Evolutionary Convolutional Neural Network: A Case Study

Amanda Lucas Pereira*, Manoela Kohler*, Marco Aurélio C. Pacheco*
*Department of Electrical Engineering
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil

***Abstract*—Most of the state-of-the-art Convolutional Neural Network (CNN) architectures are manually crafted by experts, usually with background knowledge from extent working experience in this research field. Therefore, this manner of designing CNNs is highly limited and many approaches have been developed to try to make this procedure more automatic. This paper presents a case study in tackling the architecture search problem by using a Genetic Algorithm (GA) to optimize an existing CNN Architecture. The proposed methodology uses VGG-16 convolutional blocks as its building blocks and each individual from the GA corresponds to a possible model built from these blocks with varying filter sizes, keeping fixed the original network architecture connections. The selection of the fittest individuals are done according to their weighted F1-Score when training from scratch on the available data. To evaluate the best individual found from the proposed methodology, the performance is compared to a VGG-16 model trained from scratch on the same data.**

I. Introduction

Convolutional Neural Networks (CNNs) have outperformed other kinds of neural networks in a number of real-world applications, such as image classification [1], image segmentation [2] and many others. The foundation for a good performance with a CNN can be roughly summerized in three topics: data availability, hyperparemeters tuning and its design. This latter is usually attacked by trial and error and for most state-of-the-art CNNs it is highly dependable on specialists' expertise in the field.

A number of new CNNs which are outperforming previous benchmarks have their architecture design either based on its precedents [3] or in combining existing ones [4]. Another approach that has been proposed as a way of designing new CNNs consists in applying search methods to optimize the architecture, such as Genetic Algorithms [5] and Q-Learning [6]. This paper studies the methodology of taking an existing architecture and trying to find an optimal version of it using a Genetic Algorithm, choosing by activating or deactivating some of its convolutional blocks and trying different filter sizes for each one of them.

This paper is organized as follows. Firstly, background concepts and related work are presented in Section II. Then, in Section III, the details of the proposed methodology are introduced. In Section IV, the results are exposed and discussed. Finally, in Section VI, conclusions and future works are outlined.

II. Literature Review

A. Genetic Algorithms

Genetic Algortihms (GA) is a family of search heuristics inspired by natural evolution [7]. In this context, a proposed solution for the problem such algorithm is trying to solve is called an individual, each represented by an encoded array of bits or strings which is called a chromosome [8]. A set of individuals is called population, and it may change at every iteration of the algorithm, aiming at achieving better individuals to form a generation.

Similarly to natural evolution, the optimization process consists of choosing amongst a set of solutions – the population – the fittest ones to breed or to continue into a newer set of solutions, which constitutes a new generation. This process is repeated until a certain convergence criteria is reached, which can be set as a minimum performance required from the individuals or a maximum number of generations reached during execution [9] (Figure 1).

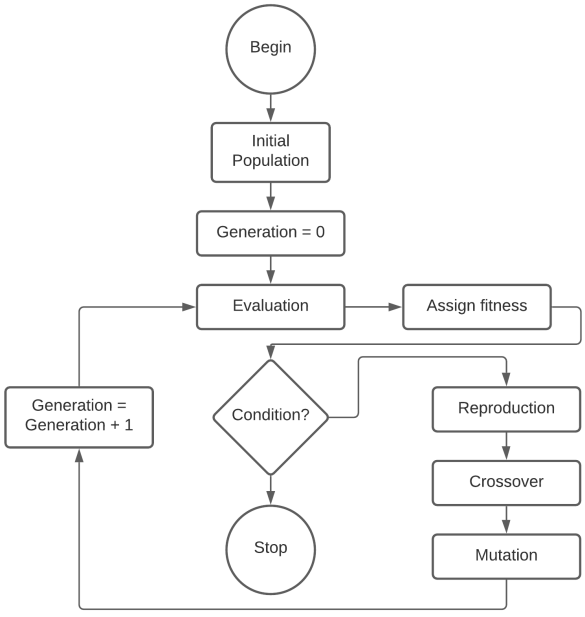

Fig. 1: The optimization loop of Genetic Algorithms.

