosen proportional to its relative fitness. Hence, the best fit individuals have a greater chance of being selected than those with lower fitness.

After selection, the crossover is carried out in which the genes from two selected parent populations are mixed. This recombination allows producing one or two child population. To ensure maximum diversity when creating offspring, all resulting population after crossover is then passed to the mutation process. Here, one or more gene values of the individual population are changed.

After mutation, we get a new offspring population which becomes the parent population for the next generation. And the same process is repeated out for the remaining generations. After all the generations are over, we get a population with the best fitness which gives us the highest accuracy. And, with that fittest population i.e. the best set of hyperparameters, we build our model which gives us high accuracy with our training and testing data. Finally, we can now predict from our model whether the given input cell is parasitized or uninfected.

V. IMPLEMENTATION AND EXPERIMENTAL RESULT

A. NIH Malaria Dataset

Malaria is a fatal and life-threatening disease caused by Plasmodium parasitic infection transmitted through the bite of female Anopheles mosquitoes. The parasites mature in the liver, are released into the bloodstream and infect the Red Blood Cells(RBCs). The dataset is used in this paper is taken from the National Institutes of Health(NIH) [16], [17]. An expert slide reader at the Mahidol-Oxford Tropical Medicine Research Unit manually annotated the images. The dataset contains 27,558 segmented cell images, with equal instances of 13,779 parasitized and 13,779 uninfected segmented red blood cell images. Positive samples contained plasmodium and negative samples contained no plasmodium but could contain other types of objects including staining artifacts/impurities. The patches of segmented red blood cells are of 3- channels (RGB) with a size variation of 110-150 pixels. The dataset is split into train and test set having the ratio of 80:20. The training set is further split into the training sample and validation sample in the ratio of 90:10 during the training of the CNN model. The images are re-sampled to 28x28x3 pixel dimensions, normalized for faster model convergence, converted into features and labels for testing and training classes and then used as the input to evaluate the CNN model [17], [18]. Fig. 4 and Fig. 5 show some samples from the dataset containing uninfected and parasitized segmented RBCs respectively.

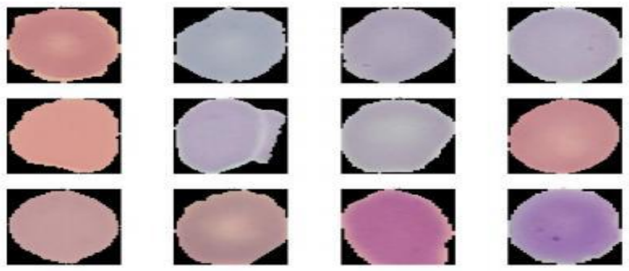


Fig.4. Uninfected RBCs images drawn from NIH Malaria dataset.

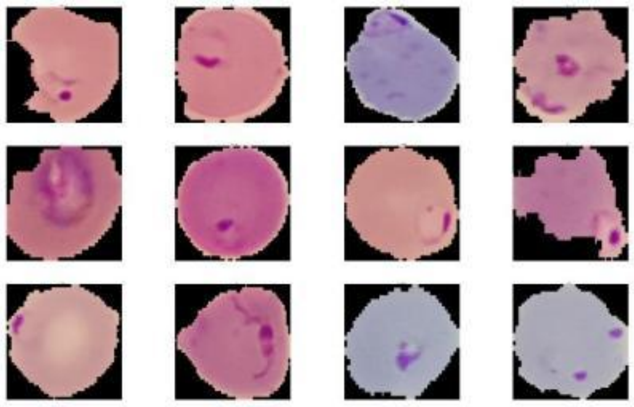


Fig. 5. Parasitized RBCs images drawn from NIH Malaria dataset.

Experimental Setup

This experiment is carried out on the Google Colaboratory environment. It is also known as Google Colab, is a product from Google Research. It is a cloud-based service through which allows anybody to write and execute python code with the help of a browser and is well suited to deep learning, data analysis, education and AI research work. Colab is a hosted Jupyter notebook without having to download, install, or run anything. It also provides free access to computing resources including GPUs, CPU, and RAM [19]. We can search the colab notebook using Google Drive. These resources available in Colab vary over time and at instant, our experiment has the following resources specification.

Table I. Hardware and software specification of Colab.

