2021 International Conference on Data Mining Workshops (ICDMW) | 978-1-6654-2427-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDMW53433.2021.00130

Neural Architecture Search and Multi-Objective Evolutionary Algorithms for Anomaly Detection

Chakkrit Termritthikun
School of Renewable Energy and Smart Grid Technology (SGtech)
Naresuan University
Phitsanulok, Thailand
chakkritt60@nu.ac.th

Lin Xu
STEM
University of South Australia
Adelaide, Australia
xuyly032@mymail.unisa.edu.au

Yemeng Liu
School of Software
Dalian University of Technology
Dalian, China
liuziweiww@outlook.com

Ivan Lee
STEM
University of South Australia
Adelaide, Australia
Ivan.Lee@unisa.edu.au

Abstract*—The processing speed and memory footprint are important factors for applications processing on resource-constrained devices such as IoT devices and embedded systems. Deep learning has been evolving continuously so that it can be used on resource-constrained devices but there are still some limitations in using it because these devices are not designed for processing complicated tasks. Further, the complexity of the Convolutional Neural Network (CNN) model is the barrier to implementation on these devices. In this paper, we have developed Neural Architecture Search (NAS) that uses a Multi-Objective Genetic Algorithm and Firefly Algorithm for creating a less complicated and robust CNN model, focusing on searching the model with faster processing time and minimum storage size. Five image datasets are applied to examine the performance of the proposed techniques, including two crack detection datasets for surface or built infrastructure inspection for industrial applications. Experimental results show that the proposed technique consistently lowers the parameter counts and processing time at comparable classification accuracies.

***Keywords*—Deep Learning, Neural Architecture Search, Multi-Objective, Genetic Algorithm, and Firefly Algorithm**

I. Introduction

The rapid development of artificial intelligence over the past several years has facilities many meaningful and valuable research. Anomaly detection is a typical application that has been applied in many fields, such as medical imaging [1], health care [2], image processing [3], network intrusion detection [4], fraud detection, and video surveillance [5] [6].

Generally speaking, anomaly detection aims to distinguish abnormal samples from normal samples. It seems like an ordinary dichotomy problem, however, in fact, anomaly detection has an inherent attribute that is the samples are extremely unbalanced. In some extreme cases, no abnormal sample exists in the training dataset [7]. Thus, the mainstream research directions can be divided into two categories.

The first category is based on normal examples or even only normal examples. In these cases, anomaly detection problems set the models which construct the normal samples distribution first, and then through these models to achieve abnormal samples detection [8][9]. For network analysis, graph-based approach can be utilized [99]. For image-based analysis, Generative Adversarial Networks (GAN) could commendably handle the unbalanced samples of anomaly detection problems [10][11].

Among them, the design of GANomaly [8] is the most representative and most influential GAN-based anomaly detection in the past a few years. The authors argue that there are a few abnormal samples in anomaly detection usually, so it is difficult to provide a model which can distinguish abnormal samples directly. Since it is not reliable to detect abnormal samples, this paper takes the opposite approach. It only trains the normal samples and if the sample in the test set does not look like the normal one, it will be distinguished as abnormal. Vary from the original GAN [12], GANomaly uses mutated GAN and auto-encoder network. It improves generator network to encoder →decoder →encoder which optimized the network to a large extent, such as gradient descent. And auto-encoder is a classical method in anomaly detection. Its solution is to use as many normal samples as possible to train a self-coding model. As a result, it can reconstruct the normal sample well. However, the abnormal samples cannot be reconstructed well because there are only a few abnormal samples in the training dataset, or even more, there is not any abnormal samples in the training set. As a result, it is not difficult to distinguish the normal and abnormal samples by imputing the reconstruction error of the image. However, this method is very susceptible to noises. So if we

978-1-6654-2427-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDMW53433.2021.00130

require a usable anomaly detection model, it needs to add various constraints on the encoder.

Auto-GAN [13] is the first attempt of using Network Architecture Search (NAS) [14] [15] in GAN. Network architecture search has achieved some successes in image classification and other tasks. However, most existing NAS methods exhibit a high computational complexity, which requires not only expensive hardware but also a lot of processing time. Sirui et.al. discuss this problem in 2019 [16] and suppose that introducing too much deviation in the approximation of the objective function of NAS is the main reason why NAS needs two stages optimization of parameters. As a result, they provide a solution to ensure one stage optimization through a stricter approximation. But this method is not applicable for large-scale network search tasks. Therefore, in order to solve this problem, they propose the DSNAS [17] method which has a differentiable searching architecture and could optimize the structure parameters and the network parameters at the same time.

The second category, as opposed to the first, considers scenarios with sufficient anomaly (defect) samples for training. In these cases, anomaly detection can also be defined as a classification task. Sometimes, if the detection area is too large and the detect location is relatively small, we can subdivide the area into small ones, and classification can be conducted through a sliding window to cover every sub-area.

Nevertheless, both approaches face the same limitation about parameters selection. It is known that there have many challenges on defining a set of appropriate parameters which results a good robust in the experiments. Those parameters include (but not limited) the kernel size, scroll position of the filter, the size of output, zero padding and so on. The number of parameters and super-parameters increase greatly with the neural network designed more and more complex. Although, the structure of neural network could be designed manually [18] [19] and the selection of parameters and super-parameters in the network is based on artificial experiences, it often not necessarily the optimal choice and always time-consuming because we need to try repeatedly.