1) Fitness Function: The process of selecting the best individuals from a given generation is done by evaluating each one of them using a fitness function, which can be described as a way of measuring how good the solutions presented are at solving the given problem [10]. The definition of a fitness function depends on the problem being solved, and should be chosen carefully with respect to the objective of the optimization process, which is finding the best solution.

2) Genetic Algorithm Operators: Genetic Algorithms start their search from an initial population and in order to make individuals evolve into the best solution it makes use of operators. There are three main types of operators: selection, mutation and crossover [7].

Similarly to natural selection, the selection operator consists of choosing the fittest individuals amongst a generation. Here, the fittest individuals represent the individuals with the highest evaluation – the highest return value from the fitness function. Selection can be performed by various methods, such as fitness-proportional, linear ranking and tournament. On selection by tournament, firstly two individuals are randomly chosen from the population and then evaluated by their fitness function return value. This process is done k times, and all winners from the tournament stage are selected for breeding.

Crossover, also called recombination, combines the genetic information of two individuals – parents – to generate new offspring. The simplest form of crossover is the single-point or 1-point crossover. This kind of crossover randomly selects a crossover point with probability p_c and then swipes the bits from both parents, resulting in two offspring with different segments of the parents' chromosomes (Figure 2). Since it provides a way of mixing the genetic information of the fittest solutions from a generation, this operator contributes for the convergence of the algorithm [11].

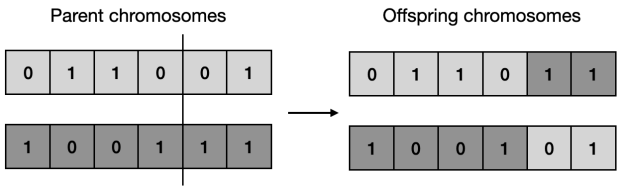

Fig. 2: One-point Crossover Operator.

The mutation operator helps keeping diversity in the solutions as the algorithm advances in the number of generations, avoiding convergence to a local optimum [8]. The operation consists in randomly choosing one bit from the individual with probability p_m – usually smaller than p_c – and changing its value (Figure 3).

B. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) refer to a variety of neural networks that are able to process data that are stored in matrix shapes, such as images. Such as others neural networks, its basic architecture consists of an input layer, a set of hidden layers and an output layer. The main aspect that separates

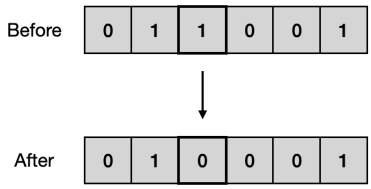

Fig. 3: Mutation Operator

these networks into its own category it is the fact that a CNN perform a convolution operation in at least one of its layers [12]. Figure 4 shows an example of a CNN, with three convolutional layers.

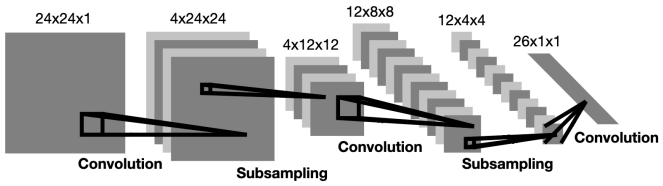

Fig. 4: An example Convolutional Neural Network, used for handwritten characters recognition. Adapted from [13]

Except for the input layer, each layer is composed of a block that performs some kind of mathematical operation on the input by applying filters and passes the obtained output through an activation function, yielding a feature map [14]. A challenge that comes from using CNNs is that given the highly computational demanding training, such networks are costly to optimize in what concerns time and resources. Adjustments performed either on its training parameters or architecture would require new training, validation and testing stages. Researchers stumbling with this problem promoted the growing field of automatic neural network architecture's design and optimization [15].

