Received 20 September 2023, accepted 4 October 2023, date of publication 11 October 2023, date of current version 23 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3323705

RESEARCH ARTICLE

Two-Phase Evolutionary Convolutional Neural Network Architecture Search for Medical Image Classification

ARJUN GHOSH[1], (Student Member, IEEE), NANDA DULAL JANA[1], (Member, IEEE), SWAGATAM DAS[2], (Senior Member, IEEE), AND RAMMOHAN MALLIPEDDI[3], (Senior Member, IEEE)

[1]Department of Computer Science and Engineering, National Institute of Technology Durgapur, Durgapur 713209, India
[2]Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata 700108, India
[3]Department of Artificial Intelligence, Kyungpook National University, Daegu 702-701, South Korea

Corresponding author: Rammohan Mallipeddi (mallipeddi.ram@gmail.com)

This work was supported by the Core Research Institute Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2021R1A6A1A03043144.

ABSTRACT Recently, convolutional neural networks (CNNs) have shown promising achievements in various computer vision tasks. However, designing a CNN model architecture necessitates a high-domain knowledge expert, which can be difficult for new researchers while solving real-world problems like medical image diagnosis. Neural architecture search (NAS) is an approach to reduce human intervention by automatically designing CNN architecture. This study proposes a two-phase evolutionary framework to design a suitable CNN model for medical image classification named TPEvo-CNN. The proposed framework mainly focuses on architectural depth search and hyper-parameter settings of the layered architecture for the CNN model. In the first phase, differential evolution (DE) is applied to determine the optimal number of layers for a CNN architecture, which enhances faster convergence to achieve CNN model architectures. In the second phase, the genetic algorithm (GA) is used to fine-tune the hyper-parameter settings of the generated CNN layer architecture in the first phase. Crossover and mutation operations of GA are devised to explore the hyper-parameter search space. Also, an elitism selection strategy is introduced to select the potential hyper-parameters of the CNN architecture for the next generation. The suggested approach is experimented on six medical image datasets, including pneumonia, skin cancer, and four COVID-19 datasets, which are categorized based on image types and class numbers. The experimental findings demonstrate the superiority of the proposed TPEvo-CNN model compared to existing hand-crafted, pre-trained, and NAS-based CNN models in terms of classification metrics, confusion matrix, radar plots, and statistical analysis.

INDEX TERMS Convolutional neural network, differential evolution, genetic algorithm, medical imaging, neural architecture search.

I. INTRODUCTION

In recent years, deep learning models, specifically convolutional neural networks (CNNs), have shown impressive performance in various computer vision tasks, including image classification [1], object recognition [2], image splicing [3] and image segmentation [4]. Several CNN models, including LeNet [5], AlexNet [6], VGGNet [7], Inception [8], ResNet [9], and DenseNet [10], have gained popularity due to their ability to solve specific problems with large numbers of parameters and design options. Hence, the performance of a CNN model varies depending on the problem nature and the architecture used in the CNN model. The performance of a CNN model is directly affected by its architectural design, which includes the layer types, arrangement, and associated hyper-parameter settings of the layers [11]. Notably, developing a CNN model is vital for a specific problem that needs human involvement, and a trial-error method leads to considerable time and computational resources. Therefore, designing a suitable

The associate editor coordinating the review of this manuscript and approving it for publication was Vivek Kumar Sehgal.

CNN model for a task like medical image classification by considering diverse architectures and hyper-parameter settings is challenging.

Neural architecture search (NAS) is an approach recently developed for automatically designing the architecture of CNN, which involves determining the appropriate layer arrangement and hyper-parameter configuration [12]. It has three components: search space, search strategy, and performance estimation strategy. The search space pertains to the representation of CNN architectures. The search strategy involves efficient exploration of the architectural search space to discover an appropriate CNN model. Finally, performance evaluation is concerned with assessing the performance of the generated CNN model architectures, aiming to speed up the search process and reduce the evaluation cost. Furthermore, recent research [13], [14] revealed that NAS can be treated as a non-linear complex optimization problem.

Recently, researchers have concentrated on several NAS components to design the CNN model architecture [15], [16], [17], [18], [19], [20]. Most of the existing works are dedicated to explore the various search strategies. Most commonly, the search strategies can be classified into reinforcement learning (RL) based [21], [22], [23], [24], [25], and evolutionary algorithm (EA) based [26], [27], [28], [29], [30], [31], [32], [33] search. RL-based strategy can not be suitable for NAS because it requires high computing resources and is often computationally expensive. On the other hand, EA-based search strategies have shown promises in NAS research due to powerful global search capabilities. Researchers have developed meta-heuristic algorithms inspired by nature and animal communities' behaviour to design optimal CNN model architecture [34], [35]. However, most of these studies significantly impact the performance of the designed CNN model architecture.

The earliest EA-based algorithm is Stanley and Miikkulainen's neuroevolution of augmenting topologies (NEAT) [36], which simultaneously optimizes a feed-forward neural network's topology structure and weights. To learn deep CNN structures automatically, Xie and Yuille [26] used a genetic algorithm (GA) that utilizes an encoding technique to represent every network structure as a binary string with a fixed length. Sun et al. [30] presented an encoding scheme that varies with the length and a novel fitness evaluation method using GA to construct an optimal CNN model for the image classification task. Most of these studies encoded CNN model architectures randomly from pre-defined architectural search spaces, including layer types and the total number of individual layers. It can be observed that more randomness may cause the architecture design to get trapped in a local optimum, which leads to premature solution [37]. Therefore, starting the initial population of CNN architecture with the exemplary layer types is an excellent research opportunity. These will accelerate the required convergence rate to develop a worthy CNN model. Furthermore, it is also essential to properly adjust the hyper-parameters of each layer to enhance the performance of a CNN structure.

This work proposes a two-phase evolutionary framework to design a suitable CNN model for biomedical image classification tasks named TPEvo-CNN. The first phase includes architecture depth search (ADS), which can find the optimal number of layer types of the CNN model by using the conventional differential evolution (DE) algorithm. On the other hand, the second phase is concerned with the hyper-parameter search (HPS) of the CNN layers generated in the first phase by using an improved genetic algorithm (GA), which is enhanced with introduced crossover and mutation operations. Overall, the contributions in this work are as follows:

- A two-phase evolutionary framework for designing compatible CNN model architecture is proposed to explore the NAS-based biomedical image classification.
- Phase I (architecture depth search) finds suitable convolutional, pooling, and fully connected layer types for the target tasks using the DE algorithm. This architectural layer information is transferred to Phase II.
- Phase II (hyper-parameter search) acts as hyper-parameter optimization by tuning the hyper-parameters of the layer architectures obtained from Phase I through GA. Here, crossover and mutation operators are proposed to explore the hyper-parameter search spaces.
- The experiment is conducted on six biomedical image datasets, including four types of COVID-19, pneumonia, and skin cancer datasets, to measure the efficiency and effectiveness of the proposed model.
- A comprehensive comparison among the results of various hand-crafted, pre-trained, NAS-based CNN models and the proposed model itself are conducted based on classification metrics, confusion matrices, radar plots, and statistical analysis to validate the obtained results.

The subsequent sections of the article are organized as follows: Section II outlines the fundamentals of CNN model architecture, differential evolution, genetic algorithm, and related work relevant to our investigation. The proposed approach is described in Section III. Section IV presents the datasets description, parameter settings, and system configuration. The experimental results of the proposed model and a comparison with existing state-of-the-art methods are presented in Section V. Finally, Section VI provides the conclusion of this study and outlines the future scope.

II. PRELIMINARIES

A. CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional neural networks (CNNs) are one class of deep learning models primarily used for computer vision tasks [38]. These networks comprise several layers arranged in a specific sequence to build an operational CNN model. These layers can be classified into convolution denoted as Convo, pooling (Pool), and fully connected (FC) layers. The convolution layer serves as the primary unit of CNN and is responsible for extracting the hierarchical features from the input data [39]. The size of the convolutional layer in a CNN is determined by the number of filters, the size of the filters, and the stride size. The pooling layer reduces the dimensionality of the feature map generated by the convolutional layer with hyper-parameters such as pooling kernel size, stride size, and pooling type. The most common types of pooling used are max pooling and average pooling. Max pooling retains the most prominent features from the convolved feature map, while average pooling calculates the average value of the features from the selected feature map. After a few convolutional and pooling layers sequences, the obtained featured map is flattened into a single column and fed to the FC layers for classification. Finally, the probability distributions of the CNN model outputs are computed through a softmax layer.

There is no fixed guideline or rule to provide a precise number of Convo, Pool, and FC layers that can be used in a CNN model. Furthermore, the arrangement of these layer types is also a challenging task. It can consist of sets of Convo layers followed by Pool or alternated Convo and Pool layers. Thus, the overall number of each layer type, their placement in the CNN architecture, and their corresponding hyper-parameter configurations have a vast exploring research area for solving a problem.

B. DIFFERENTIAL EVOLUTION (DE)

DE is a population-based meta-heuristic algorithm developed to search global solutions for non-linear complex optimization problems [40]. It has four steps: initialization, mutation, crossover, and selection to guide the search process. Initially, it starts with a population (X) of individuals ($\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{NP}$), which are initialized randomly within the search space. Mutation operation produces donor vector ($\mathbf{v}_i^g$) of the i^{th} individual at generation g. The commonly used mutation operation is defined as:

$$\mathbf{v}_i^g = \mathbf{x}_{r_1}^g + F \times \left(\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g\right), \quad (1)$$

where, $\mathbf{x}_{r_1}^g$, $\mathbf{x}_{r_2}^g$, and $\mathbf{x}_{r_3}^g$ are three mutually exclusive distinct individuals randomly taken from the population. F is a positive parameter denoted as the scale factor that scales the difference between two vectors. After generating the donor vector, the crossover operation is performed. The most common binomial crossover is mathematically defined in Eq. (2).

$$u_{j,i}^g = \begin{cases} v_{j,i}^g & \text{if } rand_j\,[0,1] \leq CR \text{ or } j = j_{rand} \\ x_{j,i}^g & Otherwise, \end{cases} \quad (2)$$

where $u_{j,i}^g$ denotes the j^{th} dimension of the i^{th} individual at g^{th} generation. The crossover operation begins by generating a random number for each individual, j_{rand}. Additionally, a random number, $rand_j \in (0, 1)$, is generated for each dimension of each vector. These values are then compared to the crossover rate, CR, and j_{rand} to determine whether the crossover will be applied to the dimension. Subsequently, a trial vector, $\mathbf{u}_i^g$, is created, and its fitness is compared to that of the parent vector to identify the superiority between them. The mutation, crossover, and selection are then repeated sequentially until the maximum number of generations or a user-defined stopping criterion is reached.

C. GENETIC ALGORITHM (GA)

GA is a meta-heuristic approach that follows the natural selection process of the environment [41]. The algorithm uses a natural selection strategy to develop high-quality solutions for complex optimization problems. The GA comprises five steps: population initialization, fitness evaluation, selection, crossover, and mutation. GA involves creating an initial population consisting of all feasible solutions for a given problem. Each individual, known as a chromosome, comprises a collection of genes. Individuals are evaluated using a fitness function, which describes the capability to form an optimal solution. Individuals with higher fitness values will have a greater chance of being selected to reproduce in subsequent generations. Depending on the application and the search problem, several fitness functions are used [42]. Classification accuracy, for example, is a typical fitness operation for various classification problems. In the selection step, individuals are chosen based on their fitness values for the next generation. One of the essential components of the GA is the crossover operation. This operation produces new offspring from the individual parents by exchanging information of genes based on the crossover probability P_c. Some genes in new offspring are mutated using mutation operations based on the mutation probability P_m to increase population diversity from generation to generation. This GA process ends when the stopping criteria are met, or the maximum computing budget is reached. The final step involves reporting the best individual found across all generations as the optimal solution.