Neural Architecture Search (NAS) [21] aims to find the most suitable set of parameters and appropriate network structures for a particular task. This method does not require following the previous designed structures and do not rely on human expertise. So, it optimizes the architecture of the neural network tremendously and makes designing the most robust model become possible. Another improvement of NAS is that the architecture of it is flexible enough, so it is suitable for both large and small datasets. Search space is one of the core elements in the NASNet, which defines a set of neural network structures (cells) that can be searched. Fig.1 shows an example of the NASNet. Compared with normal architecture (left), the NASNet has not only one kind of input (right).

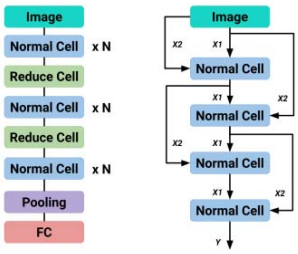


Fig. 1. NASNet Search Space.

Barret [15] made the number and size of filters in each layer into an optimization problem which needs to be determined manually before. The network can choose the best combination of these parameters by itself. Then the authors optimized the algorithm [22] that they just search in some special cells, normal cell and reduction cell used in down-sample, instead of searching in the whole networks and have gotten a higher speed and better performance. However, the limitation of this method is that its high computational demand which consumes a lot of time. Although, the authors have continue to improve the algorithms [23] [24], it still needs a few hundred of GPU-days. So some solutions that can speed up this progress have been proposed, for example, wight sharing [25] and differentiable equation [26], which greatly reduce the requirement of hardware and time.

However, the most existing NAS methods requires not only expensive hardware but also a lot of time. For example, it takes up to 2,000 GPU days to complete the task that using reinforcement learning (RL) based on CIFAR-10 dataset [22]. While AmoebaNet [27] takes 3,150 GPU days to find the best model using evolutionary algorithm and based on the same dataset.

Furthermore, those algorithms are unable to run on small devices because they require excessive processing time. As a result, it is necessary to simplify the architecture of NAS network and speed up the research rate. There has been a lot of remarkable works of this problem, including the SMBO method (sequential model-based optimization) [24], desired structure definition method [28], weight-sharing [25], and using DARTS model [26]. Sirui et.al. discuss this problem in 2019 [16] and suppose that introducing too much deviation in the approximation of the objective function of NAS is the main reason why NAS needs two stages optimization of parameters. As a result, they provide a solution to ensure one stage optimization through more strict approximation. But this method is not applicable for large-scale network search tasks. Therefore, in order to solve this problem, they propose the DSNAS [17] method which has a differentiable searching architecture and could optimize the structure parameters and the network parameters at the same time.

In this paper, a Multi-Objective Genetic Algorithm Network model (MOGA-Net) is proposed to reduce the search time and find the best neural network structure that most suitable for a given dataset. Our approach follows a typical process: first of all, the structure and content of target dataset has been analyzed and identified. The architecture of the model has been designed during the process of training set while the effectiveness of it has been measured based on the testing dataset. The above process, that are analyzing, developing, and measuring, will be iterative until the best model has been selected for the target dataset. This best architecture of the model plays high efficiency and requires small storage space, so that it could be run on the devices with insufficient processing capacity.

The contributions of this paper include the design of MOGA-Net and FA-Net, by applying the Multi-Objective Genetic Algorithm and the Firefly Algorithm to the NAS network, respectively. The proposed techniques are examined using a crack image dataset, and further experiments are conducted on other classification dataset to examine the robustness across different applications.

This paper is organized as follows. Section II describes the methodology of this paper. The experiments and results are

presented in Section III, while Section IV presents the concluding remarks of this paper.

II. METHOD

A. Multi-Objective Genetic Algorithm

Genetic algorithm (GA) [29] is originated from the computer simulation of biological systems. It is a method of random global search and optimization developed by imitating the biological evolution mechanism of nature, which draws lessons from both Darwin's evolution theory and Mendel's genetic theory. GA is an efficient method for finding the optimal global solution using a parallel search technique, which could automatically acquire and accumulate knowledge about the search space during the process of searching. The search process is controlled adaptively while the best solution is obtained eventually.

Genetic algorithm evolves from the initialize population which is the potential solution set of the research problem. Here a population is composed of a certain number of individuals encoded by genes. Each individual is an entity with characteristic chromosomes. The chromosome is the main carrier of genetic material, that is, the collection of multiple genes. Its internal performance (i.e. genotype) is a certain gene combination, which determines some individual characteristics that we call it phenotype. For example, the color of hair is determined by a certain gene combination in the chromosome that controls this characteristic performance. Therefore, it is necessary to realize the mapping from phenotype to genotype, which can be defined as the binary coding. In general, the problem to be solved in genetic algorithm will be mapped into a mathematical problem, a feasible solution of which is called a chromosome. This feasible solution is generally composed of multiple elements, and each element is defined as a gene on this chromosome.

