Optimized Training for Convolutional Neural Network Using Enhanced Grey Wolf Optimization Algorithm

Akram Guernine
Laboratory of Embedded Systems, Department of Computer Science
Universityof Badji Mokhtar-Annaba, Po Box.12, Sidi Amar, Annaba, Algeria
E-mail: guernine24@yahoo.fr

Mohamed Tahar Kimour
Laboratory of Embedded Systems, Department of Computer Science
Universityof Badji Mokhtar-Annaba, Po Box.12, Sidi Amar, Annaba, Algeria
Center of research on environment, Annaba, Algeria
E-mail: rahatkimm@gmail.com

Keywords: deep learning, convolutional neural network (CNN), training algorithm, grey wolf optimization algorithm

Received: April 4, 2021

Convolutional Neural Networks (CNNs) are widely used in image classification tasks and have achieved significant performance. They have different applications with great success, especially in the medical field. The choice of architecture and hyperparameter settings of the CNN, highly influences both accuracy and its convergence speed. The empirical design and optimization of a new CNN architecture require a lot of expertise and can be very time-consuming. This paper proposes an enhanced Grey Wolf Optimization (GWO) algorithm to efficiently explore a defined space of potentially suitable CNN architectures, and simultaneously optimize their hyperparameters. Moreover, we introduce a spatial resolution reduction for a given image processing task, while taking skin cancer detection as a practical application. Through conducted experiments, we have shown that the obtained results are better than other classification methods in terms of accuracy and convergence speed.

Povzetek: Opisan je Gray Wolf Optimization (GWO) algoritem za iskanje primerne arhitekture konvolucijskih nevronskih mrež.

1 Introduction

As a class of deep neural network architectures, Convolutional neural networks (CNNs) has been recently presented as the most efficient solution towards achieving better recognition accuracy including pattern recognition, computer vision, speech recognition, graphical modeling, and signal processing, artificial intelligence, and many other fields [1-8],[27-30]. Especially, they have made huge impacts in the medical imaging domain. They are mostly applied to two-dimensional data such as images and videos. CNNs were inspired by the organization of the animal visual cortex [28]. CNNs can automatically learn hand-engineered filters as they were used in traditional computer vision algorithms. This independence from prior knowledge and human intervention in feature design is the main advantage of CNNs[8]. Sparse interaction, equivariant representation, and parameter sharing are three factors that play an important role in the learning process of a CNN [29].

There are different architectures for CNN implementation, and an optimal performance may not be achievable if the same architecture is applied on several tasks, and the CNN architecture must be specified for specific tasks However, it is computationally complicated to design an efficient CNN architecture for specific tasks, as several types of machine learning tasks exist in the industries [9].

Despite the robustness of the CNNs, some of its parameters still need to be optimized; such parameters can be roughly categorized into those for learning and those for network configurations. The network configuration has been reported to influence the recognition performance of CNNs [10], [11]. Therefore, the performance of a CNN is highly dependent on its architecture and hyperparameter settings.

Several studies have strived to provide the theoretical basis for the better performance of CNNs, but yet to provide the strategies for optimizing its parameters. This implies a shift in the concepts from visual features extraction to network structure configuration and parameters optimization [11] [28-32].

Traditionally, CNNs have been designed manually by researchers with a lot of experience in this area. Since the architecture of a well-performing CNN model can depend on the problem characteristics, the optimal choice of hyper-parameters is one of the major challenges when applying CNN-based methods. Often, it is unclear for a specific application how the CNN structure relates to the

accuracy; this is why it is not trivial to find the best performing CNN structure.

In this paper, we propose a framework for finding the optimal configuration of a CNN using an enhanced version of the Grey Wolf Optimizer (GWO) algorithm, with application on skin cancer detection. GWO [1] is a meta-heuristic optimization framework, which is inspired by the behavior of the grey wolves. We have modified the original GWO by introducing chaos theory with adaptive mechanisms to prevent the algorithm from falling into local optimum ad speed up its convergence, leading to high accuracy. In this study, the major contribution is the proposal of a novel training algorithm called acGWO-CNN for the training of the CNN model based on the acGWO algorithm. Here, the adaptive GWO, we call acGWO algorithm is used to establish the optimal CNN parameters.

The optimization performance of these algorithms was compared using several criteria such as their calculated errors and accuracy. This study used the digital images of skin cancer dataset. In this dataset, the digital size of the images is 28 x 28 pixels. The set of 80% images are used to train the model while 20% of them are for the model testing.

The structure of this paper is as follows: Section 2 covers some related works. Section 3 provides the basics of the CNN and its use in image processing. Section 4 is devoted to the GWO algorithm and the improvement we gave to it. In Section 5, the proposed acGWO-CNN method is described in detail. Experiments and discussion are presented in Section 6. Finally, Section 7 concludes the paper and defines some future works.

