Malaria Cell Detection Using Evolutionary Convolutional Deep Networks

Bin Qin
Shenzhen University Information Center
Shenzhen University
Shenzhen, China
qinbin@szu.edu.cn

Yufan Wu
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
285176237@qq.com

Zhili Wang
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
912427166@qq.com

Haocheng Zheng
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
462695933@qq.com

***Abstract*—With the rapid development of deep learning and computer-vision, accurate identification of medical imaging has become one of the most important factors in medical diagnosis and decision-making. To this end, we propose a data-driven approach named Evolutionary Convolutional Deep Network (ECDN) to detect malaria parasites, which can use evolutionary algorithms to automatically generate deep neural networks, and optimized its network topology structure during the evolution process. Extensive experiments based on the large-scale thin-blood smear images data validate the effectiveness of ECDN for detecting malaria. To be specific, it has the advantage of being able to automatically generate an optimal network structure without the need for any prior knowledge of constructing a neural network, as compared to a traditional artificial convolution network. The experimental results show that the model robustness of ECDN is better. When the training set and test set are divided according to the ratio of 6 and 4, the best result is achieved, and the accuracy rate reaches 99.98%, which provides an important basis for this research.**

Keywords-Evolutionary Algorithm; Deep Learning; Malaria; Neural Network

I. INTRODUCTION

Malaria is a life-threatening disease caused by Plasmodium parasites that infect the red blood cells (RBCs)[1]. Computer-aided diagnostic (CADx) tools using machine learning (ML) algorithms applied to microscopic blood smear images have the potential to reduce clinical burden by assisting with triage and disease interpretation[2]. To overcome challenges of devising hand-engineered features that capture variations in the underlying data, Deep Learning (DL)[3], also known as deep hierarchical learning, is used with significant success. Liang[4] proposed a 16-layer CNN toward classifying the uninfected and parasitized cells. As the number of layers in the network deepens, the expressiveness and abstraction capabilities of the network become stronger, and the more parameters that can be adjusted. At present, most of the network architectures we use are manually set, and designing these architectures requires sufficient experience as a guide. This is a time-consuming and error-prone process, such as gradient explosion, over-fitting and so on. This raises a natural question: how to get the optimal network architecture and hyperparameters automatically? To this end, we have adopted the Evolutionary Convolutional Deep Networks, an algorithm that automatically calculates an effective neural architecture through evolutionary methods, and can be connected to any deep neural network platform such as Keras.

Neural architecture search (NAS)[5] is a process of automatic architecture engineering, which is the development trend of machine learning automation. NAS can be considered a sub-area of AutoML[6] and similar to hyperparameter optimization and meta-learning in some place. In this, paper, we have implemented a Neuro-Evolution[7] deep network, which can be widely used in image classification problems, automatically generate some deep neural networks according to different image classification data, and optimize the network structure in the process of evolution. The architecture search process is combined with evolutionary algorithms to achieve the effect of expanding the search space[8] and improving search efficiency. Finally, export an optimal network structure that can detect Plasmodium cells. The network structure can approximate the performance of an artificially designed network, even more than an artificially designed network. Meanwhile, we take into account the consumption of computing resources.

II. DATA DESCRIPTION

The malaria datasets we used released by NIH [9]. A total of 27,558 images of the dataset, which includes infected and uninfected images. Building an effective neural network model requires careful consideration of the network architecture and input data format.

The pre-processing are as follows.

- Sample purification: When drawing a random image of the data set, we found that some of the images in the training sample were incorrectly marked. So we will remove this part of the data to eliminate its impact.
- Image Rescaling: This operation re-adjusts the values of each dimension of the data (these dimensions may be independent of each other) such that the final data vector falls within the interval [0,1] or [−1,1].
- Data Enhancement[10]: In order to increase the diversity of data, we present the spatial diversity of the image by

means of rotation, mirroring, and cropping. The model trained based on this will also have better robustness.

In the malaria cell image, we see that the contrast between the data points of the valid image and the background color of the image is small. In other words, the noise has a greater impact on the results, so we use the histogram equalization method[11] to increase the global contrast of the image. In this paper, we use YUV-Space[12] for brightness equalization. It maps the original map to the new map according to a certain transformation formula. Among them, the cumulative distribution function of the histogram as a transformation formula is as follows.

$$S(k) = T(r_k) = \sum_{j=0}^{k} P_r(r_j) \tag{1}$$

$$P_r(r_k) = \frac{n_k}{N} \tag{2}$$

Where $P_r(r_j)$ is actually a histogram of an image with a pixel value of r_j, normalized to [0,1]. The histogram equalization calculation formula is as follows:

$$H(v) = round\left(\frac{S(v) - S_{min}}{(M \times N) - S_{min}} \times (L-1)\right) \tag{3}$$

Where S_{min} is the minimum value of the cumulative distribution function, M and N represent the number of length and width pixels of the image, L is the number of gray levels, and *v* is the pixel value in the original image.