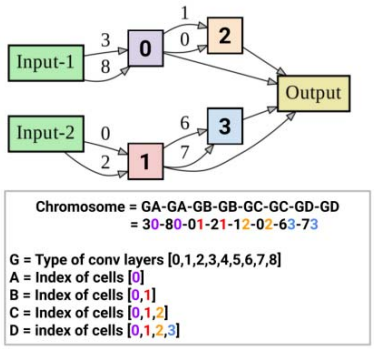


Fig. 2. The chromosome structure of the Multi-Objective Genetic Algorithm and Firefly Algorithm.

In this paper, genotype is defined as the architecture of the model. The complete structure of the model consists of the cells which searching from the search space of the NAS. Each cell merges by several layers, such as max pooling layer, average pooling layer and convolutional layer. These cells form a model genotype by interrelated ways. This is an example shown in Fig.2. In this model, the number of genes is defined as even and must meet eight types, such as GA-GA, GB-GB, GC-GC and GD-GD. Each gene composes by an operation (G) and an index (A, B, C and D). The operation set in this example defined as 9 different kinds of operations, including max pooling, average pooling and convolution, which are represented by [0,1,2,3,4,5,6,7,8]. The location of operations is represented by the indexes (A, B, C and D). Every operation needs to be connected to an index which can be specified as A=[0], B=[0,1], C=[0,1,2] and D=[0,1,2,3]. Such design allows forming various network through NAS, including networks with simple or complex structures [20].

It is necessary to use a fitness function to measure the quality of the solution for every chromosome. When the initialize populations are defined, the better and better approximate solutions are evolved by the intergenerational changes, which should according to the principle that is the survival of the fittest. In other words, in each generation of evolution, individuals need to be selected or eliminated by the fitness of the problem domain. At the same time, it would cross and mutate with the reference of genetic operations of natural genetics. In this paper, the architecture of model will be searched based on accuracy, so the fitness function will be defined as Eq. (1), which uses the accuracy obtained from the test database.

$$fitness(i) = f(X_i), \qquad i = 1,2,3 \dots, n \tag{1}$$

where n is the number of individuals. In this process, every generation will be more adaptable to the environment than the previous one. So, the optimal individual in the last generation population will be considered as the approximate optimal solution of the problem. In recent years, the genetic algorithms have been gradually extended to solve the multi-objective optimization problems. In this paper, training errors, calculation cost and parameters size are the three core indexes that we should be considered when searching the architecture of the neural network model. As a result, this is a typical multi-objective optimization problem. We defined the objective function as Eq. (2), where f is the fitness value, the integer represents the number of objectives and X is the set of individuals. The target of Eq. (2) is the fitness of all individuals must be as small as possible, thus the algorithm is known as the Multi-Objective Genetic Algorithm (MOGA).

$$\begin{aligned} &\text{minimize}\big(f_1(x), f_2(x), \dots, f_k(x)\big) \\ &\text{subject to } x \in X \end{aligned} \tag{2}$$

Crossover and mutation are the main approaches to produce new solution sets in the process of evolution. Here, crossover requires two chromosomes in the previous generation, and the new chromosomes are created by cutting and splicing chromosomes of the previous ones. Crossover can be carried out in a variety of ways, such as multi-point crossover, arithmetic crossover and uniform crossover (as shown in Fig.3). The multi-point crossover means set crossing points in chromosomes randomly and exchange some chromosomes of two paired individuals at the crossing points. The arithmetic crossover is defined as a linear combination of two chromosomes and two new individuals are produced by it. In general, the operation objects of arithmetic crossover are individuals which are represented by floating-point number coding. The uniform crossover exchanges every gene on two paired chromosomes with the same probability, and the two new individuals form from this way.

Mutation refers to replacing the gene values at some position in the chromosome with other alleles and then the new chromosome is formed. Mutation breaks through the limitation of the search, so it is more easily to find the global optimal solution for the algorithms.

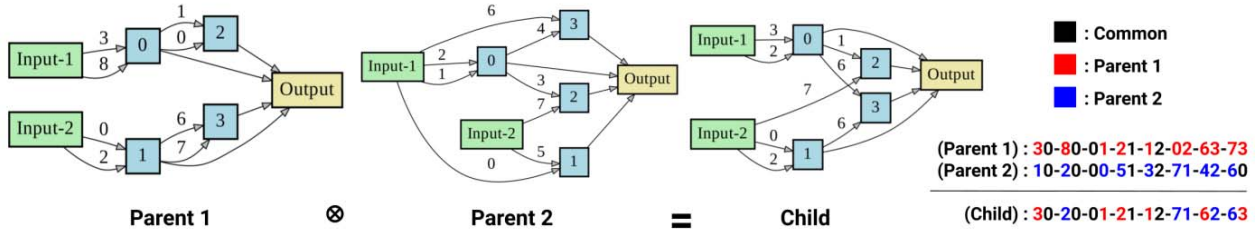


Fig. 3. Crossover Example

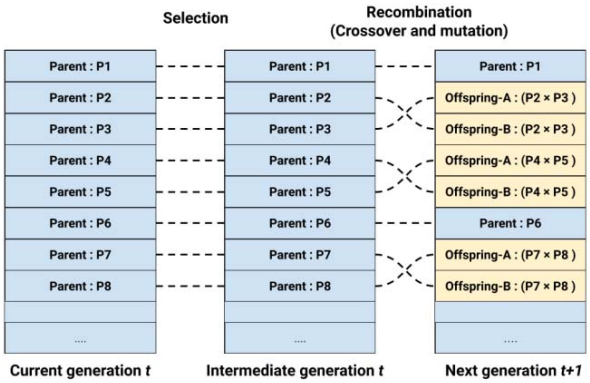


Fig. 4. Selection and recombination of the Genetic Algorithm.