2 Related works

Recently, deep learning algorithms have attracted researchers in both academically and industrially fields. Neural network shows considerable performance on various domains. However, for complex systems, neural network's accuracy significantly deteriorates. This is why, research works focused on deep neural networks to learn hyperparameters and network structures [33-36, 38-39] using evolutionary algorithms (EAs). Indeed, in [33-35], the authors proposed an appropriate method to optimize hyper-parameters in CNNs using (EAs). The numbers of filters in each CNN layer with the size of the kernel are examples of the studied hyperparameters.

In [11], to improve the recognition accuracy by optimizing the results of the solution vectors on CNN, the training process was supported by the PSO algorithm. In [17], an approach based on whale optimization algorithm (WOA) is utilized for optimizing the weight and biases in the CNN model. In [38], an integrated CNN with a genetic algorithm was proposed to learn the structure of deep neural networks. This is based on a new coding scheme, which used a binary sequence with a fixed length for representing the structure of the network. In [40], genetic algorithms were used to determine the optimal initial weights of the CNN. In [39,41], the authors have proposed an optimization mechanism of the Convolutional Neural Networks (CNNs) using Particle Swarm Optimization (PSO).

In the above-mentioned existing works, meta-heuristics have been used to optimize CNN parameters for classification and object recognition. CNN Parameters concern hidden units, learning rates, and a determined number of epochs. They are optimized to decrease learning time and error rate. Different from the above-mentioned existing works, we introduce a spatial resolution reduction for a given image processing task, while applying an improved grey wolf algorithm to generate new efficient individuals and to select the best CNN architecture.

3 Convolutional networks based image processing

Compared to a traditional neural network, CNN is structured differently. A traditional neural network works with 3-dimensional layers in width, height, and depth manner, while composed of a set of neurons. Each layer is connected to all neurons. In CNN, a layer is only connected to a small portion of neurons in the previous layer.

It is worth noting that a CNN is a multi-layer neural network that basically consists of two types of layers: convolutional layers and pooling layers. The convolutional layers are the core building block of a Convolutional network. To produce the feature maps, the input is convolved with trainable filters of a specific size called the receptive field.

The pooling layers that are located between the convolutional layers progressively reduce the spatial size of the layers and thereby decrease the computational volume and the number of parameters. The pooling layers down-sample each depth slice of the input independently, according to the defined filter size. The commonly used pooling layer between a sequence of the convolutional layer has a filter size of 2×2 with a stride of 2. CNN architectures also include other types of layers such as a fully connected layer, normalization layer etc. Therefore, there are many parameters in a CNN structure such as the number of layers, number of feature maps, receptive field or filter size, and stride size that can be evolved by a genetic algorithm based on the nature of the problem and the available data (Figure 1).

CNNs can automatically learn hand-engineered filters as they were used in traditional computer vision algorithms. This independence from prior knowledge and human intervention in feature design is the main advantage of CNNs. Sparse interaction, equivariant representation, and parameter sharing are three factors that play an important role in the learning process of a CNN [29].

In traditional artificial neural networks (NNs), the relationship between input and output components was derived from matrix multiplication. But in CNNs, by using sparse interaction and creating kernels smaller than the inputs, and applying that to the whole image, this computational burden was reduced considerably. Because of parameters haring, the network only needs to

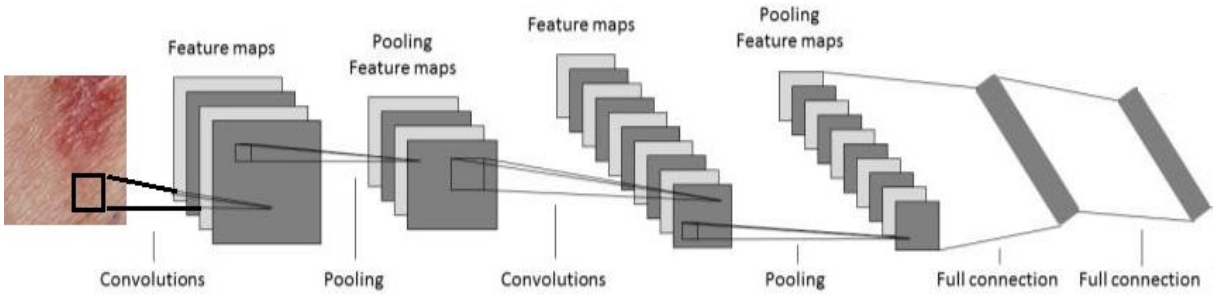

Figure 1: A typical architecture of a CNN Model.

learn one set of parameters at each location which improves the performance of CNNs over traditional NNs. Moreover, parameter sharing results in a deceptive property called equivariance in which by changing input, the output changes in the same way [30].

To date, in the image processing field, CNNs are the most efficient models for classifying images[33]. They have two distinct parts. As input, an image is provided in the form of an array of pixels. It has two dimensions for a grayscale image. The color is represented by a third dimension, of depth three to represent the fundamental colors (Red, Green, Blue). To learn spatial hierarchies of features through backpropagation, CNN is based on an adaptive approach and the fully connected layer is placed at its last building block, with other components such as convolution layers, pooling layers [16]. Features are extracted by the convolution layer, which typically performs the convolution operation and activation function. If an image is considered, they take advantage of the fact that proximity has a relation with similarity in it. To make the image processing computationally manageable, CNNs perform the filtering of connections by the proximity. In a given layer, each neuron is responsible for processing only a certain portion of the image and CNNs restrict the connections that any neuron accepts the inputs only.