D. RELATED WORKS

Designing an optimal CNN model for computer vision tasks has no pre-defined rules; therefore, it requires expertise to create such models. Minimizing human intervention while creating a suitable architecture for a given task is challenging. Researchers have recently introduced various approaches to automatically build the optimal CNN architecture for their problem. This section discusses related works on several NAS models, including RL and EA methods for image classification tasks. Furthermore, we have also discussed the various CNN models built particularly for COVID-19 datasets.

1) RL-BASED CNN MODELS

Baker and his colleagues [21] introduced an RL-based meta-modelling algorithm called MetaQNN, which aims at automatically producing high-performance CNN architectures for the classification task. To choose CNN layers and their corresponding parameters from a predefined finite search space, the authors employ Q-learning with an ϵ-greedy exploration strategy. The performance of MetaQNN is tested on three datasets, including CIFAR-100, SVHN, and MNIST datasets, and compared with existing NAS models regarding their classification accuracy. Zoph et al. [22] proposed CNN architectures design by using the RL method, which they named NASNet. NASNet searches architectural building blocks composed of Convo, Pool, and FC layers on a small CIFAR-10 dataset. The next step involves transferring the blocks to a larger dataset, ImageNet, by combining multiple copies of these blocks, each with its own set of parameters, to form a complete CNN architecture. Moreover, the authors also proposed a new regularisation approach to improve the model's generalization significantly. Cai et al. [23] suggested an efficient architecture search (EAS) framework that efficiently explores the architecture space by reusing the initial network's weights. The authors introduced a meta-controller that utilizes function-preserving transformation to increase the network's depth or layer width. Their proposed method tested on only two datasets, CIFAR-10 and SVHN. However, RL significantly impacts the automatic design of CNN models from the NAS perspective, but this requires handling both an agent and child network (generated by the agent), which is tedious and time-consuming [43].

2) EA-BASED CNN MODELS

Recently, researchers have shown a growing interest in EA to design CNN architecture for image classification tasks. Xie and Yuille [26] introduced the automatic architecture design of CNN (GeNet) for image classifications. They used a fixed-length variable encoding schema to represent each CNN architecture. They followed the selection, crossover, and mutation operations of GA on each encoded network to evolve the CNN architecture. However, the proposed work only focused on the connection between layers without considering the corresponding hyper-parameters of those layers. The model was tested on the CIFAR-10 dataset. Zhu et al. [27] introduced a method for efficient evolution of neural architecture (EENA) regularized for image classification. Their proposed EENA is guided by prior learning experiences that accelerate the search process with the crossover and various mutation operations, which can lead to reduced computational costs. The authors tested their algorithm only on the CIFAR-10 and CIFAR-100 datasets. Tan et al. [28] introduced a multi-objective optimization algorithm for evolving the network architecture of the EfficientNet-B7 model. Also, they incorporated a uniform scaling approach that scales the convolutional layer depth and width. Their proposed method is investigated on three datasets: CIFAR-10, CIFAR-100, and ImageNet. This approach might be expensive in computation due to the inclusion of multiple objective functions. Moreover, the architecture layout is also restricted to a specific CNN model. Suganamu et al. [29] proposed cartesian genetic programming (CGP) to construct high-performing CNN architectures (CGP-CNN) for classification tasks. The proposed method used CGP to encode CNN structure along with different layer connections. (1+λ) evolution strategy helps to explore the CNN architecture search, where CNN layer types and connectivities change randomly based on genetic mutation operations. In addition, they also implemented rich initialization and early stopping techniques into the CGP-CNN to accelerate the evolution process. The authors evaluated the performance of the proposed model on CIFAR-10, CIFAR-100, and ImageNet datasets and compared it with only two manual CNN designs: VGG-16 and ResNet architecture. However, the early stopping mechanism may stop the training earlier before getting the optimal solution. Fernandes and Yen [34] suggested an approach based on particle swarm optimization (PSO) for optimal CNN architecture to address the image classification tasks and named it psoCNN. They introduced a novel variable-length encoding scheme to represent each CNN architecture. PSO velocity operation updated the CNN layer types by copying them from personal/global best solutions to optimize the search process. The effectiveness of the psoCNN was tested on various image datasets, including MNIST and its variations, convex and rectangular image datasets. However, the architectural search space may need to be thoroughly explored because each new particle is constructed based on the only personal or global best particle. Table 1 summarizes the EA-based NAS studies focusing on advantages, disadvantages and limitations.

3) CNN MODELS FOR COVID-19

Recently, some studies have focused on designing better CNN models to diagnose COVID-19 patients by analyzing various chest X-ray image datasets. Wang et al. [44] developed COVID-Net, an early CNN model specifically created for identifying COVID-19 from chest X-ray images. The authors evaluated their model using a four-category classification task, distinguishing between normal, bacterial pneumonia, viral pneumonia, and COVID-19 chest X-ray images. Their experimental results demonstrated superior classification accuracy against the popular manually designed CNN model architectures. Apostolopoulos and Mpesiana [45] employed transfer learning with advanced popular CNN architectures to diagnose COVID-19 patients. They found that the proposed MobileNet CNN architecture outperformed other models regarding classification accuracy in three class category tasks, such as healthy, pneumonia, and COVID-19 X-ray datasets. Another study by Ozturk and colleagues [46] proposed the DarkCovidNet model for COVID-19 diagnosis using two different datasets: a binary dataset consisting of no-findings versus COVID-19 classes and a multi-class dataset consisting of no-findings, pneumonia, and COVID-19 classes. According to the investigation, the DarkCovidNet algorithm demonstrated better performance in identifying COVID-19 cases when working with the binary dataset than the multi-class dataset. Waheed et al. [47] designed a new model named CovidGAN to improve the performance of CNN architectures for COVID-19 diagnosis. They discovered that actual data with synthetic augmentation (CNN-SA) generated from an auxiliary classifier generative adversarial network (ACGAN) based model for COVID-19 X-ray images

TABLE 1. Summary of the EA-based CNN models.

Model	Advantage	Disadvantage	Limitation
GeNet [26]	The proposed model encodes with fixed-length binary string and defines several standard GA operations to design CNN model architecture.	Hyper-parameters of different CNN layer types are not considered during the optimization process.	The work is tested on only one limited image dataset.
EENA [27]	The suggested model is guided by prior learning experiences that accelerate the search process of crossover operation.	Various mutation operations are absorbed to mutate the generated CNN models that make the model computationally expensive.	The authors only evaluate the proposed method on limited datasets. It would also be interesting to see how the method performs on other datasets.
EfficientNet-B7 [28]	The method incorporates a uniform scaling approach that scales the convolutional layer depth and width.	This approach can be computationally expensive due to the inclusion of multiple objective functions.	The authors demonstrate the effectiveness of this method on MobileNets and ResNet only.
CGP-CNN [29]	The proposed method uses (1+λ) evolution strategy to explore the CNN architecture search.	The early stopping mechanism may stop the training earlier before getting the optimal solution.	The model compared with only two manual design models: VGG-16 and ResNet architecture.
psoCNN [34]	It proposes particle swarm optimization (PSO) to automatically search for good CNN architectures for image classification tasks.	The method relies on extensive computational resources and time to conduct a thorough architecture search.	The search space may not be explored as the new particle is constructed from the global or personal best particle.

contained crucial features and visualizations that enhanced the classification performance of COVID-19 X-ray images. Khan et al. [48] introduced a CoroNet model that utilizes pre-trained Xception architecture that was tested on the large chest X-ray dataset. The model has been evaluated in four, three, and two-category classifications, including normal, bacterial pneumonia, viral pneumonia, and the COVID-19 image dataset. This study achieved better results with minimal data pre-processing. Ezzat et al. [49] developed the GSA-DenseNet121-COVID-19 model that used the pre-trained CNN DenseNet121 architecture and employed gravitational search algorithm (GSA) to get the optimal hyper-parameters. The results demonstrated improved classification performance when using a two-class dataset of COVID-19 positive versus negative cases. Song and colleagues [50] developed a DRENet model based on the ResNet50 CNN model and included a feature extraction mechanism to extract image details. The model is tested on computed tomography (CT) images with three classes: normal or healthy, bacterial pneumonia, and COVID-19. Ibrahim et al. [51] employed supervised deep learning techniques to develop four classification models, each combining CNN, Bi-GRU, GRU with VGG19 and ResNet152V2 CNN models. The models are evaluated using publicly available CT and chest X-ray images in four classes: normal, pneumonia, lung cancer, and COVID-19. Experimental outcomes indicated that the VGG19+CNN model exhibited superior performance among the four proposed models. Louati et al. [52] developed a bi-level optimization approach called Bi-CNN-D-C for joint design and compression of convolutional neural networks. The upper level generates the new architecture with minimum convolutional blocks, while the lower level prunes the generated architecture. The co-evolutionary migration-based algorithm (CEMBA) is used as a search engine to address the bi-level architectural optimization problem. The Bi-CNN-D-C model is tested on a two-class image dataset comprising the cases of normal and COVID-19. The results suggest that the proposed model outperforms existing CNN when applied to a similar dataset. Singh et al. [53] introduced a meta-heuristic strategy to optimize deep CNN to detect COVID-19 from chest X-ray images and named it CNGOD. The deep learning model comprises depth-wise separable convolutions that independently look at cross-channel and spatial correlations. The grasshopper optimization algorithm (GOA) is used to optimize the network. The maximum probability-based cross-entropy (MPCE) loss function minimizes the classification error and improves the training time. The model is evaluated on three chest X-ray images: normal, pneumonia, and COVID-19. Table 2 summarizes the related COVID-19 CNN models by providing the proposed approaches, limitations, and the datasets used in the original work.

This study proposes an algorithmic framework incorporating a two-phase strategy for automatically evolving CNN architecture. The proposed method is tested on several medical image datasets to examine the effectiveness of the proposed work. To the best of our knowledge, only a few studies have previously examined these datasets from the NAS perspective.

III. THE PROPOSED APPROACH (TPEVO-CNN)

A. MOTIVATION

The evolution of the CNN process involves searching for an optimal architecture for a specific task, often finding the best combinations of layer types, layer connections, and connection methods (such as summation or cascading). However, the effectiveness of any CNN model depends not only on its basic architecture but also on the appropriate hyper-parameter settings of the layer types. Selecting appropriate hyper-parameters is a critical job. Manual hyper-parameter tuning is a trial-and-error process requiring enough time to evaluate each hyper-parameter setup associated with a CNN model. However, our proposed work is mainly based on separate optimization of layers

TABLE 2. Summary of the start-of-the-art CNN models applied on COVID-19 datasets.