S.N.	Specification type	Description
1.	GPU	Tesla P100-PCIE-16GB
2.	CPU	Intel(R) Xeon(R) CPU @ 2.30GHz
3.	RAM	~12.72 GB
4.	Disk	~68.4 GB
5.	IDE	Colab Notebook
6.	Programing language	Python 3

In this experiment, the genetic algorithm as explained above was implemented on Google Colab with the NIH malaria dataset with 22,046 images as its training set and another 5,512 images as its testing set.
This experiment aimed at generating an optimized best CNN architecture with the best hyperparameters such that the classification accuracy of the testing set is very high.

C. Simulation and results

In this experiment, the total number of generations is 10 and each generation constitutes 10 populations. Each population contains the hyper-parameters of a CNN architecture. So, the Genetic algorithm(GA) was executed 10 times and took the execution time around 3,241 seconds on the Google Colab environment. The first generation has randomly generated populations as shown in Table. II. For every population of a generation, a CNN model is executed and their fitness is evaluated on the basis of the classification accuracy on the testing data. These fitness values are used as the selection criteria of the populations for the crossover and mutation process of GA. There are three layers in each population. Each layer contains the hyperparameters like total number of filters (NOF), size of filters (SZF) and the activation function (ACT) used in the designing of convolutional neural networks. The fitness evaluation of every population calculated on the classification accuracy of the testing data. It was noted that each execution of GA gave the best populations after tuning the hyperparameters of each layer.

Table. II. Randomly generated population

no.	Layer 1			Layer 2			Layer 3		
	NOF1	SZF1	ACT1	NOF2	SZF2	ACT2	NOF3	SZF3	ACT3
1	240	5	ReLU	160	3	Sigmoid	176	5	ReLU
2	240	3	ReLU	192	5	ReLU	48	3	ReLU
3	128	3	ReLU	112	2	Sigmoid	64	3	ReLU
4	16	2	ReLU	192	3	ReLU	208	2	Sigmoid
5	32	2	ReLU	80	5	Sigmoid	48	3	ReLU
6	64	2	ReLU	208	2	ReLU	240	3	Sigmoid
7	80	2	Sigmoid	80	2	ReLU	32	2	ReLU
8	208	2	ReLU	240	5	Sigmoid	240	5	ReLU
9	208	2	Sigmoid	176	3	ReLU	176	2	Sigmoid
10	16	5	Sigmoid	128	5	Sigmoid	128	7	Sigmoid

Here, Fig. 6 shows the training accuracy and the testing accuracy of the proposed model on training data for 100 of populations over 10 generations. In the first 2 generations, the training accuracy is very fluctuating even some populations have approximately 50% accuracy. From 3rd to 7th generations, the training accuracy is between 84% to 96 %. In 8th generation, there is only one population that has 50% accuracy. In the 9th and 10th generations, the training accuracy is maintained between 93% to 97%.

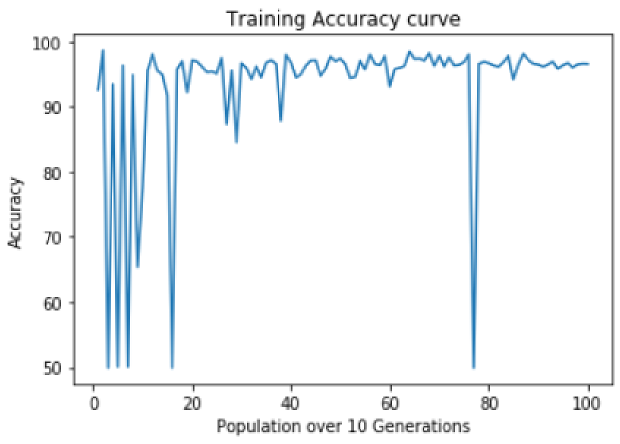


Fig. 6. Training accuracy for 100 of populations over 10 generations

Here, Fig. 7 shows the testing accuracy of the proposed model for 100 populations over 10 generations. Here, we can clearly see that the testing accuracy is quite similar to the training accuracy curve as shown in Fig. 5. The last 10th generation shows the highest testing accuracy throughout the curve.