C. Neural Networks Architecture Design

Some applications of neural networks design using Genetic Algorithms focus not only on optimizing the architecture but also in selecting the most significant features from data [16], or applying GA to various machine learning problems, not only deep neural networks [17].

In Genetic CNN [5], the authors propose an encoding method that consists in multiple stages, using binary strings to represent network architectures. The number of stages – depth of the network – is predefined, and the head of the network is also not optimized. Starting from an initial CNN framework, the process consists in replacing its convolutional layers by smaller architectures to construct the final design. The parameters of the building blocks that form these smaller architectures are fixed, and only the combination of their connections are optimized. During the optimization process, each individual is evaluated by training the proposed network architecture from scratch on MNIST [18] and CIFAR10 [19] datasets, separately

in two experiments. These two datasets were chosen due to the computational demanding characteristic of genetic algorithms.

At each stage of the network, several building blocks are stacked with their connections encoded. These connections, as seen on other CNNs, are done by some kind of spatial pooling that may not preserve the dimensions of the feature maps and therefore are only permitted in an ordered manner. The main drawback from this method is the use of predefined parameters that must be set by the user of the framework, which relies on expertise and familiarity with CNNs. Additionally, another drawback could be that the framework itself is constrained so the output architecture is limited to some specifications.

A different method is proposed in [20], based on the Genetic CNN approach. On this work, the authors attempt to produce a more automatic framework that would not require as much prior knowledge of CNNs parameters settings or architecture as the previous ones. The algorithm demands the user to define a set of building blocks, the maximum number of generations for the genetic algorithm and the dataset used for evaluating the proposed architectures. A variable-length encoding strategy along with a variable crossover operator is initialized with the first population, so the depth of the network does not need to be set beforehand. Also, this proposed methodology allows for skip-connections and makes use of asynchronous computing to accelerate the evaluation of the individuals.

The authors from Nas-Unet [21] propose an optimization based on the structure of an existing architecture known to perform successfully on segmentation tasks: U-Net [22]. As opposed to the approach of optimizing the network connections seen on Genetic CNN, Nas-Unet applies the prior knowledge on network architectures in a different way, by fixing the network U-like backbone network instead of fixing the parameters within the building blocks. With predefined network connections, the search process focuses only on finding the optimum version of the two kinds of building blocks that are supposed to fit onto the architecture: DownSC and UpSC. The architectures of DownSC and UpSC are optimized simultaneously through a more efficient version of an over-parametrized network [23].

Similarly to Nas-Unet, the present study tries to optimize the neural network architecture with respect only to its building blocks, keeping the original structure of VGG-16, known to perform well on image classification tasks. Also, the search is done using a Genetic Algorithm instead of the method used by the authors.

III. Proposed Methodology

A. Data

The original data available for the experimentation part of this work comprises a total of 24888 environmental images, each identified by one out of five different labels. Table I shows the number of samples available for each class in the original dataset. Due to GA being computationally expensive, the results presented were obtained using a subset of the original data, ensuing a much smaller set of 1250 images. Since the data is highly unbalanced, the images for the subset were chosen by undersampling the major classes so that the final dataset presents an equal distribution of samples per class.

TABLE I: Class Distribution - Original Data

Class	Samples	Percentage
Class 0	469	1.8%
Class 1	8351	33.5%
Class 2	2675	10.7 %
Class 3	12154	48.8%
Class 4	1239	4.9%

After sampling, the data was shuffled and split into training, validation and tests sets. For the first one, we used 70% of the images, for validation 20% and the remaining 10% for testing. The same splits were used in each individual's evaluation for every generation in the optimization process.

B. CNN Framework

For this work, a VGG-16 [24] network was used as the baseline framework for testing the presented methodology. The architecture for this network is composed of a stack of convolutions with a 3x3 kernel with stride of 1 followed by a max-pooling layer with a kernel of size 2 and stride of 2. At the end, there are two fully connected layers with softmax and a global average pooling layer (Figure 5).