Model	Proposed Work	Limitation	Dataset
COVID-Net [44]	This paper employs a collaborative design approach combining human-driven network design with machine-driven exploration to create the proposed model.	Large number of connections in densely-connected deep CNN architectures is a noticeable increase in computational complexity and memory overhead.	Normal, Bacterial pneumonia, Viral pneumonia and COVID-19 chest X-ray images.
MobileNet [45]	The work adopts transfer learning with advanced popular CNN architectures to diagnose COVID-19 patients.	The study solely analyzes chest X-ray images, but have not compared the proposed method with other existing detection techniques.	Healthy, Pneumonia and COVID-19 chest X-ray images.
DarkCovidNet [46]	The method employs a pre-trained ResNet model for automated COVID-19 diagnosis. The model follows an end-to-end architecture, eliminating the need for feature extraction techniques.	The chest X-ray images are obtained from only two different sources, which may limit the model's generalizability to other sources of X-ray images.	No-Findings, Pneumonia, COVID-19 chest X-ray images.
CovidGAN [47]	The model uses ACGAN that synthesizes images generated of COVID-19 chest X-rays containing crucial features and visualizations, which enhances the classification performance of the model.	The quality of the synthetic samples produced in this research could be improved by integrating more labelled data, improving the learning process of GAN.	Normal and COVID-19 chest X-ray images.
CoroNet [48]	The method employs a pre-trained Xception model for automated COVID-19 X-ray image diagnosis.	The proposed model has not been validated on a large-scale dataset or other clinical real-life applications.	Normal, Bacterial pneumonia, Viral pneumonia and COVID-19 chest X-ray images.
GSA-DenseNet121-COVID-19 [49]	The model uses the pre-trained DenseNet121 architecture and applied gravitational search algorithm (GSA) to get the optimal CNN model for detecting COVID-19 disease.	It is unclear how well it would perform on other medical images or in different medical contexts.	COVID-19 positive and COVID-19 negative chest X-ray images.
DRENet [50]	This model is based on the ResNet50 CNN model and includes a feature extraction mechanism to extract image details.	The paper mentioned that the training data is still small, which limits the model's effectiveness in solving accurate predictions for other datasets.	Healthy or normal, Bacterial pneumonia and COVID-19 CT images.
Deep-chest [51]	Supervised deep learning techniques are used to create four classification models to detect the COVID-19 CT and X-ray images	The proposed model takes much training time and is computationally expensive.	Normal, Pneumonia, Lung cancer and COVID-19 chest X-ray and CT images.
Bi-CNN-D-C [52]	A bi-level optimization approach is used to design and compress the CNN architecture jointly. The upper level generates an architecture, while the lower level prunes the generated architecture.	The paper does not provide a detailed analysis of the computational complexity of the proposed approach, which may be a concern for large-scale datasets.	Normal, Pneumonia, Lung cancer and COVID-19 chest X-ray and CT images.
CNGOD [53]	The paper proposes a novel deep CNN model generated using the grasshopper optimization algorithm (GOA) to detect COVID-19 chest X-ray images.	The proposed method has not been tested on a large dataset with different class categories.	Normal, Pneumonia and COVID-19 CT images.

and hyper-parameters in one algorithmic framework as the problem becomes more complex while both are optimized simultaneously by exponential increase of search space due to the encoding size of CNN architecture, making it computationally expensive and challenging to explore promising CNN architecture. Therefore, the architecture and hyper-parameter search of a CNN model within one algorithmic framework can significantly improve the model's performance. Thus, an efficient two-phase evolutionary framework is proposed to design an optimal CNN model architecture for solving medical image classification tasks.

In the first phase, the DE algorithm finds the optimal architectural depth of CNN models, including numbers of Convo, Pool, and FC layers. In the second phase, HPS is performed using GA with proposed crossover and mutation operations for selecting appropriate hyper-parameter settings for CNN models generated in the first phase. Moreover, an elitism selection strategy is employed to hold the promising hyper-parameters of CNN architecture for the next generation. The proposed approach aims to accelerate faster convergence to achieve a high-performing, suitable CNN model for medical image classification tasks. Detailed descriptions of each phase of the proposed method are presented in the upcoming subsections.

B. OVERVIEW OF TPEVO-CNN

The entire framework of the proposed TPEvo-CNN algorithm is depicted in Figure 1. At first, the given dataset is divided into training and testing sets. The training set is again split into two groups, namely training and validation, to evaluate the fitness of the generated CNN architecture by the proposed framework. The proposed two-phase method is conducted sequentially in Phase I and Phase II. In Phase I, a set of optimal depth CNN architectures are determined using the DE algorithm named architecture depth search (ADS). The hyper-parameter settings of the generated CNN architectures in Phase I are tuned using GA called hyper-parameter search (HPS). The crossover and mutation operations of GA are devised to achieve hyper-parameter settings of the CNN model. Moreover, an elitism selection strategy is introduced to keep the potential individuals for the subsequent generations in the HPS. Hence, the best CNN

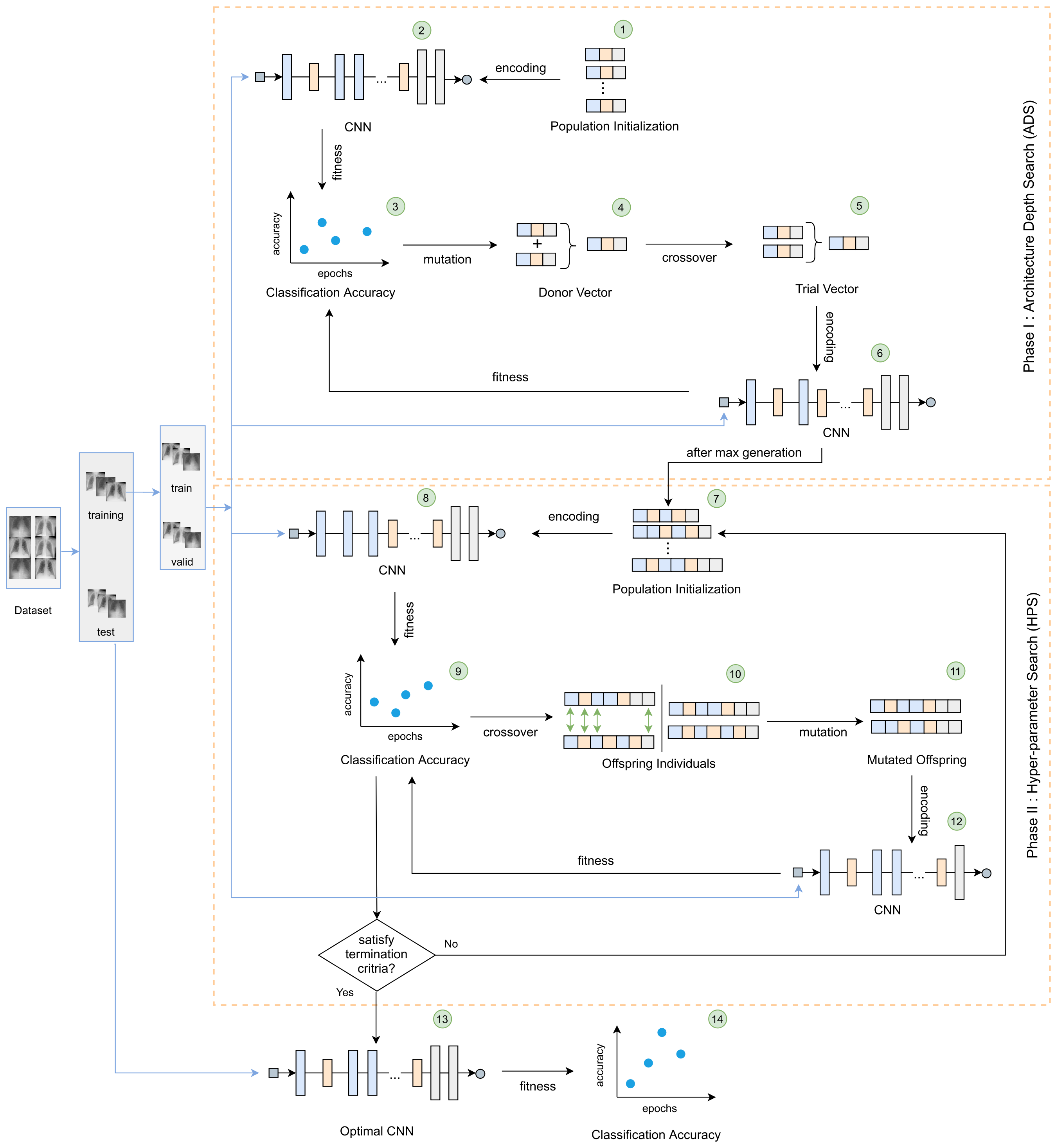

FIGURE 1. The overall framework of TPEvo-CNN algorithm.

architecture is selected, considering the highest classification accuracy based on the validation dataset within the predefined maximum generations. Finally, the obtained best CNN model is trained on the original training dataset and tested on the test dataset to record the classification accuracy for performance evaluation of the proposed method.

C. PHASE I: ARCHITECTURE DEPTH SEARCH

Optimal depth is crucial to design the initial structure of any CNN model. This phase determines the number of Convo, Pool, and FC layers in a workable CNN model. The total number of layers within a given CNN model represents its depth. In the proposed work, a conventional

Algorithm 1 Proposed Phase I (ADS)

```
Input: population size: $NP$, maximum generation:
    $g_{max}$, mutation factor: $F$, crossover rate: $CR$.
Output: optimal depth of each individual.
$X \leftarrow \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{NP}$ with $D$ dimensions;
    // Initialize population
for $i = 1$ to $NP$ do
    $\mathbf{x}_i \leftarrow$ randomly initialize the number of Convo,
     Pool and FC layer from the Section IV-B;
    // fitness evaluation
    $f(\mathbf{x}_i) \leftarrow$ compute_fitness($\mathbf{x}_i, D_{train}, D_{valid}$);
    $fitness(i) = f(\mathbf{x}_i)$;
end
while $g \leq g_{max}$ do
    for $i = 1$ to $NP$ do
        Generate three random indexes $r_1, r_2$ and
         $r_3$ with $r_1 \neq r_2 \neq r_3 \neq i$;
        $\mathbf{v}_i^g \leftarrow \mathbf{x}_{r_1}^g + F\ (\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g)$;
        // mutation
        $j_{rand} \leftarrow$ randind(1,D); // crossover
        for $i = 1$ to $D$ do
            if $rand_j(0,1) \leq CR \,||\, j = j_{rand}$ then
                $\mathbf{u}_{i,j}^g \leftarrow \mathbf{v}_{i,j}^g$;
            else
                $\mathbf{u}_{i,j}^g \leftarrow \mathbf{x}_{i,j}^g$;
            end
        end
        $fitness(i) = f(\mathbf{u}_i^g)$;
    end
    // selection for next generation
    if $f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g)$ then
        $\mathbf{x}_i^{g+1} \leftarrow \mathbf{u}_i^g$;
    else
        $\mathbf{x}_i^{g+1} \leftarrow \mathbf{x}_i^g$;
    end
end
$g \leftarrow g + 1$;
Return optimal depth of $\{\mathbf{x}_1, \ldots, \mathbf{x}_{NP}\}$;
```

DE is used to find the optimal number of different layer types to design the basic structure of a functional CNN. We devise the proposed ADS strategy in Algorithm 1. The corresponding population initialization, fitness evaluation, mutation, crossover, and selection operations are expanded in the following subsections.

1) POPULATION INITIALIZATION

Population is defined as the number of individuals initially distributed within the whole search space. Here, X is a population of NP individuals denoted as $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{NP}\}$. Each $\mathbf{x}_i$ has three dimensions representing the number of Convo, Pool, and FC layers for a CNN structure. In the initialization process, individuals in the population are initialized randomly with a pre-defined architectural search space as described in Section IV-B.

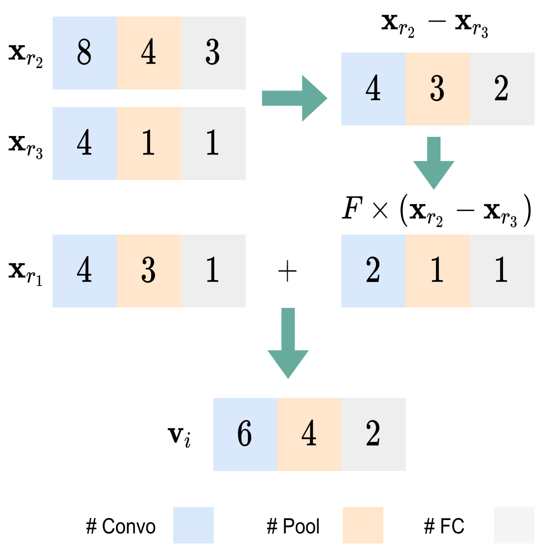

FIGURE 2. An example of mutation operation in Phase I.