B. *Firefly Algorithm*

When dealing with complex optimization problems, swarm intelligence algorithm (SIA) often shows an efficient performance. SIA is an algorithm proposed to imitate the group behavior of animals in nature. These animals have simple individual behavior, but can accomplish complex work through the cooperation between groups. The firefly algorithm (FA) is a representative example of SIA [30], which is inspired by the behavior of fireflies' glow to attract mates in nature. In nature, fireflies attract mates by making light, and weaker fireflies are attracted to stronger fireflies and move closer to them.

Based on this behavior, the firefly is represented by the points in the optimization problem's definition domain, and the luminous intensity of the firefly is represented by the fitness value of the points. Fireflies continually change their position by moving toward fireflies that emit lighter than they do. The distance of firefly movement was determined by the attraction between the two fireflies, and the degree of attraction was determined by the distance between the two fireflies. In the process of movement, the corresponding fitness value will change if the firefly position changes. Therefore, it is necessary to recalculate the fitness value of the firefly after each movement to evaluate the advantages and disadvantages of the firefly position. Firefly algorithm is to repeat the above motion to complete the search of the solution space of the optimization problem. In addition, in the firefly algorithm, there is no gender division between fireflies, that is, there is an attract and attracted relationship between any two fireflies in the population, and this attract and attracted relationship is determined by the size of the adaptation value of the two fireflies.

The simplicity and efficiency of FA making it popular for solving many engineering and scientific problems, such as resource allocation [31], medical diagnosis [32], equation solving [33]. Various extensions have been introduced on top of standard FA, such as the work by Lv et al that uses elite learning strategies to provide fireflies with better learning objects [34].

To illustrate the procedure of the FA algorithm, suppose that the dimension of the optimization problem is D, the position of firefly i is represented by $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, and N fireflies constitute P_N of firefly population.

Definition 1: Brightness intensity of firefly:

$$I = I_0 e^{-\gamma r_{ij}^2} \tag{3}$$

In practice, the brightness intensity is determined by the fitness value of fireflies, that is, $I = f(X)$.

Definition 2: Attractiveness of firefly:

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \tag{4}$$

where β_0 is the attractiveness at $r = 0$. γ is light absorption coefficient, generally set as constant 1. r_{ij} is the Euclidean distance between the firefly individuals i and j, which is determined by the following equation.

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{jd})^2} \tag{5}$$

Definition 3: Position update of firefly i towards firefly j:

$$x_{id}(t+1) = x_{id}(t) + \beta(x_{jd}(t) - x_{id}(t)) + \alpha\varepsilon \tag{6}$$

where x_{id}, x_{jd} is the position of firefly x_{jd} and j in d dimension, α is the step size factor, which is the constant on $[0,1]$. ε is the random number subject to uniform distribution on $[-0.5,0.5]$.

C. *Search space with Genetic algorithm and Firefly algorithm*

This section describes the mechanism of adapting the concept of evolution algorithms to the NAS network. A set of structural models are chosen by applying the Multi-Objective Genetic Algorithm or the Firefly Algorithm, to form network structures that yields improved performance.

The types of layers that consist of the cell have been modified based on the method of NASNet beyond the definition of parameters. In addition, we replenish two inverted residual

layers of 3×3 and 5×5 to the search space. Because MNASNet mentioned that this method can effectively reduce the complexity of the operation and keep the accuracy of the model at the same time, which just like the depth-wise separable convolution [35]. The search space consists of several formats, emerged of which the best model of architecture may be found. The structural models used in this paper include:

TABLE 1. SEARCH SPACE

Kernel	Type
3x3	max pooling
3x3	average pooling
3x3	depthwise-separable conv
5x5	depthwise-separable conv
3x3	dilated convolution
5x5	dilated convolution
3x3	inverted residuals conv
5x5	inverted residuals conv
-	skip connection

According to the Multi-Objective Genetic Algorithm and Firefly Algorithm, it is necessary to evaluate the population in every generation and determine the fitness value on the target dataset. Therefore, the GPU time of this task depends on the total number of populations and generations. From this point of view, we define the "population" of every model of NAS. The random models (population) are created, and the searching space would consider the following points: the error rate, number of parameter and computational cost including Floating Point Operations Per Second (FLOPs). Those number of populations have been iterated by several generations and all the fitness values have been marked to be compared later. In the primary stage, we specify the number of processing generations for the purpose of practical operation. The fitness values of every generation could be compared with each other and find out the best one after we design the number of generations of all the populations.