Thus, the first part of a CNN is the actual convolutional part. It works as a feature extractor from images. An image is passed through a succession of filters, or convolution kernel, creating new images called convolution maps. Some intermediate filters reduce the image resolution by a local maximum operation. Finally, the convolution maps are flattened and concatenated into a feature vector. This CNN code at the output of the convolutional part is then plugged into the input of a second part, made up of fully connected layers. The role of this part is to combine the characteristics of the CNN code to classify the image. The output is a final layer with one neuron per category. The numerical values obtained are generally normalized between 0 and 1, of sum 1, to produce a probability distribution over the categories.

Creating a new CNN is costly in terms of expertise, materials, and the amount of annotated data required. The first step is to set up the architecture of the CNN, that is, the number of layers, their sizes, and the matrix operations that connect them. The training then consists of optimizing the coefficients of the network to minimize the classification error at the output.

4 Grey wolf optimization algorithm

In this study, the nature-inspired grey wolf optimizer (GWO) algorithm [1] was chosen due to several advantages including having a few tuning parameters, fast convergence, and its good capability to address optimization problems. In addition, we have given our proper improvement to yet enhance its accuracy, while preventing it from falling into the local optimum.

4.1 Grey wolf optimizer algorithm

GWO algorithm is based on the principle of hunting. Wolves are a member of a group in which a number of the grey wolves, which contribute to hunting. Wolves in a group are classified according to the leadership quality. There are four types of wolves in a group, identified as alpha (α), beta (β), delta (δ), and omega (ω). The decision-maker in the group is α that is the leader of the hunting process.

The rest of the wolves have their dominance decreasing sequentially in this order: beta (β), delta (δ) and omega (ω). Wolves of type ω are involved in the hunting process, and they transfer their better positions to their superiors. In this process, which is the course of the chasing, the first grey wolves search the location of prey and enclose them. α, β, and δ have better information about the potential area of prey for mathematical simulation of hunting action. The mathematical model of the enclosing has the form below:

$$\vec{D} = \left| C \cdot \vec{X}_p(t) - \vec{X}(t) \right| \tag{1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{2}$$

where t is the existing iteration, A and C are coefficient vectors, Xp is the location vector of the prey, and X indicates the location vector of a grey wolf. The vectors A and C are calculated as follows:

$$\vec{A} = 2\,a\,r_1 - a \tag{3}$$

$$\vec{C} = 2\,r_2 \tag{4}$$

where components of a linearly decreased from 2 to 0 over the course of iterations.

$$a = 2(1\text{-}t/T) \tag{5}$$

where "t" stands for the minimum iteration number, and "T" is the maximum iteration number.

and r1, r2 are random values in [0, 1]. Grey wolf hunting process is calculated as following equations:

$$\begin{aligned}\vec{D}_{\alpha} &= \left|\vec{C}_1 \cdot \vec{X}_{\alpha} - \vec{X}\right| \\ \vec{D}_{\beta} &= \left|\vec{C}_2 \cdot \vec{X}_{\beta} - \vec{X}\right| \\ \vec{D}_{\delta} &= \left|\vec{C}_3 \cdot \vec{X}_{\delta} - \vec{X}\right| \\ \vec{X}_1 &= \vec{X}_{\alpha} - \vec{A}_1 \cdot \vec{D}_{\alpha} \\ \vec{X}_2 &= \vec{X}_{\beta} - \vec{A}_2 \cdot \vec{D}_{\beta} \\ \vec{X}(t+1) &= \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}\end{aligned} \quad (6)$$

where $\vec{D}_{\alpha}, \vec{D}_{\beta}, \vec{D}_{\delta}$ respectively, are the distance vectors between the $\vec{X}_{\alpha}, \vec{X}_{\beta}, \vec{X}_{\delta}$ and the $\vec{X}$, $\vec{A}_1$, $\vec{A}_2$, $\vec{A}_3$, and $\vec{C}_1, \vec{C}_2, \vec{C}_3$ are the coefficient vectors; $\vec{X}_1, \vec{X}_2, \vec{X}_3$ are the direction vectors for calculating the next position of a grey wolf; Ψ is the random coefficient factor; d is the dimension of the search space; the variable for adjusting the movement freedom is denoted by M.

4.2 Improving GWO

As a new meta-heuristic search algorithm, GWO is inspired by the social behavior of leadership and the hunting process of grey wolves. In the original GWO, a great part of the iterations is dedicated to the exploration and the rest part is concerned with the exploitation. In doing so, the impact of the right balance is overlooked, between these two parts, to guarantee an accurate approximation of the global optimum. Despite of having a good convergence rate, the convergence rate of the algorithm is affected because GWO still cannot always perform a good balance in finding the global optima. The solutions are still prone to falling into the local optimum at the later phase of the search. The consequence is premature convergence leading to suboptimal solutions.