2) FITNESS EVALUATION

Every individual is evaluated through a fitness function, which provides the quality of that individual to perform some tasks. The fitness is assessed by considering the classification accuracy of the CNN model concerning the validation dataset. Each individual needs to convert a workable CNN architecture during the fitness evaluation phase. Therefore, after selecting the number of convo, pool, and FC layers, we randomly define the hyper-parameters of each layer from the pre-defined hyper-parameters set mentioned in Section IV-B. It should be noted that the first and last layers of each individual must be limited to Convo and FC according to the CNN framework. The popular categorical cross-entropy function [54] is used to calculate the error value of each generated CNN architecture. Then, the error value is converted to classification accuracy by subtracting the corresponding error value from 1. The Xavier initialization [55], rectified linear unit (ReLU) [56], and Adam optimizer [57] are used in this work to initialize weights, activation function, and optimizing parameters of the CNN model.

3) MUTATION

DE performs the mutation operation to generate the donor vector, $\mathbf{v}_i^g = (\mathbf{v}_{i,1}^g, \mathbf{v}_{i,2}^g, \ldots, \mathbf{v}_{i,D}^g)$ corresponding to the target vector $\mathbf{x}_i^g$. Among the different mutation schemes, the $DE/rand/1$ scheme is used widely in various applications due to its easy implementation and good diversity in the search space [58]. This mutation scheme is used to explore the depth of CNN model architecture in our study and illustrated in Figure 2. In the example, $\mathbf{x}_{r_1}$, $\mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$ are three mutually exclusive random individuals chosen from the population

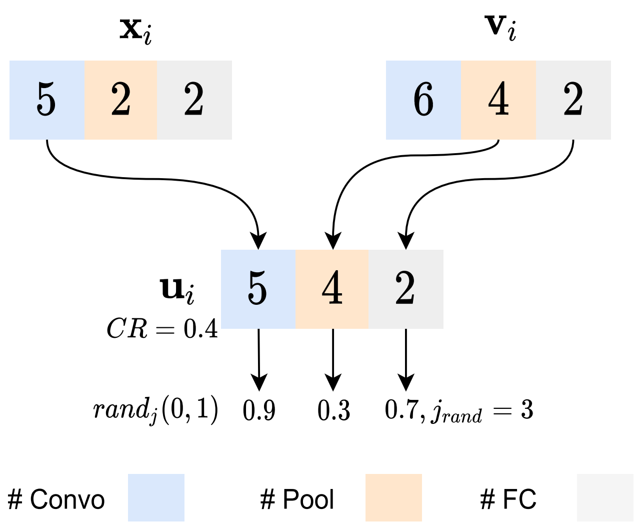

FIGURE 3. Example of crossover operation in Phase I.

X to generate the donor vector corresponding to the target vector $\mathbf{x}_i$ respectively. The difference ($\mathbf{x}_{\mathbf{r}_2}$-$\mathbf{x}_{\mathbf{r}_3}$) is computed between $\mathbf{x}_{\mathbf{r}_2}$ and $\mathbf{x}_{\mathbf{r}_3}$ on the corresponding dimension and the absolute value is considered. Boundary checking is done if both $\mathbf{x}_{\mathbf{r}_2}$ and $\mathbf{x}_{\mathbf{r}_3}$ have the same number of similar layer types in the same j^{th} dimension. Then, the difference vector is multiplied with the scaling factor F, and the corresponding real number (floating point) is converted into an integer using the round-off technique. Here, F is set as 0.6 based on the literature study of the DE algorithm [58]. Finally, $\mathbf{x}_{\mathbf{r}_1}$ is added to generate the donor vector $\mathbf{v_i}$ corresponding to the target vector $\mathbf{x_i}$.

4) CROSSOVER

After mutation, a trial vector $\mathbf{u}_i^g = (\mathbf{u}_{i,1}^g, \mathbf{u}_{i,2}^g, \ldots, \mathbf{u}_{i,D}^g)$ is obtained according to the binomial crossover operator on $\mathbf{x}_i^g$ and $\mathbf{v}_i^g$, as shown in Figure 3. In the crossover, one component of $\mathbf{v_i}$ is picked randomly and assigned to the j_{rand} variable. A set of random numbers $r \in (0, 1)$ is generated for each dimension (j) for creating trial individual, $\mathbf{u_i}$. Finally, if $r \leq CR$ or $j = j_{rand}$, then j^{th} component of $\mathbf{u_i}$ comes from the corresponding component of $\mathbf{v_i}$; otherwise from $\mathbf{x_i}$. This process acts as of Eq. (2).

5) SELECTION

Individuals selected for the next generation ($g + 1$) are based on the classification accuracy as compared between the target vector $\mathbf{x_i}$ and the trail vector $\mathbf{u_i}$ which is defined in Eq.(3).

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g & \textit{if } f(\mathbf{u}_i^g) \geq f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g & \textit{Otherwise} \end{cases} \quad . \tag{3}$$

Here, f represents the fitness function that reveals the classification accuracy concerning the validation set. The mutation, crossover, and selection processes are repeated to the maximum number of generations as the user provides.

Finally, a set of individuals are collected consisting of the architectural information about the number of Convo, Pool, and FC layer for a workable CNN model, and all the individuals are transferred to Phase II for hyper-parameter tuning of these layer architectures.

Algorithm 2 Proposed Phase II (HPS)

```
Input: population size: $NP$, maximum generation: $T$,
       crossover probability: $P_c$, mutation
       probability: $P_m$.
Output: optimal hyper-parameter settings of each
        individual.
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{NP}\} \leftarrow$ population initialization of
  hyper-parameters with the predefined ranges;
$fitness \leftarrow$ compute_fitness ($\mathbf{x}_i$);
for $t = 1$ to $T$ do
    $Q_t \leftarrow \emptyset$;
    while | $Q_t$ |< $NP$ do
        $o_1, o_2 \leftarrow$ compute_crossover ($p_1, p_2, \rho_c$);
        // generate two offspring
           individuals using proposed
           crossover strategy from $P_{t-1}$

        $o_1, o_2 \leftarrow$ compute_mutation ($o_1, o_2, \rho_m$);
        // mutate two offspring
           individuals using proposed
           mutation strategy
        $Q_t \leftarrow Q_t \cup o_1 \cup o_2$;
    end
    $fitness \leftarrow$ compute_fitness ($Q_t$);
    // calculate the fitness of each
       individual in $Q_t$
    $p_t \leftarrow$ compute_selection ($P_{t-1}, Q_t$);
    // choose the individuals for the
       next generation based on the
       proposed selection method
end
Return optimal hyper-parameter of $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$;
```

D. PHASE II: HYPER-PARAMETERS SEARCH

In this phase, the GA algorithm is used to fine-tune the hyper-parameters of the CNN architecture generated in Phase I. The pseudo-code of the proposed HPS is presented in Algorithm 2. At the beginning, each individual (found during the ADS phase) is randomly assigned the associated hyper-parameters to each layer type from predefined ranges provided in the Subsection IV-B. After that, the evolution of hyper-parameters is performed by the GA algorithm with the proposed crossover and mutation operations. Furthermore, an elitist selection approach is introduced to select potential CNN hyper-parameter settings for the next generation.

Algorithm 3 Fitness Evaluation

```
Input: population size: $NP$, training data: $D_{train}$,
       validation data: $D_{valid}$, maximum epoch
       number: $epo_{max}$.
Output: individual fitness.
for each individual $\mathbf{x}_i$ do
    CNN ← converts the encoded information of an
      individual into a CNN architecture ;
    Train CNN on $D_{train}$ for a total of $epo_{max}$ epochs;
    $\mathbf{x}_i$.fitness ← accuracy of CNN on $D_{valid}$;
end
Return fitness of each individual;
```

1) POPULATION INITIALIZATION

Population initialization impacts the performance of the NAS study. Initialization of CNN architectures, i.e., individuals, can be made to distribute throughout the search space. In Phase II, the population size remains the same as in Phase I because we must find hyper-parameter settings of the same number of CNN architectures generated in Phase I. The obtained individuals containing the number of Convo, Pool, and FC layers from Phase I are then converted into a functional CNN architecture randomly and can be represented as $\mathbf{x}_i = (\text{Convo}, \text{Pool}, \text{Convo}, \text{Pool}, \ldots, \text{FC})$. In TPEvo-CNN, Convo layers select the filter size c_f and the number of filters c_n from a predefined range, while the stride size c_s is set as 1×1. Similarly, for Pool layers, the kernel size p_k is chosen from a predefined range, while the stride size p_s is considered as 1×1 respectively in this study. For each pooling layer, the pooling type (p_{type}) is randomly assigned as either average or max pooling. For the FC layer, the number of neurons f_n is selected from the user-specified range. Thus, each element in the vector representation $\mathbf{x}_i$ refers to a layer type and its associated hyper-parameter settings. The limits of the hyper-parameters for each layer type are furnished in Section IV-B.

2) FITNESS EVALUATION

Every individual is evaluated through a fitness or objective function which provides the individual's rank or quality. In this phase, an individual's fitness is assessed in the same way as in Phase I by considering the classification accuracy of the CNN model with respect to the validation dataset. Algorithm 3 outlines the fitness evaluation process of Phase II. First, each generated CNN model must be trained using the given training dataset D_{train}. The Xavier initialization [55], rectified linear unit (ReLU) [56], and Adam optimizer [57] are used in this work to initialize weights, activation function, and optimizing parameters of the CNN model.

3) CROSSOVER

Crossover is designed to make the population diverse by sharing information among individuals. The performance of any suggested algorithm can be enhanced by implementing an efficient information-sharing tactic. Here, a crossover strategy is proposed to explore the hyper-parameters search space. The overview of the devised crossover strategy is presented in Algorithm 4.

Algorithm 4 Proposed Crossover Strategy

```
Input: population size: $NP$, individuals at generation
       $t$: $P_t$, crossover probability: $P_c$.
Output: two offspring $o_1$ and $o_2$.
$o_1, o_2 \leftarrow \emptyset$;
for $i = 1$ to $N$ do
    for $j = 1$ to 10 do
        $p_1, p_2$ ← choose two individuals randomly
          from $P_t$;
        $\delta$ ← compute the different between $p_1$ and $p_2$;
    end
end
$p_1, p_2 \leftarrow \max(\delta)$;
// selected parent individuals
$r$ ← generate a random number between (0,1);
if $r \le P_c$ then
    $len_1$ ← compute the length of $p_1$;
    $len_2$ ← compute the length of $p_2$;
    for $n = 1$ to min ($len_1$,$len_2$) do
        K($n$) ← stores the genes information of $p_1$
          randomly from [1,$len_1$);
        L($n$) ← stores the genes information of $p_2$
          randomly from [1,$len_2$);
    end
    // exchange the information
       between two parent individuals
    // K($n$) and L($n$) same layer type
    for $i = 1$ to $K$ do
        for $j = 1$ to $L$ do
            $temp$ ← K($i$);
            K($i$) ← L($j$);
            L($j$) ← $temp$;
        end
    end
    $o_1, o_2 \leftarrow p_1, p_2$;
else
    $o_1, o_2 \leftarrow p_1, p_2$;
end
Return $o_1, o_2$;
```

First, two individuals are chosen randomly from the population, and the Hamming distance (δ) between them is calculated. This process is repeated ten times, selecting the two individuals with the maximum δ value. The maximum δ value can help to select more diverse individuals from each other in terms of its hyper-parameters to perform the evolution process. The selected individuals are treated as parent individuals p_1, p_2. An example of the parent selection approach is shown in Figure 4. After the parent selection, the crossover is performed on the parents p_1 and p_2, if a

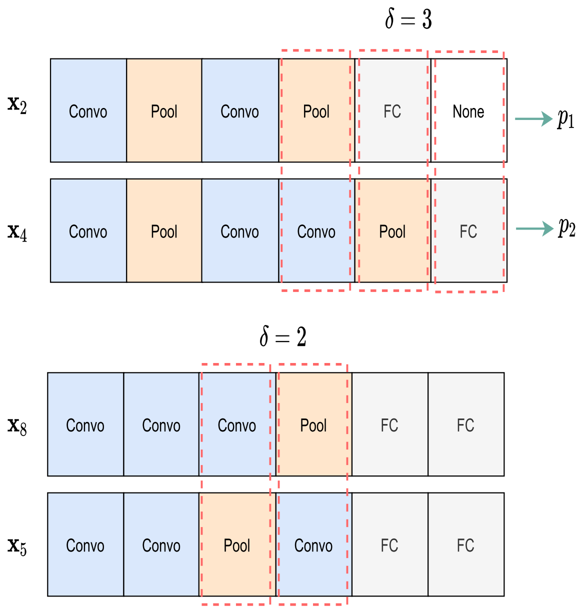

FIGURE 4. Example of proposed parent selection strategy in crossover operation.