According to the core idea of the NASNet search space, the architecture of the model is designed by the above cells following the method of Multi-Objective Genetic Algorithm and Firefly Algorithm to search for the optimal model based on the best population. The system adheres to the following evolved strategy: (1) in one generation, crossover the data between the different populations, and select one model with the highest accuracy. (2) mutate the population with the lowest error rate and create them as the next generation populations. (3) let the next generation populations as the current generation, which is considered with the lowest error rate. (4) crossover, select and mutate the current generation populations as the steps (1), (2) and (3) until the end of the last generation. The overall workflow is summarized in Fig. 5:

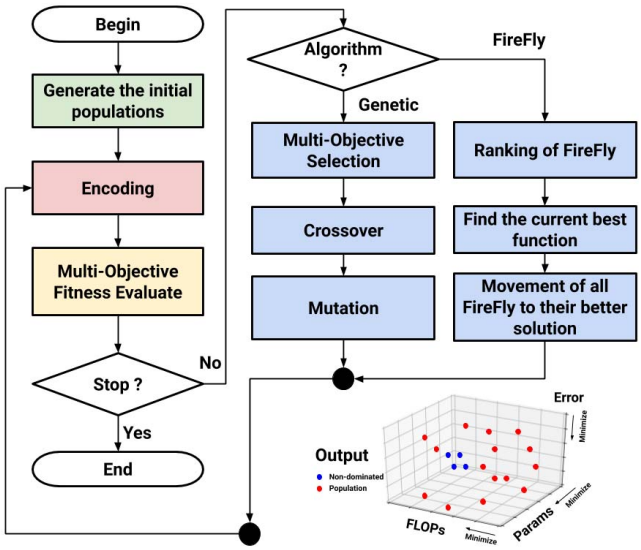


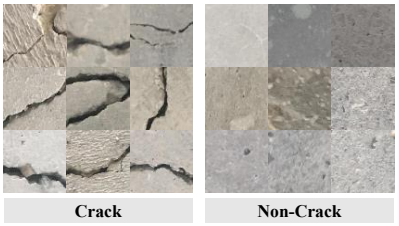
Fig. 5. Search space of Multi-Objective Genetic algorithm and Firefly algorithm.

III. EXPERIMENTS AND RESULTS

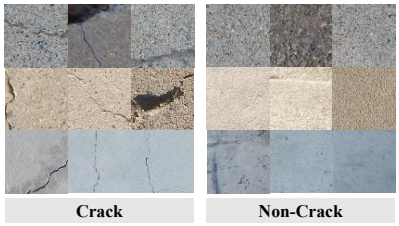
Experiments were conducted in two parts: First, determining the structure of the model with the Multi-Objective Genetic Algorithm and Firefly Algorithm running on the Intel(R) Xeon(R) W-3235 CPU @ 3.30GHz 12 Core CPU, 192 GB RAM and NVIDIA RTX 2080Ti GPU, with on the Ubuntu 18.04.3 operating system. The second part is the evaluation of the MOGA-Net and FA-Net using five selected datasets: Fashion-MNIST, CINIC-10, STL-10, Concrete Crack, and SDNET2018. The PyTorch deep learning library was used for implementing the proposed algorithms.

A. Datasets

The crack datasets used in this paper are Concrete Crack, and SDNET2018 as shown in Fig. 6. The images from these datasets can be of two categories: crack and non-crack. In Concrete Crack, there are 40,000 images with 20,000 images in each category. SDNET2018 dataset contains 56,092 images, with 8,484 crack images and 47,608 non-crack images. Further, these images are divided into training data and test data in an equal proportion of 50% each. Then, the images are resized to 112×112 pixels to suit into the CNN model.



(a) Concrete Crack dataset



(b) SDNET2018 dataset

Fig. 6. Sample of crack and non-crack images.

B. *Architecture Search on SDNET2018*

We defined 10 generations with 20 populations in each generation, giving to 200 populations, to find the model structure using the Multi-Objective Genetic Algorithm and Firefly Algorithm. The populations in the first generation were randomly populated, while the populations in generations 2-10 were evolved with the genetic algorithm or movement of the position of Firefly.

The hyper-parameters for the search process are defined as: the total number of cells (normal cells and reduce cells) is 4 layers with 8 initial channels, by training the network from scratch for 1 epoch on the SDNET2018 dataset with a batch size of 128, SGD optimizer with weight decay equal to 0.0003 and momentum equal to 0.9, the initial learning rate is 0.05 and uses the scheduler with the cosine rule, cutout regularization with length 16, and drop-path of probability is 0.2.