The filter sizes used by the original architecture for each block are 64, 128, 256, 512 and 512. On this paper, a *convolutional block* refers to a group of N convolutional layers followed by a max-pooling layer. For instance, the first convolutional block of VGG-16 consists of two convolutional layers with a filter size of 64 and a max-pooling layer that downsamples the output into the shape of 112×112×128 (Figure 5). If the first layer is included in an individual's proposed architecture, it means the first layer of such network will also be composed by two convolutional layers and a max-pooling layer, but with a varying filter size.

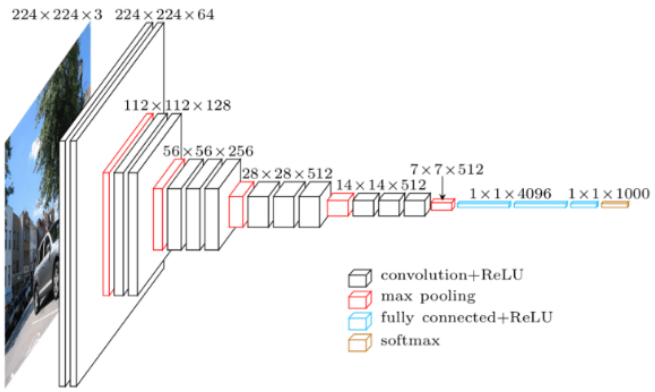

Fig. 5: VGG-16 Architecture. Source: [25]

During training, the original VGG-16 network expects a fixed-size 224×224 RGB image as an input. For this work, this fixed-size input was maintained, and the input data was reshaped as required.

C. Chromosome Encoding

For each individual, the proposed method provides a binary string representation that decodes into a network architecture which is a candidate solution to the optimization problem. Based on the convolutional blocks of VGG-16, there are a total of five blocks available to be combined for each individual, keeping the original order used in VGG-16. There is a variety of seven different possible filter sizes the algorithm can decode the bit string into, which is the array: 32, 64, 128, 256, 512, 1024, 2048.

TABLE II: Chromosome

0	Use block1
1	Use block2
2	Use block3
3	Use block4
4	Use block5
5 - 7	Filter Size (block1)
8 - 10	Filter Size (block2)
11 - 13	Filter Size (block3)
14 - 16	Filter Size (block4)
17 - 19	Filter Size (block5)

Table II shows the encoding structure of the chromosome. Positions 0 to 4 are 1-bit binary variables that define whether or not a convolutional block will be part of a given individual (model), and positions 5 to 19 are 3-bit binary variables that represent the filter size used on the convolutional layers.

The bits with respect to the filter sizes are decoded into the position of a predefined array of possible filter sizes. For example, the string "011" decodes into the second position of the array, which corresponds to a filter size of 64. The top fully-connected layers were not included in the optimization process, so they are not included in the chromosome.

If *block1* is activated, the string formed by the bits on the positions 5, 6, 7 will dictate the size of the filter. Consider the individual "11010-011-001-110-011-011". The network architecture proposed by this one will use the first two and the fourth convolutional blocks, a filter size of 64 for the first layer, of 32 for the second one and 64 for the fourth one. The layers that are not used for this configuration will have the filter sizes ignored by the function. This individual would decode into the architecture shown in Figure 6.