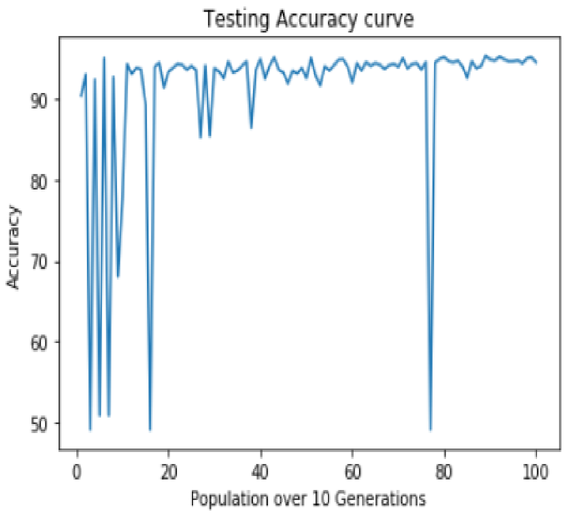


Fig. 7. Testing accuracy for 100 of population over 10 generations

From Fig.6 and Fig.7, it is concluded that the last 10th generation contains the highest testing as well as training accuracy. So the population of last generation has fit throughout the rest of the generations. Table III. gives us the data for the top 10 fit population and their accuracies. The first population is the fittest population because it contains the highest testing and training accuracy which is highlighted in the below Table III. Hence, the CNN model designed with the help of the first highlighted population's hyperparameters is most fit and accurate for the given dataset and could accurately diagnose the malaria cell with at most 95% accuracy.

Table III. Genetically generated fit population (10th generation) with their losses and accuracies

S. no .	Layer 1			Layer 2			Layer 3			Accuracy in %		Loss in %	
	NOF1	SZF1	ACT1	NOF2	SZF2	ACT2	NOF3	SZF 3	ACT3	Train	Test	Train	Test
1	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.52	95.17	9.07	13.43
2	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.49	95.10	9.89	14.10
3	208	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.48	94.97	9.46	14.76
4	208	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.83	94.86	9.54	13.38
5	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.57	94.74	10.91	16.31
6	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.23	94.70	10.22	14.10
7	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.30	94.59	9.81	14.23
8	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.22	94.59	9.70	14.04
9	64	2	ReLU	192	2	ReLU	240	3	Sigmoid	95.98	94.47	10.42	14.56
10	208	2	ReLU	192	2	ReLU	240	3	Sigmoid	96.22	94.34	9.54	13.84

Genetically optimized CNN model

The genetically optimized CNN architecture has a sequential CNN model that can effectively classify the parasitized cells and uninfected cells for diagnosing the Malaria disease as shown in Fig.8. The proposed CNN architecture has three convolutional layers and two fully connected layers (dense layers). The input to the model constitutes segmented cells of 28×28×3 pixel resolution. The first convolutional layer uses 2×2 filter size. The first convolutional layer has 64 filters with the ReLU activation function. The second convolution layer uses 2x2 filters size. It uses 192 filters with the ReLU activation function. The third convolution layer has 3x3 filter size and 240 filters with Sigmoid activation function. All the above three convolution layers have Max-pooling layers with a pooling window of 2x2 and 2 pixel strides.

The pooled output of the third convolutional layer is fed to the 1st fully connected layer that has 256 neurons and then fed into a 2nd fully connected layer that has 2 neurons. The 2nd fully connected layer is fed into the Softmax classifier that classify the type of Malaria cells. We also evaluated the performance of the genetically optimized CNN architecture such that the classification accuracy of the testing data is about 95% and the training data is about 97%. The optimized CNN model is executed in 43 seconds on the Google Colab environment with NVIDIA's Tesla P100 GPU. This model is trained over 20 epochs.

Fig. 9 and Fig. 10 show the model accuracy curve and model loss curve of training data set and validation data set w.r.t. epochs respectively.