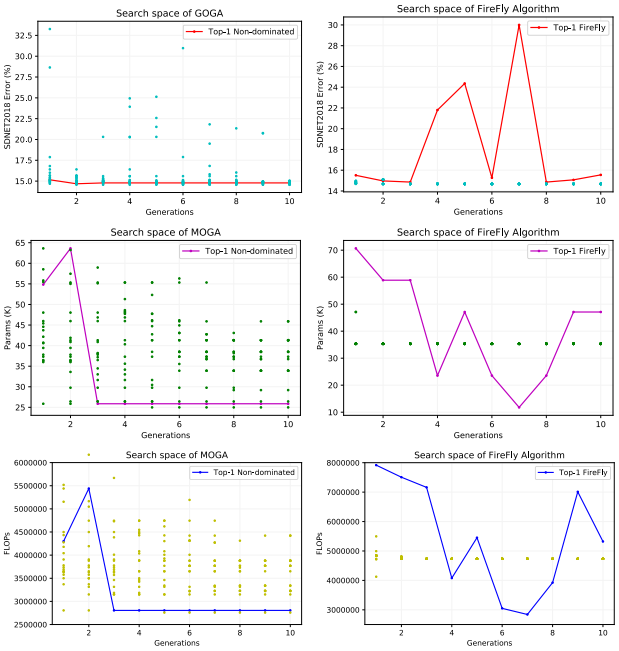


Fig. 7. Search space of MOGA-Net and FA-Net

According to the model search on the SDNET2018 dataset, MOGA finds an effective model across all performance measures including the error rate, the number of parameters, and FLOPs. From Fig.7 it is found that the MOGA can find the most effective model in each generation. However, Firefly Algorithm functions differently from MOGA because it adjusts every chromosome to the best performing chromosome of each generation. It has been observed that both algorithms have similar performance.

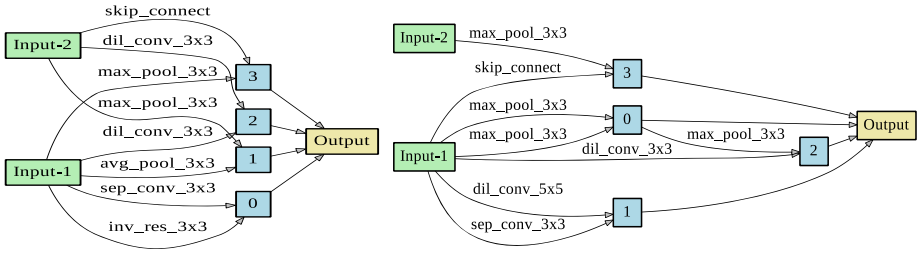
In contrast, the population of MOGA has a variety of chromosomes, because crossover is used in the emerging population and some are formed by mutations, thus causing diverse offspring of a chromosome. Also, we have used multi-objective ranking with the Firefly Algorithm population to find the appropriate population for each performance metric. The most suitable population of each generation will be defined as Top-1 Firefly for Firefly Algorithm and Top-1 Non-dominated for MOGA as shown in Fig.7.

The chromosome with good performance for each performance metric was selected from the last generation, in which the best chromosome obtained from MOGA is called MOGA-Net and the best chromosome from Firefly Algorithm as FA-Net as in Fig. 8.

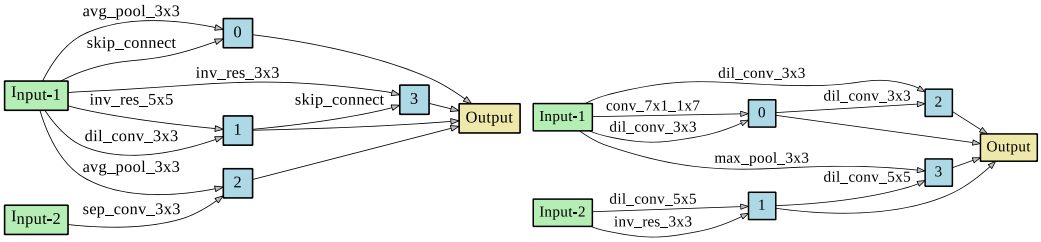
C. *Evaluation on Concrete Crack, and SDNET2018*

The same hyper-parameters as defined in the model search, except training the network from scratch is 100 epochs, were used for the evaluation of the model to determine the normal and reduce cells with MOGA-Net and FA-Net.

Table 2 presents a comparison between MOGA-Net and FA-Net with other state-of-the-art models. When considering the error performance, the PC-DARTS has the smallest error. However, the parameters of PC-DARTS is 24.8K more than the MOGA-Net parameters and 2.91K more than FA-Net parameters. The computational time of MOGA-Net is 2.45 MMac less than the PC-DARTs, while PC-DARTs overperforms the FA-Net in terms of computational time with a difference of 0.25 MMac.



(a) MOGA-Net Normal and Reduce cell.



(b) FA-Net Normal and Reduce cell.
Fig. 8. MOGA-Net and FA-Net.

TABLE 2. RESULT OF CONCRETE CRACK

Model	Concrete Error (%)	SDNET Error (%)	Params (K)	FLOPs (MMac)
NASNet [22]	0.11	8.14	56.49	8.12
AmoebaNet [23]	0.15	8.13	62.59	8.68
DARTS_V2 [26]	0.11	8.57	28.23	5.16
PC-DARTS [36]	**0.07**	**8.12**	52.43	7.33
FA-Net	0.16	8.89	49.52	7.58
MOGA-Net	0.22	9.68	**27.63**	**4.84**

D. *Transferability of Learned Architectures*

The MOGA-Net and FA-Net have been tested with the Fashion-MNIST, CINIC-10, and STL-10 datasets to determine the reliability performance of the models and compared them with other state-of-the-art models. The images of these datasets differ in the size of the image. For example, Fashion-MNIST with a size of 28 × 28 pixels, CINIC-10 with 32 × 32 pixels, and STL-10 with 96 × 96 pixels.