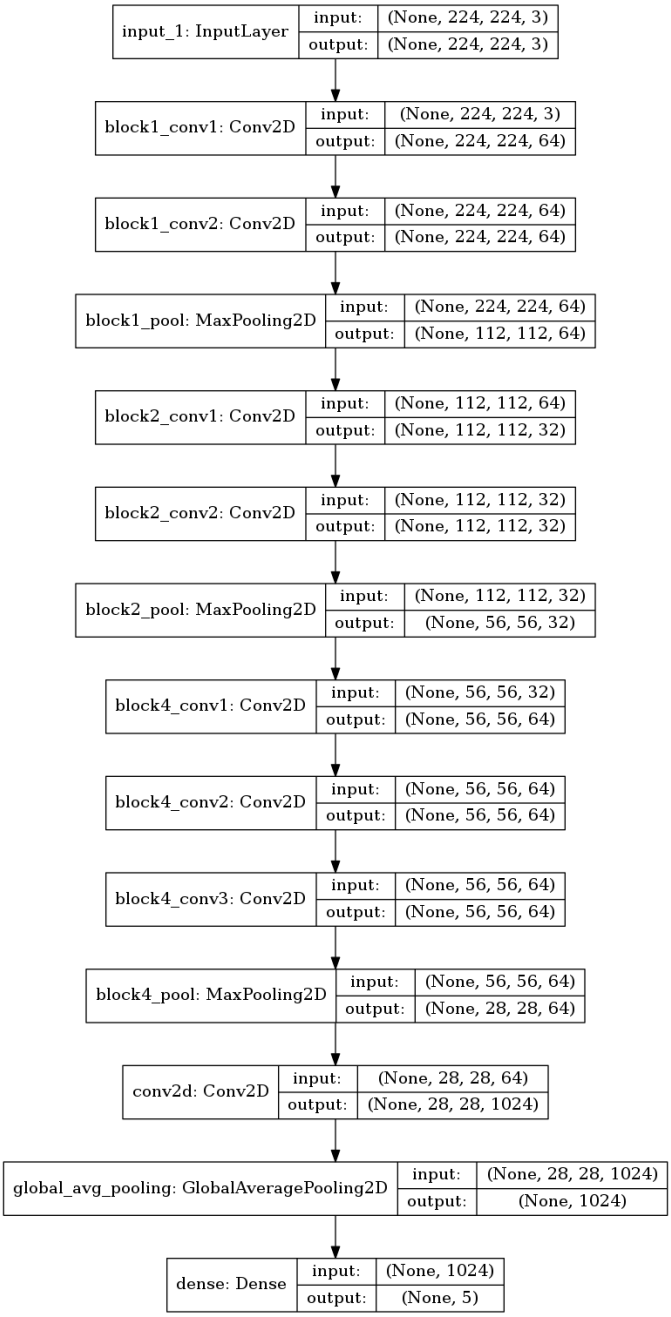

Fig. 6: Architecture proposed by the individual "11010-011-001-110-011-011".

D. Fitness Function

The fitness function was defined as a function that firstly build the model given the input individual and then run the training stage from scratch, followed by a validation and test stages. Training was done using back-propagation algorithm and a fixed batch size of 16, and the return value of the fitness function was set to be the weighted average F1-Score obtained from the test stage.

When calculating the weighted average F1-Score, initially the F1-Score is calculated for each label and then averaged with the weight given by the number of true instances for each label. With a total of N samples, being N_i the number of samples for label i and $F1_i$ the F1-Score for label i the weighted F1-score is given by equation 1.

$$Weighted - F1 = \frac{1}{N} \sum N_i * F1_i \tag{1}$$

E. GA Operators

The individuals were ranked and selected according to their evaluation by the fitness function explained in subsection

D. The crossover operator used for mating was the one-point crossover and mutation was performed with probability $p_m = 0.05$. The population was initialized randomly with 10 individuals per generation, and the algorithm was executed for a total of 20 generations. Selection was done by tournament, as the method described on Section II.

IV. Results and Discussion

A. Base CNN Architecture

In order to evaluate the results obtained by the optimum individual found by GA, the same data was used to train a VGG-16 model from scratch. The training parameters were the same as the ones used to evaluate the model architeture when running the GA algorithm, which corresponds to a batch size of 16, learning rate of 0.0001, maximum of 100 epochs, input size of 224 and early stopping with a patience of 10 epochs. Categorical Loss Entropy was used during training, with predefined weights of $5.5, 0.5, 6, 1.8, 1.2$ for class $0, 1, 2, 3$ and 4 respectively. Adam optimization was used as the gradient descent method, with an exponential decay rate for the first momentum set to 0.9.