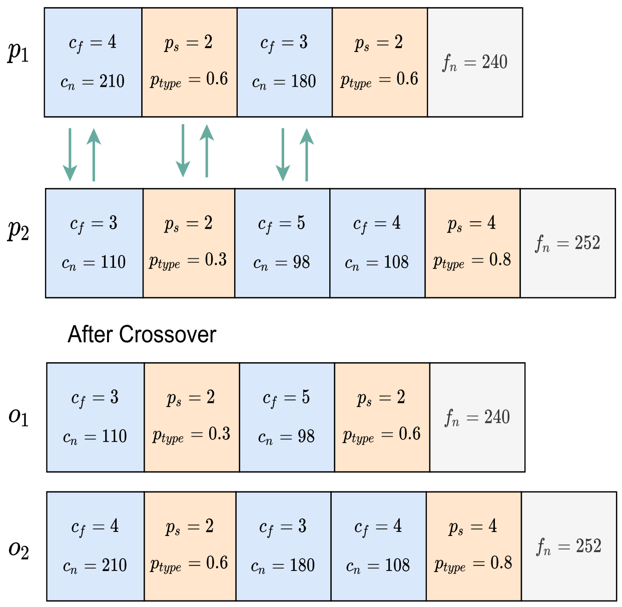

FIGURE 5. Example of proposed crossover operation in HPS phase.

random number $r \in (0, 1)$ is less than equal to crossover probability P_c. If not, then p_1 and p_2 will be considered as offspring o_1 and o_2 for the next generation, respectively. To perform the crossover, first lengths len_1 and len_2 of two parents p_1 and p_2 are computed. After that, two arrays (K) and (L) with size $min(len_1, len_2)$ are defined to store the gene information of the parents p_1 and p_2. The gene information is structured with the layer types and it's associated hyper-parameters values, which will be shared within the two arrays (K) and (L) using a variable *temp*. This analogy indicates a multi-point crossover between two parents and provides more diversity than a single-point crossover [59], [60]. Finally, two new offspring o_1, o_2 are generated from the two parents p_1, p_2 after sharing the hyper-parameters information as shown in Figure 5.

4) MUTATION

Mutation can help for more diversification in the population and also restrict to get stuck at local optimal during the execution. A mutation operation is performed when a random number $r \in (0, 1)$ is less than equal to the mutation probability P_m. First, a position is chosen randomly from the length of selected offspring **o**. The position denoted a layer which would be among Convo, Pool and FC layers. After that, another random number $\gamma \in [1, 3]$ is generated to perform the mutation operation. If $\gamma = 1$, only one hyper-parameter of the particular layer type is selected randomly and changed arbitrarily from the pre-defined ranges, as elaborated in Section IV-B. No change is made to any hyper-parameter setting of the selected layer if $\gamma = 2$. Finally, in case of $\gamma = 3$, all the hyper-parameters of the selected layer types will be modified randomly by taking their value from

Algorithm 5 Proposed Mutation Strategy

```
Input: an offspring: o, mutation probability: $P_m$.
Output: mutated offspring o.
$\gamma \leftarrow \emptyset$;
$r \leftarrow$ generate a number between (0,1);
while $r \leq P_m$ do
    $i \leftarrow$ randomly select integer between 1 to length (o);
    $\gamma \leftarrow$ randomly select integer between [1,3];
    if $\gamma = 1$ then
        $o(i) \leftarrow$ one hyper-parameter of the particular layer types will be changed randomly;
    else if $\gamma = 2$ then
        $o(i) \leftarrow$ no change is made in hyper-parameter setting;
    else
        $o(i) \leftarrow$ all the hyper-parameters of the particular layer types will be changed ;
    end
end
Return mutated offspring o;
```

Section IV-B. Algorithm 5 provides the proposed mutation operation for only one offspring individual.

5) SELECTION FOR NEXT GENERATION

Here, an elitism-based selection mechanism is suggested to construct a constant population size of individuals for the following generation. Generally, tournament or roulette wheel selection mechanisms are used to build the population for the next generation [61]. However, these techniques may skip informative individuals. Therefore, selecting an individual is critical after the mutation operation in the GA context. Thus, Algorithm 6 presents the introduced elitism-based selection

Algorithm 6 Proposed Selection Strategy

```
Input: parent population at $t$ generation: $P_t$ and
       offspring population at $t$ generation: $Q_t$.
Output: the elitism-based generation at $t+1$: $P_{t+1}$.
$P_{t+1} \leftarrow \emptyset$;
$A_t \leftarrow P_t \cup Q_t$;
$B_t[N] \leftarrow$ sort $(A_t)$ based on the maximum
  classification accuracy and $N$ represents size of array
  equal to $A_t$;
for $i = 1$ to 50% of $P_t$ do
    $P_{t+1} \leftarrow P_{t+1} \cup B_t(i)$ ;
    $A_t[N-1] \leftarrow$ item $B_t(i)$ is removed from $A_t$;
end
while $| P_{t+1} | < | P_t |$ do
    $p \leftarrow$ select an individual from $P_t$ by using a
      binary tournament selection method;
    $P_{t+1} \leftarrow P_{t+1} \cup p$;
end
Return $P_{t+1}$;
```

approach. First, an auxiliary population A_t is constructed by combining the current population P_t and offspring Q_t at generation t. Next, A_t is sorted in descending order based on the classification accuracy and stored in an array B_t. After that, the top 50% individuals are considered for the following population P_{t+1} from B_t and these individuals are removed from A_t. To keep the population size *NP*, remaining individuals for P_{t+1} will be selected from P_t using the famous binary tournament method [61] of GA. Finally, *NP* individuals are generated for the population P_{t+1}.

E. PUTTING ALL TOGETHER

The Algorithm 7 outlines the entire framework of the suggested TPEvo-CNN method. First, the ADS phase is conducted for optimal architecture search to determine the number of Convo, pool, and FC layers in an individual of population *NP* as shown in Algorithm 1. Afterwards, optimal hyper-parameters are set according to the Algorithm 2 for the CNN architecture generated in Phase I. Finally, the best CNN (i.e., $\mathbf{x}_{best}$) is nominated based on its maximum classification accuracy. Further, the $\mathbf{x}_{best}$ CNN model architecture is trained on original training data and finally tested on the test data to collect the accuracy of the proposed framework for performance evaluation.

F. COMPUTATIONAL TIME COMPLEXITY OF TPEVO-CNN

The computational time complexity (CC) mainly depends on the specific problem analysis and fitness evaluation part of an algorithm. The CC of the TPEvo-CNN is computed from several components of the algorithm that are in two phases. To enhance the representation, let N denote the population size, D indicate the number of dimensions, and T represent the maximum number of generations in the algorithm. The CC of population initialization is in $O(N \times D)$, while the CC of the proposed ADS and HPS are in $O(T \times N \times D)$. The CC for individual selection of the next generation is in $O(T \times N)$. Therefore, the total CC of TPEvo-CNN is in $O((2 \times T \times N \times D) + (N \times D) + (T \times N))$.

Algorithm 7 Framework of the TPEvo-CNN Method

```
Input: training dataset: $D_{train}$, test dataset: $D_{test}$.
Output: best individual $X_{best}$ and its test result.
Perform architecture depth search (ADS) in Phase I
  according to Algorithm 1;
Perform hyper-parameters search (HPS) in Phase II
  according to Algorithm 2 ;
$\mathbf{x}_{best} \leftarrow max(fitness)$; // best CNN model
// train the best CNN model
for $e_1 \leq train\_epoch$ do
    $\mathbf{x}_{best}^{train} \leftarrow$ train $(\mathbf{x}_{best}, D_{train})$;
end
// test the best CNN model
for $e_2 \leq test\_epoch$ do
    $f(\mathbf{x}_{best}) \leftarrow$ test $(\mathbf{x}_{best}^{train}, D_{test})$;
end
Return $\mathbf{x}_{best}$ and its test result;
```

IV. EXPERIMENTAL DESIGN

This section details the six medical image datasets used in our study, as well as the parameter settings of the proposed method and the system configuration for the experiments.

A. DATASETS

The experiment is conducted on six publicly available medical image datasets to evaluate the effectiveness and efficiency of the proposed framework. The datasets include four types of COVID-19 datasets [46], [62], [63], [64], which are categorized on image types and different classes collected from the various sources, and also on pneumonia [65] and skin cancer [66] datasets. The four types of COVID-19 datasets are chest X-ray images with binary class,[1] chest X-ray images with multi-class,[2] Computer Tomography (CT) images[3] and chest radiography images.[4] For the simplicity, these four datasets are denoted as COVID-19x$_1$, COVID-19x$_2$, COVID-19-CT and COVID-19-Radiography respectively in this work. The samples of each COVID-19 dataset are divided into a training set and a testing set in a ratio of 7 : 3 due to the large number of samples in these datasets. In contrast, the lower number of samples compared to the COVID-19 datasets, pneumonia, and skin cancer datasets are divided in a ratio of 9 : 1 for training and testing purposes. The details of each dataset, such as dataset name, data type, input size, total samples, training samples, test samples, and the number of classes, are provided in Table 3. Moreover,

[1]https://data.mendeley.com/datasets/8h65ywd2jr/3

[2]https://github.com/muhammedtalo/COVID-19

[3]https://www.kaggle.com/datasets/plameneduardo/sarscov2-ctscan-dataset?select=COVID

[4]https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

TABLE 3. An overview of six datasets used in the experimental study of TPEvo-CNN approach.

Dataset Name	Data type	Input size	Total samples	#training	#test	#classes
COVID-19x$_1$	X-ray images (Chest)	180 × 180	9544	6679	2865	2
COVID-19x$_2$	X-ray images (Chest)	180 × 180	1125	787	338	3
COVID-19-CT	CT images	180 × 180	2481	1736	745	2
COVID-19-Radiography	Radiography images (Chest)	180 × 180	21165	14813	6352	4
Pneumonia	X-ray images (Chest)	180 × 180	4657	4190	467	2
Skin Cancer	Skin lesion images	180 × 180	3297	2966	331	2

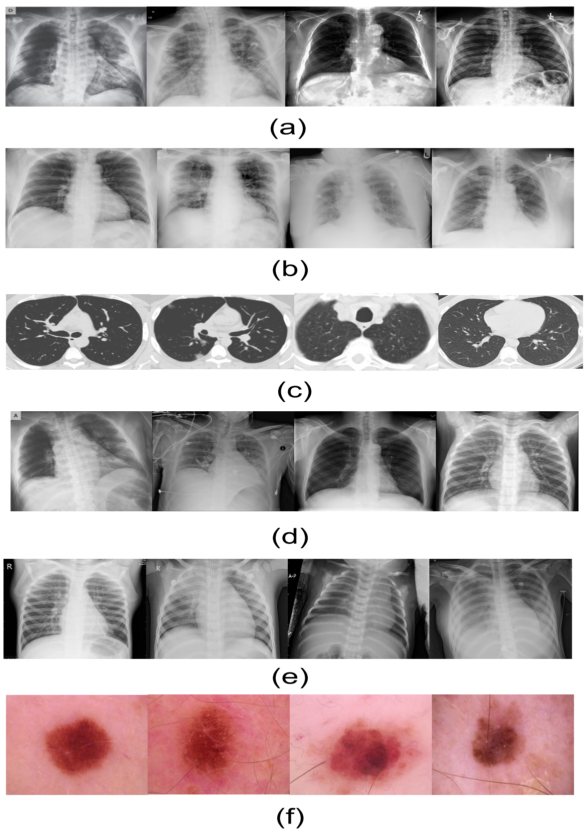

FIGURE 6. Few sample images of (a) COVID-19x$_1$ (b) COVID-19x$_2$ (c) COVID-19-CT (d) COVID-19-Radiography (e) Pneumonia and (f) Skin Cancer datasets.

a few random samples of the four types of COVID-19, pneumonia and skin cancer images are presented in Figure 6.