The training hyper-parameters were changed, to evaluate the MOGA-Net and FA-Net with these datasets where the number of all cells (normal and reduce cells) was set to 20 layers with 36 initial channels by training the network from scratch in 100 epochs with a batch size of 128, SGD optimizer with weight decay of 0.0003 and momentum of 0.9, initial learning rate is 0.025, the scheduler with cosine rule, Cutout regularization with length equal to 16, the drop-path of probability was 0.2 and auxiliary towers of weight were 0.4.

TABLE 3. RESULT OF FASHION-MNIST

Model	Fashion-MNIST Error (%)	Params (K)	FLOPs (MMac)
NASNet [22]	**4.85**	3.82	476.41
AmoebaNet [23]	4.86	3.14	386.11
DARTS_V2 [26]	5.07	3.34	417.64
PC-DARTS [36]	4.99	3.63	439.52
FA-Net	5.01	2.80	351.60
MOGA-Net	**4.85**	**2.52**	**314.80**

TABLE 4. RESULT OF CINIC-10

Model	CINIC-10 Error (%)	Params (K)	FLOPs (MMac)
NASNet [22]	11.22	3.83	624.23
AmoebaNet [23]	11.04	3.14	506.29
DARTS_V2 [26]	10.95	3.34	547.47
PC-DARTS [36]	**10.94**	3.63	576.06
FA-Net	11.73	2.81	461.23
MOGA-Net	11.19	**2.53**	**413.16**

TABLE 5. RESULT OF STL-10

Model	STL-10 Error (%)	Params (K)	FLOPs (MMac)
NASNet [22]	16.01	3.83	5618.05
AmoebaNet [23]	14.16	3.14	4556.62
DARTS_V2 [26]	13.27	3.34	4927.21
PC-DARTS [36]	**12.99**	3.63	5184.50
FA-Net	13.42	2.81	4151.02
MOGA-Net	15.37	**2.53**	**3718.39**

We found that when testing MOGA-Net with the Fashion-MNIST dataset, MOGA-Net has the smallest error value, which is equal to NASNet, while the MOGA-Net model achieved the lowest number of parameters and computational cost as shown in Table 2. However, PC-DARTS achieved the lowest error value for CINIC-10 and STL-10 datasets as in Table 3 and Table, respectively. It is astounding when we consider FA-Net in Table 4, in which it has high error values in other datasets, but for the STL-10 dataset, the error is less than MOGA-Net and it is the same with DARTSv2. FA-Net has a few parameters and computational cost is less than that of the DARTSv2 model.

IV. CONCLUSION

This paper presented Neural Architecture Search with Multi-Objective Genetic Algorithm and Firefly Algorithm, focusing on finding the CNN models with small parameters and less computational cost. According to the experimental results on Fashion-MNIST, CINIC-10, STL-10, Concrete Crack, and SDNET2018 datasets, we found that MOGA-Net achieved the

least number of parameters with lower computational cost than other state-of-the-art models such as NASNet, AmoebaNet, DARTSv2, and PC-DARTS. At the same time, we have seen the ability of the Firefly Algorithm to modify the chromosome of the model to look like as close to the best performing model. The findings suggested that the proposed algorithm is suitable for deployment on devices with limited resources such as IoT devices and embedded systems.

REFERENCES

[1] T. Fernando et al., "Neural Memory Plasticity for Medical Anomaly Detection", Neural Networks, vol. 127, pp. 67–81, Jul. 2020.

[2] K. I. Withanage, I. Lee, R. Brinkworth, S. Mackintosh, and D. Thewlis, "Fall recovery subactivity recognition with RGB-D cameras," IEEE Transactions on Industrial Informatics, vol. 12, no. 6, pp. 2312-2320, 2016.

[3] B. Tu, X. Yang, N. Li, C. Zhou, and D. He, "Hyperspectral Anomaly Detection via Density Peak Clustering", Pattern Recognition Letters, vol. 129, pp. 144–149, Jan. 2020.

[4] S. Zavrak and M. İskefiyeli, "Anomaly-based Intrusion Detection from Network Flow Features using Variational Autoencoder", IEEE Access, vol. 8, pp. 108346–108358, 2020.

[5] K. Doshi and Y. Yilmaz, "Continual Learning for Anomaly Detection in Surveillance Videos", in Proceedings of the IEEE/ CVF Conference on Computer Vision and Pattern Recognition Workshops 2020, pp. 254–255.

[6] S. C. Wong, V. Stamatescu, A. Gatt, D. Kearney, I. Lee, and M. D. McDonnell, "Track everything: Limiting prior knowledge in online multi-object recognition," IEEE Transactions on Image Processing, vol. 26, no. 10, pp. 4669-4683, 2017.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey", ACM Computing Surveys (CSUR), vol. 41, no. 3, p. 1-58, Jul. 2009.

[8] S. Akcay, A. Abarghouei, and T. Breckon, "GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training", in Asian Conference on Computer Vision, Cham, 2018, pp. 622–637.

[9] S. Yu, F. Xia, Y. Sun, T. Tang, X. Yan, and I. Lee. Detecting Outlier Patterns with Query-based Artificially Generated Searching Conditions, IEEE Transactions on Computational Social Systems, vol. 8, no. 1, p. 134-147, Feb 2021.