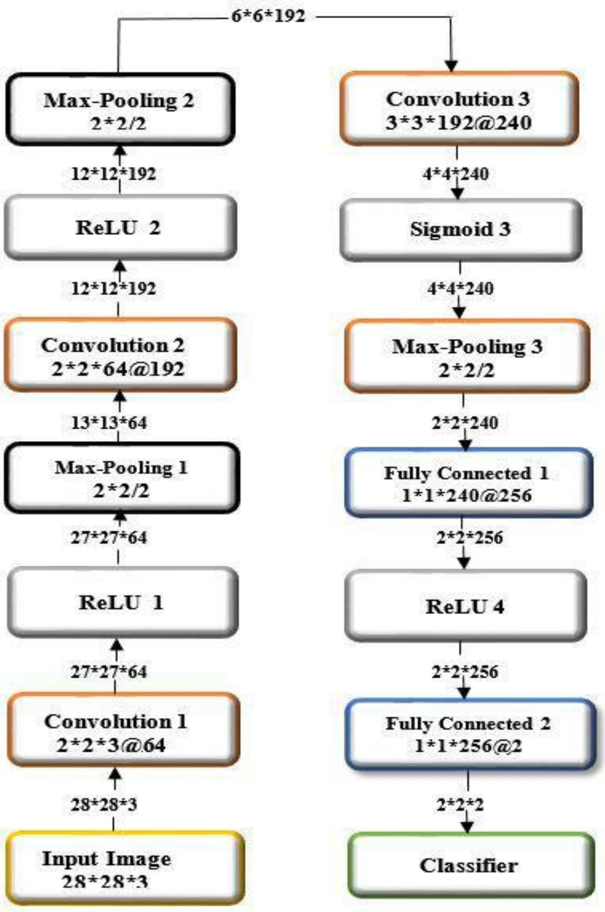


Fig. 8. Architecture of genetically optimized CNN model

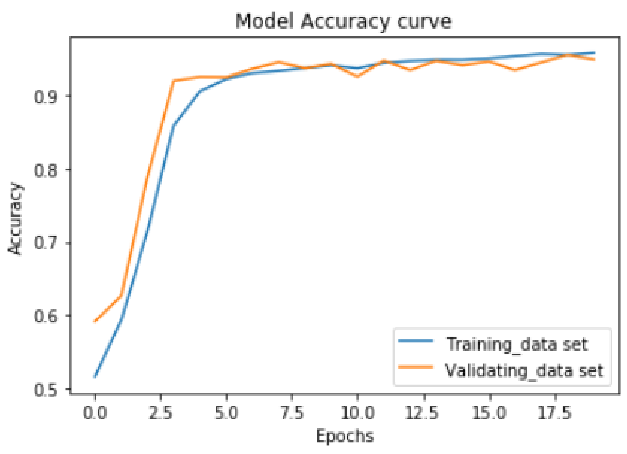


Fig. 9. Accuracy curve of the CNN Model for the training and validating data set w.r.t. no. of Epochs

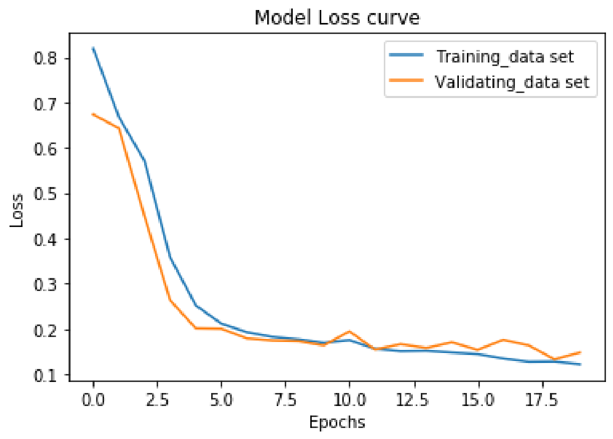


Fig. 10. Loss curve for the model for Training data set and Validating data set w.r.t. no. of Epochs

VI. CONCLUSION

In this paper, we proposed an automatic architecture design algorithm for CNNs with the help of genetic algorithm (in short named CNN-GA) which discovered a fittest CNN architecture to classify the parasitized cells and uninfected cells of NIH malaria dataset with 95.17% classification accuracy on testing data. We also found that, as the proposed algorithm was executed with high speed on Tesla P100 multi-core GPU of Google Colab environment, it reduced the training time and testing time to less than 1 hour. Now, we can also use this optimized CNN architecture design algorithm for other image classification problems. In the future, we can further optimize and increase the efficiency of the CNN by tuning the hyperparameters in fully connected layers.