Moreover, the six datasets are pre-processed through image pre-processing techniques for compatibility with the proposed CNN model, the pre-trained and NAS-based CNN models. In our experiment, before feeding each training example to the model, all the images in the datasets are resized to a uniform resolution of 180 × 180. This standardization eliminates the variations in image sizes and aspect ratios, ensuring consistent inputs for the CNN models. Subsequently, each image is normalized by subtracting its mean and then dividing the resulting value by standard deviation (std), as stated in Eq. (4).

$$\text{normal}(x) = \frac{x - \text{mean}(x)}{\text{std}(x)}. \tag{4}$$

In this context, "x" represents a single image and "std(x)" denotes the standard deviation of the pixel values in that image with a maximum possible value of 255. This normalization step reduces the sensitivity to the pixel intensity variations, aiding faster convergence and improving the overall performance of the CNN models. Additionally, we further employ one-hot encoding to transform categorical class labels into binary vector representations. Together, these prepossessing steps enhance the fairness and consistency of model comparisons, leading to improved convergence and reliable performance across the different datasets.

B. PARAMETER SETTINGS

In this study, parameters of the proposed method are considered based on the recommendation of the existing literature as DE [40], GA [61], and deep learning [67] along with the limited computational resources in our hand. The associated parameter settings used in the proposed TPEvo-CNN are provided in Table 4. Here, the population size is fixed at 20 throughout the evolution processes, while maximum generations are set at 10 and 20, respectively, for Phase I and II. ADS is responsible for finding the optimal depth of the CNN architecture, where each layer type varies within a pre-defined range. The range of the Convo layer varies from 3 to 8, the Pool layer from 1 to 4, and the FC layer from 1 to 3, respectively. In Phase I, for DE the F and the CR are set as 0.6 and 0.4, respectively, while in GA, the P_c and the P_m values are set as 0.9 and 0.7, respectively, for Phase II in our study.

On the other hand, HPS is concerned about the optimal hype-parameter settings of the generated CNN structures. In this regard, hyper-parameter ranges can be different from layer to layer. For the Convo layer of the CNN, the range of the filter size (c_f) lies from 3 × 3 to 7 × 7, the stride size is set as 1 × 1, and the number of filters (c_n) is selected from the range of 3 to 256. In case of Pool layer, kernel size (p_k) alters from 2 × 2 to 4 × 4 and stride size is fixed at 2 × 2. Moreover, the pooling type (p_{type}) varies based on randomly generated number defined as:

$$p_{type} = \begin{cases} avg\_pooling, & rand(0,1) \leq 0.5, \\ max\_pooling, & Otherwise. \end{cases} \tag{5}$$

Finally, the number of neurons (f_n) for FC layers ranges from 1 to 300.

Furthermore, to train the CNN models, we utilized ReLu or rectified linear unit [56] as activation function, Xavier weight

TABLE 4. Initial parameter settings for TPEvo-CNN.

Parameter	Setting
Initialization	
Population size	20
# DE generation	10
# GA generation	20
# Convo	3 to 8
# Pool	1 to 4
# FC	1 to 3
DE (Phase I)	
Scaling factor (F)	0.6
Crossover rate (CR)	0.4
GA (Phase II)	
Crossover probability (P_c)	0.9
Mutation probability (P_m)	0.7
Structure of CNN	
Filter size of Convo layer	3 to 7
Stride size of Convo layer	1
# filter in Convo layer	3 to 256
Kernel size of Pool layer	2 to 4
Stride size of Pool layer	2
Type of Pool layer	Average or Max
# neurons in FC layer	1 to 300
CNN Training	
Initialization of Weight	Xavier
Activation function	ReLu
Optimization function	Adam
Fitness function	Categorical cross-entropy
Dropout connection rate	0.5
Learning rate	0.001
# batch size	32
# epochs for single individual	1
# epochs for final individual	100

initialization [55], and Adam optimization [57] algorithm due to their popularity in deep learning, respectively. The popular categorical cross-entropy [54] is used to compute the classification accuracy of generated CNN models. The proposed approach maintains a fixed learning rate of 0.001 during execution. To speed up the training speed, we consider a batch size of 32 for batch normalization (BN) and 50% dropout connection, as recommended in [68] and [69]. Moreover, we set 20% of the training images as a validation sample and conducted the fitness evaluation on it by considering a single epoch number as recommended in most of the existing NAS-based CNN model [29], [34] for evaluation. Nevertheless, the final CNN model architecture obtained through the TPEvo-CNN model is trained and tested by considering 100 epoch numbers to evaluate the performance in our study.

C. SYSTEM CONFIGURATION

The TPEvo-CNN algorithm is developed using Python 3.6.9, Tensorflow 1.15.0, and Keras libraries 2.3.1. The Experiment is Conducted on a 7820 Dell Precision Workstation equipped with Ubuntu 18.04, 2.5GHz Intel 5th Generation Processor, RAM 16GB, and Nvidia Quadro Graphics Card (RTX5000).

V. RESULTS AND DISCUSSION

In this section, we present the experimental results of the proposed framework and compare them with the diverse state-of-the-art CNN models on six medical image datasets. The results are analyzed and validated based on the different performance evaluation metrics, such as classification metrics, confusion matrix, statistical analysis, and radar plots explained in the following subsections. Moreover, the achieved top ten CNN model architectures through the TPEvo-CNN are discussed with respect to all the experimented datasets.

A. CLASSIFICATION METRICS

The proposed TPEvo-CNN is evaluated on various classification metrics, including accuracy, precision, recall, and F1-score [70]. Accuracy mentions the ratio of correctly predicted samples to the total number of samples in a dataset and is defined as:

$$\text{Accuracy} = \frac{\mathit{True\ Positive} + \mathit{True\ Negative}}{\mathit{Total\ Samples}}. \tag{6}$$

True Positive denotes the positive class that the model accurately predicts, whereas True Negative represents the outcomes when a model correctly predicts the negative class.

Precision is a classification metric that measures the ratio of True Positive samples to the total number of samples predicted as positive. It is defined as the number of *TruePositive* samples divided by the sum of True Positive and False Positive samples, as follows:

$$\text{Precision} = \frac{\mathit{True\ Positive}}{\mathit{True\ Positive} + \mathit{False\ Positive}}. \tag{7}$$

Here, False Positive denotes the result predicted wrongly as a positive class against the negative class of the actual class.

The recall is a metric that measures the number of True Positive samples predicted as positive by the model from the total number of True Positive and False Negative samples, which can be formulated as:

$$\text{Recall} = \frac{\mathit{True\ Positive}}{\mathit{True\ Positive} + \mathit{False\ Negative}}. \tag{8}$$

False Negative represents the model incorrectly predicting a negative class while the actual class is positive.

On the other hand, the Precision-Recall metrics for multi-class problems can be calculated with averaging technique of classes by using the same formula as defined in Eq. (7) and (8). First, the multi-class problem is divided into binary-class problems using the One-vs-Rest class. For example, if we have a k class, the k class can be binarized into k tasks. It can be represented by One-vs-Rest averaging technique as the 1st class Vs remaining $k-1$ classes $(2, 3, \ldots, k-1)$. Therefore, the Precision-Recall metric for the kth class can be computed by the following Eq.(9) and (10).

$$\text{Precision}_{\,k} = \frac{\mathit{TruePositive}_k}{\mathit{TruePositive}_k + \mathit{FalsePositive}_k}, \tag{9}$$

$$\text{Recall}_k = \frac{\mathit{TruePositive}_k}{\mathit{TruePositive}_k + \mathit{FalseNegative}_k}, \tag{10}$$

where $\mathit{TruePositive}_k$ and $\mathit{FalsePositive}_k$ denotes the true positive value of the kth class and the sum of the false positive values of the first $k-1$ number of classes corresponding to the kth class. Similarly, $FalseNegative_k$ is the sum of the false negative values of the first $k-1$ number of classes corresponding to the kth class. Finally, the Precision-Recall metrics for multi-class problems with respect to a model are calculated by taking the average of the corresponding metrics for all the classes and defined as:

$$\text{Precision} = \frac{\sum_{k=1}^{K} \text{Precision}_k}{K} \tag{11}$$

$$\text{Recall} = \frac{\sum_{k=1}^{K} \text{Recall}_k}{K} \tag{12}$$

The F1-score is the harmonic mean of precision and recall metrics. It is the same for both binary and multi-class classification problems and mathematically defined in Eq. (13).

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

1) COMPARISON WITH STATE-OF-THE-ART COVID-19 MODELS

The efficiency and effectiveness of the proposed TPEvo-CNN is compared with the state-of-the-art CNN models regarding classification metrics such as accuracy, precision, recall and F1-score concerning four COVID-19 datasets categorized based on image types and a number of classes. We have considered the compared CNN models such as COVID-Net [44], MobileNet [45], DarkCovidNet [46], CNN-SA [47], CoroNet [48], GSA-DenseNet121-COVID-19 [49], DRENet [50], Deep-chest [51], Bi-CNN-D-C [52], CNGOD [53] because these models have experimented on the same category of COVID-19 datasets as carried in our work. Notably, the reproducibility of the developed models for the purpose of comparison may only sometimes be feasible due to the unavailability of the code implementation, computational resources, and the requirements of high execution times as recommended in the literature [12], [13]. For this reason, we have collected only the results of the compared CNN models from the original work and reproduced them in our study. Moreover, their experimental setups of the compared models, as mentioned in the original work are summarized in Table 5.

The accuracy, precision, recall and F1-score values of the proposed method and the compared models (as reproduced directly from the original work) are shown in Table 6 for the COVID-19 datasets concerning image types and numbers of classes. Here, '−' sign reveals that the model has not been experimented on the corresponding dataset. The model COVID-Net obtained an accuracy, precision, recall, and F1-score as 83.5%, 82.3%, 87.23% and 83.92%, respectively, in four class category classification problems. The MobileNet obtained better results with three class classification problems, and associated accuracy, precision, and recall are 96.78%, 96.46%, and 98.66%, respectively. The model DarkCovidNet produced more accurate results on two categories than three, as in column 5 to 8 of Table 6. The CNN-SA model is evaluated on a two-class classification problem and achieved 95% prediction result on all classification metrics. The CoroNet performed better for the binary classification than the multi-class classification, as shown in columns 5, 6, and 7 in Table 6. The GSA-DenseNet121-COVID-19 model achieved accuracy, precision, recall, and F1-score are 98.38%, 98.5%, 98.5% and 98%, respectively, in case of two class classification problem. The model DRENet produced 93% in all four classification metrics in three class classification challenges. The model Deep-chest generates the same results, 98.05%, as its classification accuracy and recall, whereas the precision and F1-score are 98.43% and 98.24%, respectively. The recent Bi-CNN-D-C model can generate 96.81% accuracy for a binary class classification problem. Finally, the model CNGOD gives the result as accuracy, precision, recall, and F1-score as 96.83%, 96.52%, 96.63% and 97.08%, respectively, for a three-class classification challenge. The proposed TPEvo-CNN produces better classification results on COVID-19x$_1$ dataset among all four COVID-19 datasets, which are represented in the bold form in Table 6. The generated CNN model also performs well against most state-of-the-art COVID-19 models. TPEvo-CNN achieved 98.2% classification accuracy, second after the model GSA-DenseNet121-COVID-19 reported 98.5% accuracy, which seems to be the same. The suggested model is fourth regarding the 98% recall result, behind CoroNet (two classes) and GSA-DenseNet121-COVID-19 model. Finally, the F1-score attained a 98% result and shared the second position with the GSA-DenseNet121-COVID-19 model after the CoroNet with two classes and the Deep-chest model. Finally, we can observe that the proposed model is competitive compared to the COVID-19 CNN models.