So, to reduce this effect and enhance its efficiency, we propose some improvements to GWO, by introducing combined adaptation and chaotic mechanisms, while considering different proportions according to the hierarchy of grey wolves: alpha, beta, and delta, in a way that allows guiding the search process towards the best elements.

Chaos system. Chaos is a category of unique deterministic random-like process found in a non-linear dynamical system, which is a non-periodic, non-converging; and confined it has been applied extensively in different fields including computer science, operations research, physics, biology, etc. A lot of chaotic maps have been explored [6]. Specifically in this work, for GWO algorithm, instead of the random number generator, we introduce a chaotic logistic map system in the first GWO stage to generate the initial population with high diversity. Moreover, we use this type of chaotic map in the second GWO phase to select different trial individuals from the population in the mutation operation. The logistic map is one of the most well-known one-dimensional discrete-time systems with a chaotic behavior. It is widely used in especially the computer science field, due to its simple and elegant form. It is expressed as follows.

$$\text{Logistic map, } Xn+1 = 4\lambda\, Xn\,(1-Xn), \quad (7)$$

where X represents the population at any given time t, and λ is a control parameter that represents the growth rate. Generally, the λ constant is in the range of [3.57, 4]. In the study, its value was determined as 3.9, for the logistic map to be totally in a chaotic state [11].

We use this type of logistic chaotic system at the generation of the initial population phase, and at the population evolution phase.

Initial population generation. In the initial population generation, every element of an individual is generated using:

$$minValue + chaos \times (maxValue - minValue) \quad (8)$$

where, minValue and maxValue stand for minimum and maximum values of every element of an individual. Chaos is a chaotic randomly generated value ranging in [0, 1]. The initial population is generated by the chaotic maps, which can form a feasible solution space, with a good distribution by the properties of randomicity, and ergodicity of chaos. Chaotic sequences can guarantee the diversity of the initial population, speed up its convergence, and enhance global search capability.

At the evolution phase, the transition between exploration and exploitation phases is generated by the adaptive values of "*a* ", and "*A*". To find a global optimum with a fast convergence speed, in acGWO , we balance these two phases by fine-adjusting the parameters "*a*" and "*A*", guiding the value of "*a*" to decrease from 2 to 0 as follows:

$$\lambda = (1-t)2/T2$$

$$a = 2 - chaos(t)\lambda \quad (9)$$

where chaos(t) is a logistic map chaotic random value ranging in [0, 1] , "T" stands for the maximum number of the iterations, and "t" stands for the current iteration. Associated with "A" in equation (3), the parameter "a" decreases linearly from "2" to "0" over the courses of the iterations. To efficiently manage the exploration and the exploitation, we modify the original equation of the parameter "a" as follows:

$$A = 2a \times chaos(t) - a\,, \quad (10)$$

where chaos is a chaos-based random value in [0, 1].

The parameter "C" exhibits a random behavior in the range of [0,1] using the equation (4). It is very useful when the local optima stagnation occurs, especially in the final iterations. It directly affects the exploration of GWO in a way to avoid the local optima, though it is pure random walk behavior. We introduce the chaos theory, via reformulating the parameter C as follows:

$$C(t) = 2 \times chaos(t), \quad (11)$$

parameter	baseline	optimization value
number of filters in C1	6	4-100
number of filters in C2	16	4-100
kernel size in C1	5	3,5,7
kernel size in C2	5	3,5,7
activation function in C1	sigmoid	sigmoid, relu, tanh
activation function in C2	sigmoid	sigmoid, relu, tanh
activation function in FC1	sigmoid	sigmoid, relu, tanh
activation function in FC2	sigmoid	sigmoid, relu, tanh
number of neurons in FC1	120	4-200
number of neurons in FC2	84	4-200
batch size in the training	10	10-100
optimizer	SGD	SGD

Table 1: Wolf definition in our approach.

where t is the current iteration and chaos(t)is a chaos-based random value in [0, 1].

The chaotic varying value can mitigate the local optima due to the non-repetition and the ergodicity property of the chaos [1,4]. Thus, the enhanced GWO algorithm we call acGWO with a better hunting mechanism is proposed. It focuses on a proper balance between the exploration and the exploitation, leading to optimal performance, and promising candidate solutions.

Adapting the weighted factors. In the original GWO, individuals are updated by considering the average combination of the alpha, beta, and delta wolves. This mechanism guides individuals in the same proportion towards the best elements. However, it has been proved that this is not the best strategy [14], since that mechanism produces a limited exploration of the search space. To remedy this problem, we defined the weights in a way that allows guiding the search process towards the best elements, but considering different proportions according to the hierarchy of the grey wolves. Therefore, in acGWO, wolves are updated using the following formulation:

$$X = chaos(t)\ (X1 + X2/2 + X3/3) \quad (12)$$

where X1, X2, and X3 are the solutions of alpha, eta ad delta wolves. We have conducted experiments on some benchmark functions, to compare acGWO with the GWO and other popular metaheuristics, including genetic algorithms, differential evolution algorithm, PSO algorithm, and BA. We have found that acGWO exhibits faster convergence and better accuracy over all these metaheuristics.