III. Evolutionary Convolutional Deep Network

A. Evolutionary Algorithms

Evolutionary algorithm (EA)[13] is an algorithm used to solve the optimization problem, which is inspired by the idea of "survival of the fittest" in biology. Using evolutionary algorithms, we first should set the initial size of the population, then define our fitness function to assess how much each chromosome can solve the problem, and then iterate multiple times until a predetermined condition is met. Table I shows the evolutionary algorithm used in this paper.

TABLE I. Evolutionary Algorithm

Evolutionary Algorithm
input: max_generation: maximum number of EA generations population_size: population_size begin: generation ← 0. initial chromosomes. evaluate the initial population. while generation≤ max_ generation do select parents. create offsprings using the mutation operation. select the excellent chromosome with the highest fitness. evaluate generation. generation ← (generation + 1). return the best chromosome.

B. Initial Chromosome

The algorithm first initializes the first-generation population through user-defined parameters and then completes the creation of chromosomes, which represents the solution of the first-generation optimization problem. By assessing each chromosome, you can see how close it is to the ideal solution. The chromosomes that created the first-generation population are shown in Table II, and the chromosomes encoding the network structure are shown in Table III.

TABLE II. Create an Initial Population of Chromosomes

Create An Initial Population of Chromosomes
input: population_size: population_size begin: for i in population_size: Generate chromosome Add chromosome to initial population

TABLE III. Create Network Structure Encode Chromosomes

Create Network Structure Encode Chromosomes

input:
layer_list: network layer list
layer_list_len: maximum number of network layer list
add_pooling_layer_chioce: whether to add max pooling layer
begin:
initial an empty chromosome.
for i in layer_list_len -1:
if i == 0:
create the input cnn layer
append new_layer to chromosome
else:
if layer_type == 'cnn':
CreateLayer()
if new_layer is fully_connected_layer:
CreateLayer()
append new_layer to chromosome
Randomly create fully_connected_layer and append to chromosome.
return chromosome
 	
Function CreateLayer(layer_type)
if layer_type ← 'cnn':
Randomly create convolution
if add_pooling_layer_chioce:
Randomly create max_pooling layer
create BatchNormalization layer
else:
Randomly create fully connected
Randomly create dropout layer

C. Parent Selection

In each generation of evolutionary algorithms, we must ensure that parents are selected before mutation to use genetic operators and mutation mechanisms to derive offspring. The three common methods of parent selection are roulette selection methods, rankings and game selection. In this paper, we use the roulette selection method to select the parent. A detailed mutation process is introduced in Table IV.

D. Fitness Function

The selection of the fitness function directly affects the convergence speed of the genetic algorithm. Besides, the genetic algorithm is based on the fitness function to find the optimal solution if the fitness function is not well selected.

The complexity of the fitness function is the main component of the complexity of the genetic algorithm, so the design of the fitness function should be as simple as possible. Only in this way