2) COMPARISON WITH PRE-TRAINED AND NAS-BASED CNNS

Further, to validate the performance of the proposed model, we have considered the state-of-the-art pre-trained and NAS-based CNN models for comparison. Due to time and resource limitations, all these models, including our proposed method, experimented on only one type of COVID-19 dataset that contains chest X-ray images and two classes. Moreover, the details configuration, i.e., that is architectural information and parameter settings are kept unchanged for pre-trained models such as LeNet-5 [5], AlexNet [6], VGG-16 [7], VGG-19 [7], Inception-V3 [8], ResNet-50 [9], DenseNet-201 [10], DenseNet-121 [10]. On the other hand, the architectural information is borrowed from existing literature of NAS-based CNN models, including MetaQNN [21], NASNet [22], EAS [23], GeNet [26], EENA [27], EfficientNet-B7 [28], CGP-CNN [29], and psoCNN [34]. In contrast, parameter settings for these models are kept the same as in the proposed method. Finally, the TPEvo-CNN, pre-trained, and NAS-based CNN models are executed in the same system configuration by considering the same performance

TABLE 5. Summary of the experimental setups of the compared models as mentioned in the literature. 'x' indicates unavailability of information.

Method	Data Augmentation	Activation Function	Optimizer	Learning Rate	Batch Size	Epoch Size	Dropout	Run Environment
COVID-Net [44]	Yes	ReLu	Adam	$2e^{-4}$	64	22	x	Keras with Tensorflow
MobileNet [45]	x	ReLu	Adam	x	64	10	Yes	x
DarkCovidNet [46]	x	ReLu	Adam	x	64	100	Yes	x
CNN-SA [47]	Yes	x	Adam	0.0002	64	2000	x	x
CoroNet [48]	x	x	Adam	0.0001	10	80	x	Keras with Tensorflow 2.0
GSA-DenseNet121-COVID-19 [49]	x	x	Adam	x	64	30	Yes	Keras with Tensorflow
DRENet [50]	Yes	ReLu	Adam	0.001	28	10	x	x
Deep-chest [51]	Yes	x	Adam	0.0006 to 0.0009	x	10	x	x
Bi-CNN-D-C [52]	x	ReLu	Adam	0.1	128	50	x	x
CNGOD [53]	Yes	ReLu	Adam	x	32	100	x	x

evaluation strategies such as accuracy, precision, recall, and F1-score.

The classification metrics, the number of parameters and the execution time of the proposed TPEvo-CNN and compared models on COVID-19x_1 are presented in Table 7 while the best results are shown in boldface. It can be observed that the proposed method performs better than all other NAS models in terms of accuracy, precision, recall, and F1-score, which are 98.2%, 98.5%, 98.8%, and 98.8%, respectively. TPEvo-CNN takes 2.21 million in parameters after only the CGP-CNN model, which is 2.01 million. Except for psoCNN and CGP-CNN, the optimal CNN architecture generated by the proposed method takes about 8407.95 seconds to produce the final output on the test dataset.

In addition, the performance of the proposed method is also evaluated in terms of classification metrics using an augmented COVID-19x_1 dataset. For this purpose, a random transformation technique is applied among shear, zoom, rotation, shifting, and horizontal flipping to increase the number of samples in the dataset. The size of the COVID-19x_1 dataset is increased from 6679 (2830 COVID-19 samples and 3849 normal samples) training samples to 20037 (8490 COVID-19 samples and 11547 normal samples). Similarly, test samples are increased from 1214 to 3642 COVID-19 and 1651 to 4953 normal samples. Finally, the results of classification metrics for the TPEvo-CNN model with data augmentation (TPEvo-CNN + Augment) are included at the bottom of Table 7. The obtained results reveal the superiority compared to other models along with the proposed method without augmentation concerning accuracy (98.8%), precision (98.7%), recall (98.4%), and F1-score (98.5%). However, it has produced 3.59 million parameters and taken 13680.62 seconds as execution time, which is computationally expensive.

Moreover, the effectiveness and efficiency of the TPEvo-CNN model is measured on pneumonia and skin cancer datasets. Tables 8 and 9 summarise the test outcomes, including classification accuracy, total parameters, and execution time for the compared models. In the case of the pneumonia dataset, as given in Table 8, the proposed method obtained a better 79.4% classification accuracy than all popular NAS models and placed fifth with 4.89 million of total parameters. The generated optimal CNN model takes approximately 5295.88 second to get the final test result after the existing psoCNN and CGP-CNN models. The proposed model achieves a better accuracy 83.4% for the skin cancer dataset than all NAS-based popular CNN models. The optimal CNN model generates 4.53 million parameters and stands fifth after the models psoCNN, CGP-CNN, LeNet-5, and EfficientNet-B7. The optimal CNN architecture in TPEvo-CNN takes 4873.93 seconds to generate the final test accuracy, the fourth in execution time after the CGP-CNN, psoCNN and MetaQNN models. In addition, Figure 7 also depicts the precision, recall, and F1-score of the generated optimal CNN models concerning the epochs number in both training and test cases individually for COVID-19x_1, pneumonia, and skin cancer datasets. Thus, the proposed model demonstrated competitive results for the three datasets compared to pre-trained and NAS-based CNN models.

B. CONFUSION MATRICES

The obtained confusion matrices are shown in Figure 8 individually in each of the six datasets. For the COVID-19x_1 dataset, the top left of the confusion matrix represents the

TABLE 6. The performance of the TPEvo-CNN compared to the results of other COVID-19 CNN models. A dash ('–') is used if no test results are reported. The results of the TPEvo-CNN method are highlighted in bold.

Method		Total original samples	Data type	# classes	Accuracy	Precision	Recall	F1-Score
COVID-Net [44]		310 Normal 330 Pneumonia Bacterial 327 Pneumonia Viral 284 COVID-19	Chest X-ray	4	83.5%	82.3%	87.23%	83.92%
MobileNet [45]		504 Healthy 714 Pneumonia (400 bacterial + 314 viral) 224 COVID-19	Chest X-ray	3	96.78%	96.46%	98.66%	-
DarkCovidNet [46]		500 No-Findings 125 COVID-19	Chest X-ray	2	98.08%	98.03%	95.13%	96.51%
DarkCovidNet [46]		500 No-Findings 500 Pneumonia 125 COVID-19	Chest X-ray	3	87.02%	89.96%	85.35%	87.37%
CNN-SA [47]		721 Normal 403 COVID-19	Chest X-ray	2	95%	95%	95%	95%
CoroNet [48]		310 Normal 330 Pneumonia Bacterial 327 Pneumonia Viral 284 COVID-19	Chest X-ray	4	89.6%	90%	89.92%	89.8%
CoroNet [48]		310 Normal 657 Pneumonia (330 bacterial + 327 viral) 284 COVID-19	Chest X-ray	3	95%	95%	96.9%	95.6%
CoroNet [48]		310 Normal 284 COVID-19	Chest X-ray	2	99%	98.3%	99.3%	98.5%
GSA-DenseNet121-COVID-19 [49]		99 COVID-19 Positive 207 COVID-19 Negative	Chest X-ray	2	98.38%	98.5%	98.5%	98%
DRENet [50]		86 Healthy 100 Bacterial Pneumonia 88 COVID-19	CT images	3	93%	93%	93%	93%
Deep-chest [51]		3500 Normal 5856 Pneumonia 20000 Lung Cancer 4320 COVID-19	Chest X-ray + CT images	4	98.05%	98.43%	98.05%	98.24%
Bi-CNN-D-C [52]		310 Normal 284 COVID-19	Chest X-ray	2	96.81%	-	-	-
CNGOD [53]		125 Normal 500 Pneumonia 500 COVID-19	CT images	3	96.83%	96.52%	96.63%	97.08%
TPEvo-CNN	**COVID-19x$_1$**	**4044 COVID-19 5500 Normal**	Chest X-ray	**2**	**98.2%**	**98.5%**	**98%**	**98%**
	COVID-19x$_2$	125 COVID-19 500 Normal 500 Pneumonia	Chest X-ray	3	97.3%	96%	95.3%	95.6%
	COVID-19-CT	1252 COVID-19 1229 Normal	CT images	2	95.4%	95.5%	95.5%	95%
	COVID-19-Radiography	3616 COVID-19 6012 Lung_opacity 10192 Normal 1345 Viral Pneumonia	Chest radiographs	4	97%	94.8%	95.3%	94.7%

correctly predicted samples *True Positive* for the COVID-19 class, while the bottom right represents the number of correctly predicted samples for the normal class as *True Negative*. The top right indicates the samples that belong to the COVID-19 class but can be treated as normal (*False Positive*). At the same time, the bottom left defines the number of samples that belong to the normal class but are treated as COVID-19 (*False Negative*). The same follows for all other five datasets, including COVID-19x$_2$, COVID-19-CT, COVID-19-Radiography, pneumonia, and skin cancer dataset.

The confusion matrix results provide valuable insights into the performance of the proposed TPEvo-CNN algorithm across different datasets. In the COVID-19x$_1$ dataset, the algorithm achieved a high number of correct predictions of image samples (2814) compared to a relatively less number of incorrect prediction samples (51). Similarly, for the COVID-19x$_2$ dataset, 329 samples are correctly predicted, and only nine are incorrectly predicted. Moving on to the COVID-19-CT dataset, the proposed algorithm demonstrated 711 correct predicted samples and only 34 incorrect predictions. The multi-class dataset (COVID-19-Radiography) showcased the algorithm's efficiency by accurately predicting 6160 image samples while producing only 192 incorrect predictions. The proposed model predicted 371 samples correctly for pneumonia cases, while only 96 samples generated incorrect predictions. Finally, the algorithm performed well in the skin cancer dataset by accurately predicting 276 samples against only 55 incorrect predictions. Overall, these results suggest that the proposed TPEvo-CNN shows promising performance for the image classification of six medical image datasets.

C. STATISTICAL ANALYSIS

We have performed a statistical analysis on each experimental dataset to assess the proposed model's performance. The

TABLE 7. A comparative analysis of models for the COVID-19x$_1$ dataset based on accuracy, precision, recall, F1-score, parameter number (in millions), and execution time (in seconds).