The CNN was trained in this study using an enhanced algorithm for two reasons, the first is its simple structure and its ease of implementation, and the second is its parameter-less nature.

5 The proposed method

As mentioned earlier, the main contribution of this paper is to develop a training algorithm for CNN, meaning that finding the best values for the parameters of CNN. In this paper, our enhanced Grey Wolf optimizer algorithm is used as the training algorithm, for two reasons: first, it has a simple structure, while the second reason; it does not have any controlling parameters.

In the proposed method, the main configuration of the CNN architecture is based on LeNet-5 [31-32]. This is a relatively simple architecture consisting of two convolutional layers, two pooling layers, two fully connected layers, and an output layer. We choose it because it is often used as comparison architecture in experiments. The pooling layer uses max pooling.

In Table 1, we reuse original parameter values that have been also used in [31] to facilitate comparison. It shows the baseline parameters and optimization parameters of LeNet-5 where the convolution layers are C1 and C2, and the total coupling layers are FC1 and FC2. The number of filters in the convolution layers C1 and C2 is optimized between 4 and 100, and the kernel size is 3, 5, and 7. The number of neurons in full connect layers optimizes between 4 and 200. The activation function of each layer uses sigmoid, *relu*, and *tanh*, and the batch size is optimized from 10 to 100. The optimizer uses *Adam* or *Stochastic Gradient Descent (SGD)* with a learning rate of 0.01, respectively. In building the model, we used the open-source neural network library Keras. All experiments are performed on a machine with an Intel Core i7-6600K processor, 8GB RAM, GeForce GT650M -2GB.

5.1 The encoding of the wolves

A CNN is composed of two basic parts of feature extraction and classification. Feature extraction includes several convolution layers followed by max-pooling and an activation function. The classifier usually consists of fully connected neural network layers. In the proposed method, the weights set of the neural network is considered as the structure of each star. For this purpose, a vector of real values has been used. This vector contains all the weights of the neural network.

5.2 Initialization of the wolves

In terms of the population initialization, after the size of the population is set up, wolves are randomly created until reaching the population size. Each wolf is initialized using equation (8). The value of the classification error

Input: dataset, *#Epoch, #PopSize, Upper Bound, Lower Bound*
Output: Best Solution
Procedure:
build the initial population Xi (i = 1, 2, …, N), using equation (8)
Update the weights and biases of CNN using each wolf in the population
Calculate the fitness of each wolf of the population via equation (13)
Xα = the first-best wolf
Xβ = the second-best wolf
Xδ = the third-best wolf
while (t < Max number of iterations)
  for each wolf
    Update the position of the current wolf by equation (12)
    Check the boundaries of the current wolf
    Update the weights and biases of CNN using the current wolf
    Evaluate the fitness value of the current wolf via equation (12)
  end for
  Update ***a*** by equation (9)
  Update A and C by equation (10) and (11), respectively
  Update Xα, Xβ and Xδ
  t = t+ 1
end while
return Xα as the best solution

Figure 2: Pseudocode of the acGWO-CNN algorithm.

depends on the correctly classified samples, meaning that, the CNN is used to classify the samples.

6 Experiments and discussion

In this study, we experiment with DermIS Digital Database. Each image is given a defined label. In the experiment, optimization has been performed with acGWO-CNN every 5 epochs, and learning is performed based on the obtained parameters. Each experiment is performed 10 times, to finally take only the average value.

Table 2 depicts the comparison between our methods and the above-mentioned existing ones using the accuracy and variance metrics. At the epoch 5 of the learning, the accuracy of CNN optimized with acGWO has 98.78%, which is higher than 93.01% of CNN without optimization. When looking at any epoch, it is found that acGWO-CNN can obtain higher accuracy than Baseline CNN. Especially in the 1st epoch, Baseline CNN is 19.24%, while acGWO-CNN has a high accuracy of 96.52%, and the variance value is also low at 0.12, so it can see that the optimized CNN converges to high accuracy at an early stage. From this, it is thought that the proposed acGWO-CNN can find the optimal parameters and can obtain better results compared to the baseline CNN. Also, the average optimization calculation time for acGWO-CNN is 2621 seconds. Thus, as depicted by Table 2, compared to the four other existing methods, the proposed acGWO-CNN presents the best results according to the accuracy and variance metrics.

epoch	baseline CNN		GA-CNN		PSO-CNN		WOA-CNN		acGWO-CNN	
	accuracy	variance value	accuracy	variance value	accuracy	variance value	accuracy	variance value	accuracy	variance value
1	19.24	46.21	86.21	0.28	89. 04	0.173	91.09	0.163	96.52	0.12
2	76.22	38.85	87.82	0.21	90.16	0.148	92.87	0.141	97.97	0.048
3	87.45	2.45	90.12	0.20	91.04	0.123	93.13	0.118	98.02	0.08
4	91.02	0.54	93.17	0.19	94. 05	0.114	95. 12	0.113	98. 35	0.06
5	93.01	0.22	94.22	0.18	95.48	0.110	96.13	0.082	98.78	0.02

Table 2: Comparison using the accuracy and the variance metrics.