[10] F. Carrara, G. Amato, L. Brombin, F. Falchi, and C. Gennaro, "Combining GANs and AutoEncoders for Efficient Anomaly Detection", in 2020 25th International Conference on Pattern Recognition (ICPR), pp. 3939-3946.

[11] D. Smolyak, K. Gray, S. Badirli, and G. Mohler, "Coupled IGMM-GANs with Applications to Anomaly Detection in Human Mobility Data", ACM Trans. on Spatial Algorithms and Systems (TSAS), vol. 6, no. 4, pp. 1-14, Jun. 2020.

[12] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A Survey on GANs for Anomaly Detection", arXiv preprint arXiv:1906.11632, 2019.

[13] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural Architecture Search for Generative Adversarial Networks", in Proceedings of the IEEE/ CVF International Conference on Computer Vision, 2019.

[14] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing Neural Network Architectures using Reinforcement Learning", arXiv preprint arXiv:1611.02167, 2016.

[15] B. Zoph and Q. Le, "Neural Architecture Search with Reinforcement Learning", arXiv preprint arXiv:1611.01578, 2016.

[16] S. Xie et al., "SNAS: Stochastic Neural Architecture Search", arXiv preprint arXiv:1812.09926, 2018.

[17] S. Hu et al., "DSNAS: Direct Neural Architecture Search without Parameter Retraining", in Proceedings of the IEEE/ CVF Conference on Computer Vision and Pattern Recognition 2020, pp. 12084–12092.

[18] C. Termritthikun, Y. Jamtsho, and P. Muneesawang, "An Improved Residual Network Model for Image Recognition using a Combination of Snapshot Ensembles and the Cutout Technique", Multimedia Tools and Applications, vol. 79, no. 1, pp. 1475–1495, Jan. 2020.

[19] C. Termritthikun, Y. Jamtsho, and P. Muneesawang, "On-Device Facial Verification using NUF-Net Model of Deep Learning", Engineering Applications of Artificial Intelligence, vol. 85, pp. 579–589, Oct. 2019.

[20] C. Termritthikun, Y. Jamtsho, J. Ieamsaard, P. Muneesawang, and I. Lee, "EEEA-Net: An Early Exit Evolutionary Neural Architecture Search," Engineering Applications of Artificial Intelligence, vol. 104, p. 104397, 2021.

[21] T. Elsken, J. Metzen, and F. Hutter, "Neural Architecture Search: A Survey", the Journal of Machine Learning Research, vol. 20, no. 1, pp. 1997-2017, 2019.

[22] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le, "Learning Transferable Architectures for Scalable Image Recognition", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, pp. 8697–8710.

[23] E. Real, A. Aggarwal, Y. Huang, and Q. Le, "Regularized Evolution for Image Classifier Architecture Search", in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 1, Jul. 2019.

[24] C. Liu et al., "Progressive Neural Architecture Search", in Proceedings of the European Conference on Computer Vision (ECCV) 2018, pp. 19-34.

[25] H. Pham et al., "Efficient Neural Architecture Search via Parameter Sharing", in International Conference on Machine Learning (PMLR) 2018, pp. 4095-4104.

[26] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search", arXiv preprint arXiv:1806.09055, 2018.

[27] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", in Advances in Neural Information Processing Systems 2012, vol. 25, pp. 1097-1105.

[28] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical Representations for Efficient Architecture Search", arXiv preprint arXiv:1711.00436, 2017.

[29] A. Gómez, Y. Achaerandio, and P. Isasi, "Evolutionary Convolutional Neural Networks: An Application to Handwriting Recognition", Neurocomputing, vol. 283, pp. 38-52, 2018.

[30] X. Yang, "Nature-Inspired Metaheuristic Algorithms", Luniver Press, 2010.

[31] H. Wang et al, "A New Dynamic Firefly Algorithm for Demand Estimation of Water Resources", Information Sciences, vol. 438, pp. 95–106, Apr. 2018.

[32] N. Long, P. Meesad, and H. Unger, "A Highly Accurate Firefly based Algorithm for Heart Disease Prediction", Expert Systems with Applications, vol. 42, no. 21, pp. 8221–8231, Nov. 2015.

[33] M. Ariyaratne, T. Fernando, and S. Weerakoon, "Solving Systems of Nonlinear Equations using a Modified Firefly Algorithm (MODFA) ", Swarm and Evolutionary Computation, vol. 48, pp. 72–92, Aug. 2019.

[34] qL. Li et al., "Multi-Objective Firefly Algorithm based on Compensation Factor and Elite Learning", Future Generation Computer Systems, vol. 91, pp. 37-47, 2019.

[35] M. Tan et al., "MNASNet: Platform-Aware Neural Architecture Search for Mobile", in Proceedings of the IEEE/ CVF Conference on Computer Vision and Pattern Recognition 2019, pp. 2820–2828.

[36] Y. Xu et al., "PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search", arXiv preprint arXiv :1907.05737, Apr. 2020.

[37] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, H. Liu. Graph Learning: A Survey, IEEE Transactions on Artificial Intelligence, vol. 2, no. 2, p. 109-127, 2021.