TABLE III: Results - Baseline

	F1-Score
Class 0	46.15
Class 1	0.00
Class 2	58.18
Class 3	0.00
Class 4	81.69

	Weighted Average
F1-Score	40.90
Recall	50.80
Precision	36.24

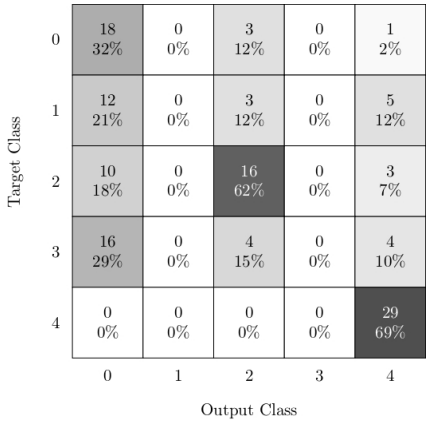

Fig. 7: Confusion matrix for the baseline model.

The results for the testing stage of the baseline model are presented in Table III and Figure 7. From both F1-Score values per class and the confusion matrix, we see that there were no correct classifications for classes 1 and 3. Most of the samples from these classes were labeled by the model as class 0, which might point some similarity between such classes.

B. Best individual from GA

The best individual obtained at the end of the genetic algorithm optimization process is given by "00011-011-110-111-011-110" (Figure 8). The optimum solution proposes a VGG-16 based network with only the fourth and fifth blocks, with a filter size of 64 and 256, respectively. The metrics obtained by the model constructed from decoding this string are shown in Table IV.

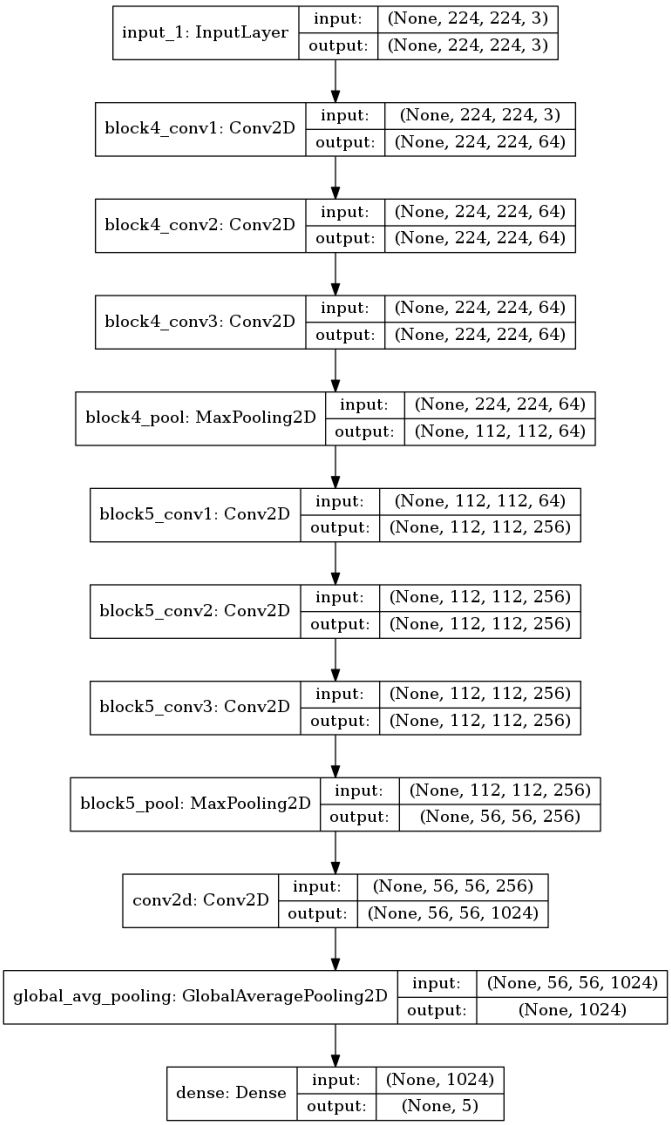

Fig. 8: Network architecture proposed by the best individual.