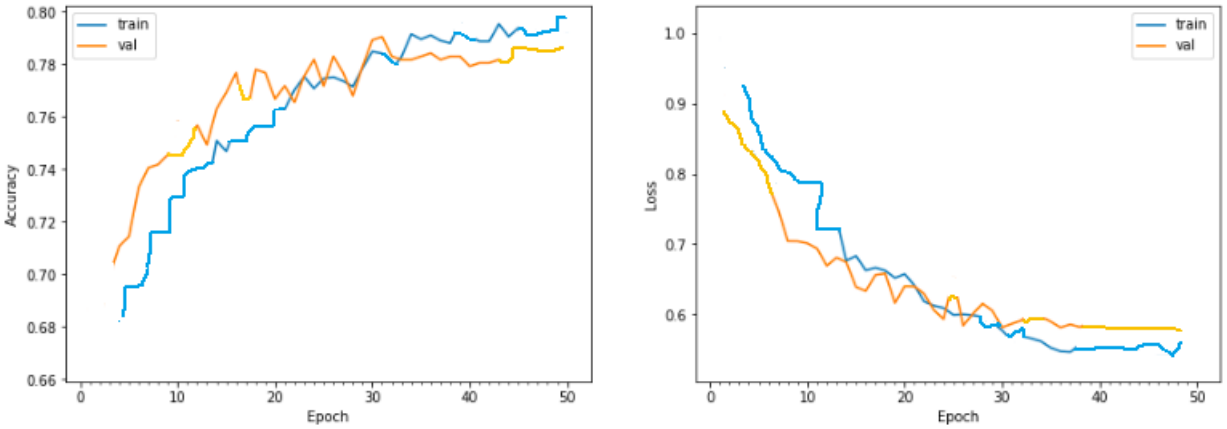

Figure 3: Model accuracy and loss.

Moreover, we have conducted other experiments with greater numbers of iterations. From these experiments, the acGWO-CNN has found parameters that can be successfully learned through optimization. These parameters define a best CNN architecture. The population size used in this study is equal to 40 wolves, while the maximum number of iterations is equal to 20. The proposed method is compared against four existing deep learning methods, that are, standard CNN, GA-based CNN [8], PSO-based CNN [11], and Whale optimization algorithm-based CNN[17]. The competitive accuracy and loss are illustrated in Figure 3.

7 Conclusion and future work

This paper proposed a new automatic CNN architecture design approach, based on adaptive Grey Wolf Optimizer (acGWO) algorithm for Convolutional Neural Network (CNN). Experiments are performed using a skin dataset, and achieved better results than the baseline CNN. In the results using the skin cancer dataset, the accuracy of the baseline CNN is 93.01% at the 5th epoch, compared to 98.78% for acGWO-CNN. Baseline CNN has a higher variance value at the beginning of the first epoch, such as 46.21, while acGWO-CNN is 0.12, and it has been found that it converged with high accuracy from the first epoch. In addition, baseline CNN has a large variance value of 0.22 at the 5th epoch, and a stable learning is not possible. On the other hand, acGWO-CNN exhibited a variance of 0.08 at the 3rd epoch at the highest and can be stably learned. From these results, it is considered that the optimization by acGWO can find very good hyperparameters and can provide high classification accuracy. As a future work, we plan to investigate the effectiveness of the proposed method by applying it to images in various fields. We also plan to investigate other metaheuristics-based CNN architectures to compare the effectiveness of the computational cost and complexity.

References

[1] Mirjalili, S. M., Lewis, A. Grey wolf optimizer. (2014).*Advances in Engineering Software*, Elsevier, 69, 46–61. https://www.sciencedirect.com/science/article/abs/pii/S0965997813001853

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006. http://www.cs.toronto.edu/~fritz/absps/ncfast.pdf

[3] Ali Bakhshi, Nasimul Noman, Zhiyong Chen. (2019) Fast Automatic Optimization of CNN Architectures for Image Classification Using Genetic Algorithm, IEEE Congress on Evolutionary Computation (CEC), IEEE, pp. 1283-1290 https://ieeexplore.ieee.org/abstract/document/8790197

[4] C. Letellier. (2013). Chaos in nature. *World Scientific Publishing Company*, pp.81 https://www.deboecksuperieur.com/ouvrage/9782711791408-le-chaos-dans-la-nature

[5] K. Price, R. Storn. (1997). Differential evolution: a simple evolution strategy for fast optimization. *Dr Dobb's J Software Tools* 22(4).pp 18–24. https://link.springer.com/article/10.1023/A:1008202821328