REFERENCES

1. Robert Ouellette, Matthew Browne, Kotaro Hirasawa, "Genetic algorithm optimization of a convolutional neural network for autonomous crack detection", IEEE, 2004, doi: 10.1109/CEC.2004.1330900.
2. Tiago Carneiro, Raul V. Medeiros Da Nóbrega, Thiago Nepomuceno,Gui-Bin Bian, Victor Hugo C. De Albuquerque, and Pedro P. Rebouças Filho, "Performance analysis of Google Colaboratory as a tool for accelerating deep learning applications", DOI: 10.1109/access.2018.2874767, IEEE Access.
3. Daniel Weimer, Bernd Scholz-Reiter, Moshe Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection" CIRP Annals - Manufacturing Technology (2016).
4. Shikha Agrawal, Jitendra Agrawal, "Neural Network Techniques for Cancer Prediction: A Survey", Procedia Computer Science 60 (2015) ,pp. 769 – 774.
5. Ashray Bhandare and Devinder Kaur, "Designing Convolutional Neural Network architecture using genetic algorithms", Int'l Conf. Artificial Intelligence, ICAI'18.
6. Yu-Tzu Chang, Jinn Lin, Jiann-Shing Shieh and Maysam F. Abbod, "Optimization the initial weights of artificial neural networks via genetic algorithm applied to hip bone fracture prediction", Hindawi Publishing Corporation Advances n Fuzzy Systems Volume 2012, Article ID 951247, 9 pages doi:10.1155/2012/951247.
7. Earnest Paul Ijjina, C. Krishna Mohan, "Human action recognition using genetic algorithms and convolutional neural networks", Elsevier, November 2016.
8. Amin Kabir Anaraki, Moosa Ayati, Foad Kazemi, "Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms", Biocybern Biomed Eng (2018), https://doi.org/10.1016/j.bbe.2018.10.004
9. Ranjeeth Kumar Sundararajan, Sivagurunathan S, Venkatesh S, Jeya Pandian M, "Convolutional neural network based medical image classifier" International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-3, September 2019.
10. Kirankumar Kataraki, Sumana Maradithaya, "Scalable Handwritten Digit Recognition Application using Neural Network and Convolutional Neural Network On Heterogeneous Architecture", International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-3, September 2019.
11. Sanjay Kumar Ahuja, Manoj Kumar Shukla, Kiran Kumar Ravulakollu, "Surface Corrosion Grade Classification using Convolution Neural Network", International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-3, September 2019.
12. Frank H. F. Leung, H. K. Lam, S. H. Ling, and Peter K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm", IEEE Transactions On Neural Networks, Vol. 14, No. 1, January 2003.
13. John R. Koza, "Survey Of genetic algorithms and genetic programming", Wescon conference record, 1995.
14. Whitley, D. (1994), "A genetic algorithm tutorial. statistics and computing", 4(2). doi:10.1007/bf00175354.
15. R. Chelouah, P. Siarry, "A continuous genetic algorithm designed for the global optimization of multimodal functions", Journal of Heuristics, 6: 181–213 (2000).
16. https://ceb.nlm.nih.gov/repositories/malaria-datasets/
17. Sivaramakrishnan Rajaraman, Sameer K. Antani, Mahdieh Poostchi, Kamolrat Silamut, Md A. Hossain, Richard J. Maude, Stefan Jaeger, and George R. Thoma. "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images." PeerJ 6 (2018): e4568.
18. Aimon Rahman, Hasib Zunair, M Sohel Rahman, Jesia Quader Yuki, Sabyasachi Biswas, Md Ashraful Alam, Nabila Binte Alam, M.R.C. Mahdy, "Improving malaria parasite detection from red blood cell using Deep Convolutional Neural Networks",arXiv:1907.10418, July 2019.
19. https://research.google.com/colaboratory/faq.html

AUTHORS PROFILE

Manjit Jaiswal is currently working as an Assistant professor in Computer Science and engineering Department, Institute of technology, Guru Ghasidas Vishwavidyalaya A Central University Bilaspur,Chhattisgarh, 495009, India. He received Master of Technology (M-Tech) degree in 2012 from Maulana Azad National Institute of Technology, Bhopal, Madhya Pradesh, India. He has published more than 12 papers in reputed journals and conference like IEEE, UGC approved, etc. His research interest fields are Parallel Computing, Distributed System, Algorithms, Routing Algorithm, Machine Learing, IOT etc. He has more than 7 years of teaching experience.

Aditya Sahu is currently pursuing Bachelor of Technology in Computer Science and Engineering from School of Studies in Engineering and Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India. He is currently studying in the final year. His research interest fields are Machine Learning, Computer Vision, Network Security.

Md Tausif Zafar is currently pursuing Bachelor of Technology in Computer Science and Engineering from School of Studies in Engineering and Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India. He is currently studying in final year. His research interest fields are Machine Learning, Computer Vision, Network Security.