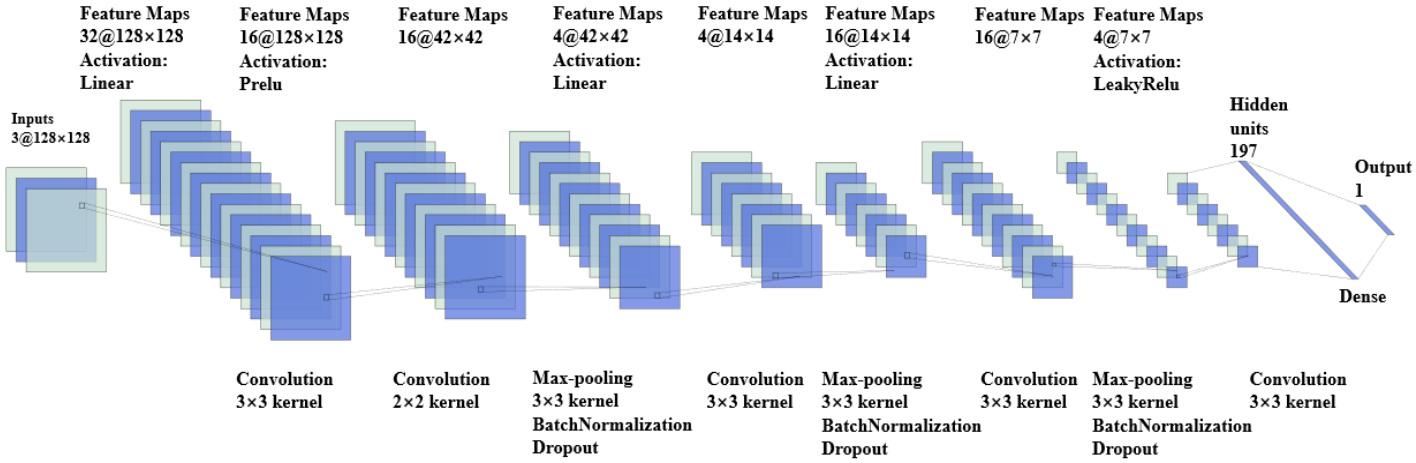

Fig. 1. Optimal network structure overview.

can we minimalize the time complexity of the calculation. The fitness function used in this paper is as follow.

$$fitness(i)=accuracy(i)+punish\_factor(i) \tag{4}$$

$$punish\_factor(i)=\frac{1}{layer\_num+unit\_num} \tag{5}$$

Where accuracy represents the accuracy of the network *i*, layer_num is the number of the layer, unit_num is the number of units in the fully connected layer.

TABLE IV. MUTATION OPERATION

Create An Initial Population of Chromosomes
input: threshold: threshold of mutation operation max_num: maximum number of layer layer_num: number of current layer list layer_type: randomly create layer type begin: layer = CreateLayer(layer_type) if threshold < layer_num < max_num: Add, Replace or Delete layer If layer_num <= threshold: Add or Replace layer If layer_num == max_num: Replace or Delete layer

IV. EXPERIMENTS

A. Experimental Setup

The hyperparameters setting is listed below.

- Physical calculation environment: 1080ti graphics card.
- Number of filters in 2D convolution: [10,100]
- Filter size for 2D convolution: [1,6]
- Network layer number range: [2,10]
- Kernel size for 2D max pooling: [1,6]
- Number of units in fully connected layers: [32,256]
- Dropout rate: (0,1]
- The selection for activation function of 2D convolution layer: [linear, leaky relu, prelu, relu]
- The selection for activation function of fully connected layer: [linear, sigmoid, softmax, relu]
- The selection for activation function of last fully connected layer: [sigmoid, softmax]

For each individual in the population, the number of iterations trained is no more than 13 times, effectively reduces the time spent by the algorithm on network training to improve the efficiency of the algorithm.

B. Experimental Results

In this paper, the self-organizing network structure obtained by EA is shown in Fig. 1. Through a lot of iteration and training, we finally get a model with a 8-layer network structure.

We found that this network structure showed excellent results by dividing the training set and test set for different proportions. The results of this network structure at different segmentation rates are shown in Table V.

TABLE V. ECDN PERFORMANCE AT DIFFERENT SEGMENTATION RATES OF THE DATASET

Ratio	Precision	Accuracy	F1 Score	Recall
1 : 9	0.9986	0.9997	0.9986	0.9986
2 : 8	0.9992	0.9992	0.9992	0.9992
3 : 7	0.9996	0.9987	0.9998	0.9998
4 : 6	0.9996	0.9998	0.9998	0.9998

In this experiment, we used traditional CNN and VGG-16 as the baseline, and the corresponding convolutional layers were 8 layers and 13 layers, respectively. Compared with these models, our model has better accuracy, robustness and simpler network structure, in which the convolutional layer has 5 layers. Since this is an image classification problem, it focuses on the accuracy of the model. Finally, the accuracy of our ECDN model

is 99.98%, which is beyond our expected results. Since this is an image classification problem, it focuses on the accuracy of the model. Finally, our ECDN model has an accuracy rate of 99.98%, which exceeds our expected results.

C. Model Evaluation

Model performance evaluation is an essential element. For the comprehensive verification of ECDN, we use four indicators to evaluate metrics including *accuracy*, *precision*, *F1 Score* and *recall*. We use the NIH officially published malaria dataset to conduct experiments that are expected to accurately identify malaria cells. In this paper, the precision of the ECDN model has reached 99.9%. The results are shown in Table VI.

TABLE VI. MODEL PERFORMANCE INDEX COMPARSION

Model	Precision	Accuracy	F1 Score	Recall
CNN	0.97	0.98	0.98	0.97
VGG-16	0.95	0.95	0.95	0.94
ECDN	0.99	0.99	0.99	0.99

D. Confusion matrix

To more comprehensively evaluate the network model, we introduce the Confusion matrix, which used to evaluate the model's performance. They are the most important indicators for evaluating the performance of a classification model.