[6] Coelho Ld, Ayala HV, Mariani VC.(2014). A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. *Appl Math Comput*, Elsevier, 234. 452-459. https://www.sciencedirect.com/science/article/abs/pii/S0096300314002124

[7] U. Yüzgeç, M. Eser. (2018). Chaotic based differential evolution algorithm for optimization of baker's yeast drying process. *Egyptian Informatics Journal*, Elsevier, 19. 151–163 https://www.sciencedirect.com/science/article/pii/S1110866517302839

[8] M. Suganuma, S. Shirakawa, and T. Nagao. (2017). A genetic programming approach to designing convolutional neural network architectures. *in Proceedings of the Genetic and Evolutionary Computation Conference*, ACM, Germany, 497–504. https://dl.acm.org/doi/10.1145/3071178.3071229

[9] B. Wang, Y. Sun, B. Xue, and M. Zhang. (2018). Evolving Deep Convolutional Neural Networks by

Variable-Length Particle Swarm Optimization for Image Classification. *IEEE Congress on Evolutionary Computation*, IEEE, Brazil. 1-8. https://ieeexplore.ieee.org/abstract/document/8477735

[10] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. (2009). What is the best multi-stage architecture for object recognition?. *IEEE International Conference on Computer Vision*, IEEE, Japan, 12. 2146-2153 https://ieeexplore.ieee.org/document/5459469

[11] T. Yamasaki, T. Honma, and K. Aizawa. (2017). Efficient Optimization of Convolutional Neural Networks Using Particle Swarm Optimization. *IEEE 3rd International Conference on Multimedia Big Data*, IEEE, Usa, 3. 70-73. https://ieeexplore.ieee.org/document/7966719

[12] J. Kennedy, R. Eberhart. (1995). Particle swarm optimization. *IEEE International Conf. on Neural Networks*, IEEE, Australia, 4. pp.1942-1948. https://ieeexplore.ieee.org/abstract/document/488968

[13] S. Kumar, D. Datta, S. Kumar Singh, A. T. Azar, and S. Vaidyanathan. (2015). Black hole algorithm and its applications. Computational Intelligence Applications in Modeling and Control., Springer International Publishing, https://www.springerprofessional.de/en/black-hole-algorithm-and-its-applications/2262522

[14] H. J. Kelley. (1960). Gradient Theory of Optimal Flight Paths. .*aiaa.*, vol. 30, no. 10, pp. 947–954. https://arc.aiaa.org/doi/10.2514/8.5282

[15] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu.(2017). Towards Better Analysis of Deep Convolutional Neural Networks, *IEEE Trans Vis Comput Graph, IEEE*,pp. 91 – 100 https://ieeexplore.ieee.org/document/7536654/metrics#metrics

[16] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. (2015). Understanding neural networks through deep visualization, *ICML Deep Learning Workshop*, ArXiv, France. https://arxiv.org/abs/1506.06579

[17] Long Zhang, Hong Jie Gao, Jianhua Zhang, Benjamin Badami. (2020). Optimization of the Convolutional Neural Networks for Automatic Detection of Skin Cancer, *Open Medcine*, De Gruyter Open Access, 15: 27-37. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7026744/#

A. A. Alzaidi, M. Ahmad, H. S. Ahmed, and E. Al Solami. (2018). sine-Cosine Optimization-Based Bijective Substitution-Boxes Construction Using Enhanced Dynamics of Chaotic Map. *Complexity*, hindawi, vol. 2018. https://www.hindawi.com/journals/complexity/2018/9389065/

[18] A. M. Taha, S.-D. Chen, and A. Mustapha. (2015). Bat Algorithm Based Hybrid Filter- Wrapper Approach. *Adv Oper Res*, hindawi, vol. 2015. https://www.hindawi.com/journals/aor/2015/961494/

A. M. Taha, S.-D. Chen, and A. Mustapha. (2015). Natural Extensions: Bat Algorithm with Memory. *J. Theor. Appl. Inf. Technol.*, vol. 79, no. 1, pp. 1–9. http://dspace.uniten.edu.my/handle/123456789/10127

Ogudo, K.A.; Muwawa Jean Nestor, D.; Ibrahim Khalaf, O.; Daei Kasmaei, H. (2019). A Device Performance and Data Analytics Concept for Smartphones' IoT Services and Machine-Type Communication in Cellular Networks. *Symmetry*, *11*, 593. https://www.mdpi.com/2073-8994/11/4/593

[19] S. Q. Salih, A. A. Alsewari, B. Al- Khateeb, and M. F. Zolkipli. (2019). Novel Multi-swarm Approach for Balancing Exploration and Exploitation in Particle Swarm Optimization. *Recent Trends in Data Science and Soft Computing*, pp. 196–206. https://link.springer.com/chapter/10.1007/978-3-319-99007-1_19