The F1-score worsened for all the 5 classes and one can notice a shift from the data misclassified as class 0 to being classified as class 2 (Figure 9), which opposes the hypothesis that the baseline network labeled most samples as class 0 due to similarity between classes 0, 1 and 3. The F1-Score for classes 1 and 3 remain 0, as no samples from these classes were correctly classified by the network.

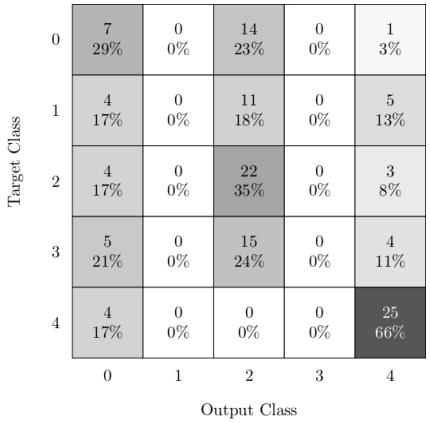

Fig. 9: Confusion matrix for the model proposed by the best individual.

TABLE IV: Results - Best Individual

	F1-Score
Class 0	30.43
Class 1	x 0.00
Class 2	48.35
Class 3	0.00
Class 4	74.63

	Weighted Average
F1-Score	34.16
Recall	43.54
Precision	28.85

V. Conclusions and Future Work

This study has shown the feasibility of optimizing existing manually-factured network architecture designs with Genetic Algorithms. Although the performance of the best individual found at the optimization procedure does not surpass the baseline performance of the VGG-16 network, some improvements can highly impact the GA final result.

On the behalf that the optimization process using GA takes a relatively large amount of time to complete, the experiments done for this work had to be limited and it would be interesting to hold experiments with the complete dataset.

Additionally, it would be of good effect to try different GA parameters such as the number of generations and population size. There is also ground for implementing improvements in the fitness function, accounting for robusticity of its results and computational efficiency. Moreover, setting the starting seed of the optimization process as the baseline VGG-16 architecture could help the algorithm converge to superior results.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[4] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2018, pp. 3–11.

[5] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[6] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Blockqnn: Efficient block-wise neural network architecture generation," 2018.

[7] D. A. Coley, *An introduction to genetic algorithms for scientists and engineers*. World Scientific Publishing Company, 1999.

[8] X.-S. Yang, *Nature-inspired optimization algorithms*. Academic Press, 2020.

[9] S. Sen, "Chapter 4 - a survey of intrusion detection systems using evolutionary computation," in *Bio-Inspired Computation in Telecommunications*, X.-S. Yang, S. F. Chien, and T. O. Ting, Eds. Boston: Morgan Kaufmann, 2015, pp. 73–94. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128015384000045

[10] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[11] A. J. Umbarkar and P. D. Sheth, "Crossover operators in genetic algorithms: a review." *ICTACT journal on soft computing*, vol. 6, no. 1, 2015.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[14] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc Buffalo NY, Tech. Rep., 1961.

[15] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," 2019.

[16] K. M. Hamdia, X. Zhuang, and T. Rabczuk, "An efficient optimization approach for designing machine learning models based on genetic algorithm," *Neural Computing and Applications*, vol. 33, no. 6, pp. 1923–1933, 2021.

[17] Y. Yuan, W. Wang, G. M. Coghill, and W. Pang, "A novel genetic algorithm with hierarchical evaluation strategy for hyperparameter optimisation of graph neural networks," *arXiv preprint arXiv:2101.09300*, 2021.

[18] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[19] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[20] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[21] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "Nas-unet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44 247–44 257, 2019.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[23] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eYHoC5FX

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[25] T. Sugata and C. Yang, "Leaf app: Leaf recognition with deep convolutional neural networks," *IOP Conference Series: Materials Science and Engineering*, vol. 273, p. 012004, 11 2017.