According to the confusion matrix, the experiment has achieved the expected result for malaria detection. Specifically, we evaluate the network structure on the premise of 6:4 segmentation of the dataset. In 5517 positive samples, only 3 samples classified to parasitized class by mistake, while in 5507 negative samples, all samples are correctly classified. Fig. 2 shows the confusion matrix of the ECDN model.

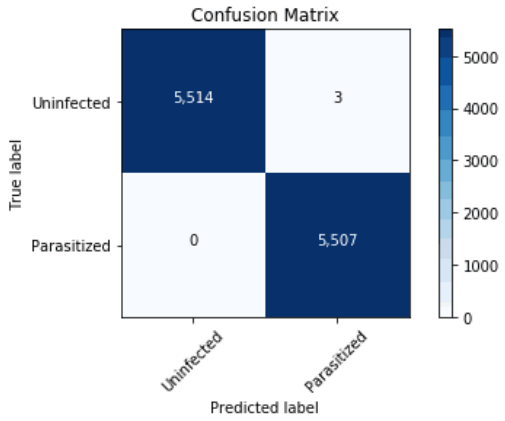

Fig. 2. Confusion Matrix.

V. CONCLUSION

In this paper, we propose a novel model based on the evolutionary algorithm, which can solve the problem that the design of network architecture and parameters in deep learning needs to manually interfere. A unique perspective of ECDN is that it can integrate different neural networks to automatically select the composition of the optimal network structure and corresponding parameters. Extensive experiments prove that the network model we trained is more accurate and stable than the traditional artificial tissue network.

In the future, we will consider adding cross-operations to our research and verifying the effectiveness of our model on more datasets, further improving mobility of the model to explore better network architectures.

ACKNOWLEDGMENT

This research was partially supported by grants from the Ministry of Education and the "Shenzhen University-Sangfor Cloud Computing Innovation Teaching Laboratory" (Grant No. 201801277010).

REFERENCES

[1] Rajaraman S, Jaeger S and Antani S K. Van Veldhuizen, "Performance evaluation of deep neural ensembles toward malaria parasite detection in thin-blood smear images," *https://peerj.com/articles/6977/*, May 28, 2019.

[2] Poostchi, Mahdieh, "Image analysis and machine learning for detecting malaria," *Translational Research*, vol. 194, pp. 36-55, Apr. 2018.

[3] LeCun Y, Bengio Y, Hinton G, "Deep learning," *nature*, vol. 521, pp. 436-444, May. 2015.

[4] Liang, Zhaohui, Andrew Powell, Ilker Ersoy, Mahdieh Poostchi, Kamolrat Silamut, Kannappan Palaniappan, Peng Guo. "CNN-based image analysis for malaria diagnosis," in *IEEE International Conference on Bioinformatics and Biomedicine*, Shenzhen, Guangdong, 2016, pp. 493-496.

[5] Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li LJ, Fei-Fei L, Yuille A, Huang J, Murphy K. "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision*, Zurich, 2018, pp. 19-34.

[6] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, Song Han. "AMC: AutoML for Model Compression and Acceleration on Mobile Devices," in *The European Conference on Computer Vision*, Germany, 2018, pp. 784-800.

[7] E.Real. "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, Sydney, 2017, pp. 2902-2911.

[8] Dufourq, Emmanuel, and Bruce A. Bassett. "Eden: Evolutionary deep networks for efficient machine learning," in *Pattern Recognition Association of South Africa and Robotics and Mechatronics*, South Africa, 2017, pp. 110-115.

[9] U.S. National Library of Medicine, "Malaria Datasets", *https://ceb.nlm.nih.gov/repositories/malaria-datasets*, Apr 16. 2018.

[10] Yan, Zhicheng. "Automatic photo adjustment using deep neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, pp. 11-15, May. 2016.

[11] Kim, Yeong-Taeg, "Contrast enhancement using brightness preserving bi-histogram equalization," *IEEE Transactions on Consumer Electronics*, vol. 43, pp. 1-8, Feb. 1997.

[12] Shi S, Wang L, Jin W, et al. "Color night vision based on color transfer in YUV color space," in *International Symposium on Photoelectronic Detection and Imaging 2007: Image Processing. International Society for Optics and Photonics*, Beijing, 2008, pp. 66230B.

[13] Coello, C.A.C., Lamont, G.B. and Van Veldhuizen. "Evolutionary algorithms for solving multi-objective problems", in *International Conference on Natural Computation*, New York: Springer, 2007, pp. 79-104.