[20] Z. A. Al Sudani, S. Q. Salih, Z. M. Yaseen, and others. (2019). Development of Multivariate Adaptive Regression Spline Integrated with Differential Evolution Model for Streamflow Simulation. *J. Hydrol*, Elsevier, pp. 1–15. https://www.sciencedirect.com/science/article/abs/pii/S0022169419302513

[21] A. P. Piotrowski, J. J. Napiorkowski, and P. M. Rowinski. (2014). How novel is the novel black hole optimization approach?. *Inf. Sci. (Ny)*, Elsevier,v 267. pp 191-200 https://www.sciencedirect.com/science/article/abs/pii/S0020025514000462

[22] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy. (2015). Simulated Annealing Algorithm for Deep Learning. in *Procedia Computer Science*, Elsevier,V 72, 2015, pp 137-144 https://www.sciencedirect.com/science/article/pii/S1877050915035759

[23] A. R. Syulistyo, D. M. J. Purnomo, M. F. Rachmadi, and A. Wibowo. (2016). Particle swarm optimization (PSO) for training optimization on convolutional neural network (CNN). *J. Ilmu Komput*. dan Inf, vol. 9, no. 1, pp. 52–58, https://jiki.cs.ui.ac.id/index.php/jiki/article/view/366

[24] Boukaye Boubacar Traore, Bernard Kamsu-Foguem, Fana Tangara. (2018). Deep convolution neural network for image recognition, *Ecological Informatics*, Elsevier, pp257-268. https://www.sciencedirect.com/science/article/abs/pii/S1574954118302140

I. Arel, D. C. Rose, T. P. Karnowski, et al. (2010). Deep machine learning new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, IEEE, vol. 5, no. 4, pp. 13–18. https://ieeexplore.ieee.org/document/5605630

[25] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda. (2003). Subject independent facial expression recognition with robust face detection using a

convolutional neural network. *Neural Networks*, Elsevier, vol. 16, no. 5-6, pp. 555–559. https://www.sciencedirect.com/science/article/abs/pii/S0893608003001151

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. (2016). *Deep learning*, vol. 1. MIT press Cambridge https://www.deeplearningbook.org/

[27] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, Elsevier, vol. 234, pp. 11–26. https://www.sciencedirect.com/science/article/abs/pii/S0925231216315533#!

[28] Tatsuki Serizawaa, and Hamido Fujita. (2017). Optimization of Convolutional Neural Network Using the Linearly Decreasing Weight Particle Swarm Optimization, *Computer Science*, ArXiv. https://arxiv.org/abs/2001.05670

[29] Haiman Tian et al. (2018). Automatic Convolutional Neural Network Selection for Image Classification Using Genetic Algorithms, *IEEE International Conference on Information Reuse and Integration*, Usa. Pp 444-451. https://ieeexplore.ieee.org/abstract/document/8424742

[30] D. Hossain, G. Capi, and M. Jindai. (2018). Optimizing deep learning parameters using genetic algorithm for object recognition and robot grasping, *Journal of Electronic Science and Technology*, vol. 16, no. 1, pp. 11–15. http://www.journal.uestc.edu.cn/article/doi/10.11989/JEST.1674-862X.61103113

[31] O. E. David and I. Greental. (2014). Genetic algorithms for evolving deep neural networks, *in Genetic and Evolutionary Computation Conferenc*e, Usa. pp. 1451–1452. https://dl.acm.org/doi/proceedings/10.1145/2576768

[32] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton. (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm, *in Workshop on Machine Learning in High-Performance Computing Environments*, ACM, pp. 4:1–4:5. https://dl.acm.org/doi/10.1145/2834892.2834896

[33] F. H.-F. Leung, H.-K. Lam, S.-H. Ling, and P. K.-S. Tam. (2003). Tuning of the structure and parameters of a neural network using an improved genetic algorithm, *IEEE Transactions on Neural networks*, vol. 14, no. 1, pp. 79–88. https://ieeexplore.ieee.org/document/1176129

[34] L. Xie and A. Yuille. (2017). Genetic CNN. *in IEEE International Conference on Computer Vision*, pp. 1388–1397. https://ieeexplore.ieee.org/document/8237416/

[35] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton. (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm, *in Workshop on Machine Learning in High-Performance Computing Environments*, Usa. p. 4. https://www.osti.gov/biblio/1567643-optimizing-deep-learning-hyper-parameters-through-evolutionary-algorithm-mlhpc-proceedings-workshop-machine-learning-high-performance-computing-environments-article

[36] E. P. Ijjina and K. M. Chalavadi. (2016). Human action recognition using genetic algorithms and convolutional neural networks, *Pattern Recognition*, Elsevier, vol. 59, pp. 199–212. https://www.sciencedirect.com/science/article/abs/pii/S0031320316000169

[37] Arie Rachmad Syulistyo et al. (2016). Particle Swarm Optimization (PSO) For Training Optimization on Convolutional Neural Network (CNN), *Journal of Computer Science and Information*. 9/1,pp 52-58 https://jiki.cs.ui.ac.id/index.php/jiki/article/view/366