Method	Model	Accuracy	Precision	Recall	F1-Score	#Parameters	Execution time
Pre-trained Model	LeNet-5 [5]	80.7%	76.5%	75%	76%	3.78	8964.91
	AlexNet [6]	92.3%	90.5%	89.5%	90%	60.37	33098.07
	VGG-16 [7]	92.5%	91%	90%	90.5%	14.97	12786.94
	VGG-19 [7]	96.7%	97%	94.5%	96%	20.28	15851.77
	Inception-V3 [8]	91.8%	91%	88.5%	89.5%	22.85	11679.05
	ResNet-50 [9]	91%	90%	87.5%	89%	24.64	12139.05
	DenseNet-201 [10]	90%	88.5%	86.5%	87.5%	19.31	17686.92
	DenseNet-121 [10]	95.4%	96.5%	92.5%	94.5%	7.56	14343.94
NAS-based Model	MetaQNN [21]	87.3%	84.5%	83.5%	83.5%	11.18	8445.85
	NASNet [22]	94.7%	95%	92%	93%	4.81	9894.13
	EAS [23]	96%	96.5%	93.5%	95%	8.21	9969.54
	GeNet [26]	91.2%	90.5%	88%	89%	9.61	10304.51
	EENA [27]	93.6%	92.5%	91%	92%	8.49	9875.33
	EfficientNet-B7 [28]	94%	93.5%	91%	92.5%	4.71	11062.17
	CGP-CNN [29]	93.6%	92.5%	91.5%	92%	**2.01**	6032.91
	psoCNN [34]	96.2%	97%	94%	95%	2.32	**4807.95**
	TPEvo-CNN	**98.2%**	**98.5%**	**98%**	**98%**	2.21	8407.95
	TPEvo-CNN + Augment	98.8%	98.7%	98.4%	98.5%	3.59	13680.62

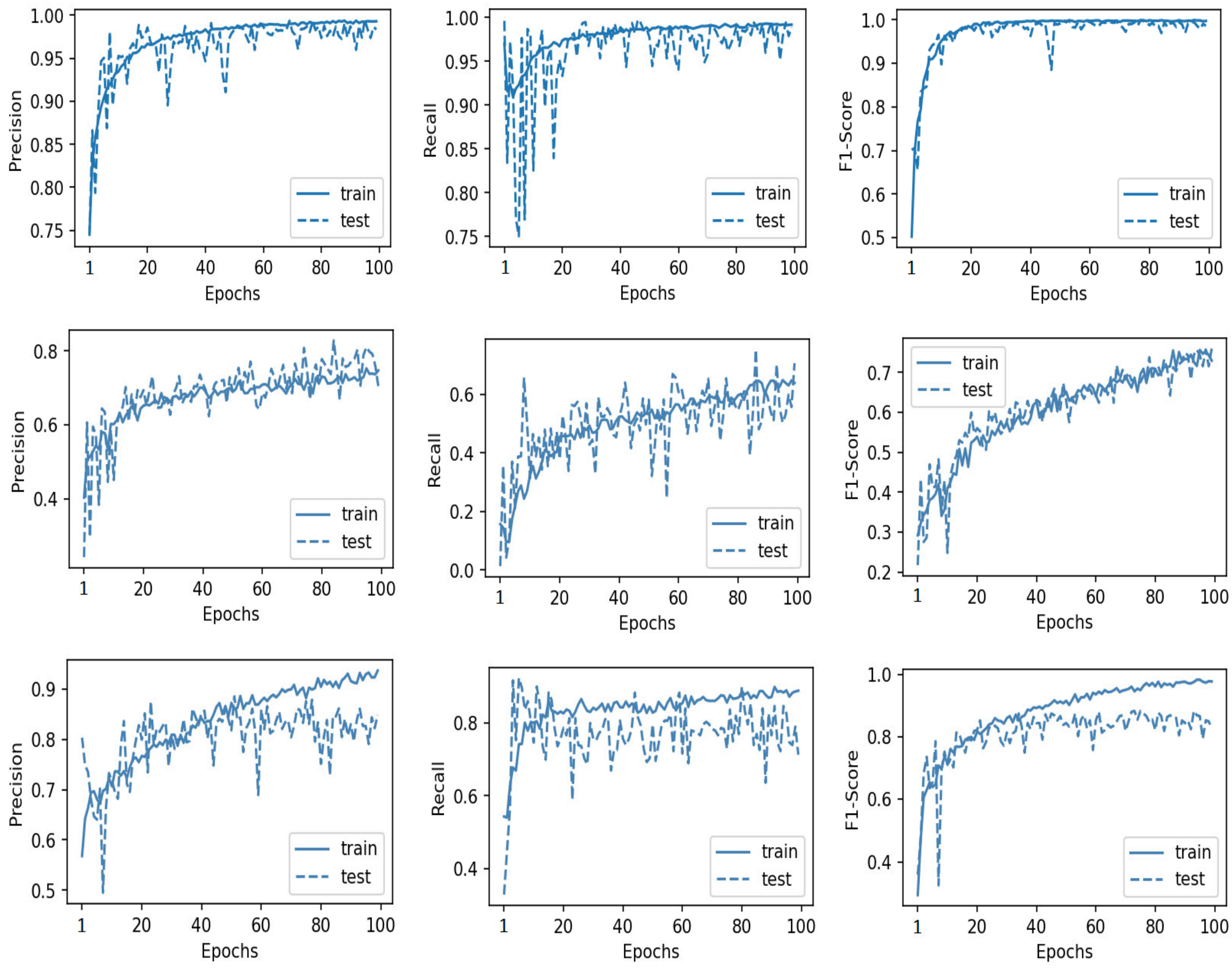

FIGURE 7. The generated precision, recall and F1-score by the best CNN model for COVID-19x$_1$ (top), Pneumonia (middle) and Skin Cancer (bottom) dataset.

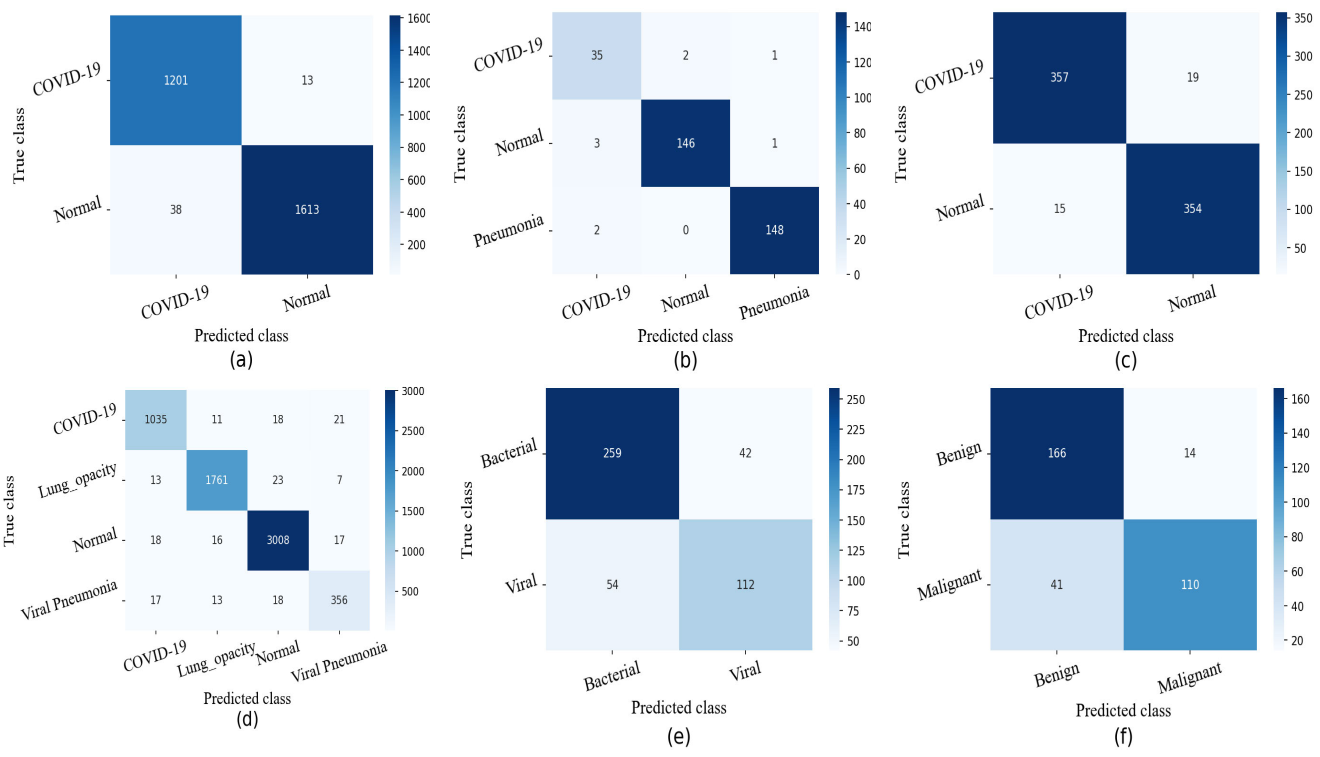

FIGURE 8. The resulting confusion matrices are shown for the following datasets: (a) COVID-19x$_1$, (b) COVID-19x$_2$, (c) COVID-19-CT, (d) COVID-19-Radiography, (e) Pneumonia and (f) Skin Cancer.

TABLE 8. Comparative analysis of TPEvo-CNN on the pneumonia dataset with respect to classification accuracy, number of parameters (in million) and execution time (in second).

Model	Accuracy	#parameters	Execution time
LeNet-5 [5]	67.9%	3.78	5821.60
AlexNet [6]	76.2%	60.37	10164.17
VGG-16 [7]	76%	14.97	13359.98
VGG-19 [7]	77%	20.28	14267.39
Inception-V3 [8]	74.3%	22.85	14823.07
ResNet-50 [9]	72%	24.64	15691.99
DenseNet-201 [10]	78%	19.31	13095.82
DenseNet-121 [10]	76%	7.56	12184.76
MetaQNN [21]	73.2%	10.78	10305.65
NASNet [22]	74.3%	4.98	6304.33
EAS [23]	76%	7.89	9061.14
GeNet [26]	71.7%	9.12	9112.09
EENA [27]	71.1%	9.97	7095.37
EfficientNet-B7 [28]	75.8%	4.38	6818.35
CGP-CNN [29]	74%	3.19	3943.45
psoCNN [34]	75.2%	**3.01**	**3853.86**
TPEvo-CNN	**79.4%**	4.89	5295.88

TABLE 9. Comparative analysis of TPEvo-CNN on the skin cancer dataset with respect to classification accuracy, number of parameters (in millions) and execution time (in seconds).

Model	Accuracy	#parameters	Execution time
LeNet-5 [5]	77%	3.78	4975.43
AlexNet [6]	82.5%	60.37	29711.37
VGG-16 [7]	80.7%	14.97	13718.01
VGG-19 [7]	82.2%	20.28	14010.10
Inception-V3 [8]	81.9%	22.85	13430.86
ResNet-50 [9]	81%	24.64	13797.47
DenseNet-201 [10]	82.8%	19.31	16989.65
DenseNet-121 [10]	81.6%	7.56	14667.49
MetaQNN [21]	79.8%	10.63	4120.85
NASNet [22]	83.1%	4.81	5816.17
EAS [23]	78.2%	8.09	8019.14
GeNet [26]	81.9%	8.98	8914.43
EENA [27]	73.3%	9.07	9207.62
EfficientNet-B7 [28]	79.2%	4.09	6529.41
CGP-CNN [29]	79.8%	2.89	**3097.89**
psoCNN [34]	82.8%	**2.56**	3582.03
TPEvo-CNN	**83.4%**	4.53	4873.93

Cohen Kappa (CK) score [71], a statistical method, is used to quantify the agreement between the actual and predicted class labels for a classification model. This score can be calculated with the help of the raw data and the values of the confusion matrix generated by the classifier. The CK score is defined mathematically as:

$$CK = \frac{\rho_o - \rho_e}{1 - \rho_e}. \tag{14}$$

Here, ρ_o is the observed agreement between actual and predicted values. It can be calculated as the ratio of the sum of diagonal values and the sum of non-diagonal values of any confusion matrix. Whereas ρ_e represents the expected agreement in terms of probability measure that the true values and false values agree by chance, which helps us to understand the agreement due to random chance rather than true observation. The observed and expected agreement quantify the reliability or agreement between the classifier

TABLE 10. CK scores of the proposed TPEvo-CNN model for each dataset.

Dataset	CK score
COVID-19x$_1$	0.96
COVID-19x$_2$	0.95
COVID-19-CT	0.90
COVID-19-Radiography	0.95
Pneumonia	0.53
Skin cancer	0.63

and the experimental dataset. The higher CK scores reveal the stronger agreement, while the lower indicates the less agreement for a classifier.

The results of the CK score for the proposed TPEvo-CNN model on six medical image datasets are presented in Table 10. It can be observed from the results that the levels of agreement of the proposed model vary on different datasets. For the COVID-19 datasets, the CK scores vary between 0.90 and 0.96, which is close to the 1, recommended as strong agreement. Hence, the proposed CNN model is consistent in classifying the COVID-19 datasets. Surprisingly, the CK scores are obtained as 0.53 and 0.63 for pneumonia and skin cancer datasets, respectively, through the TPEvo-CNN model. It demonstrates a moderate agreement between the proposed model with the corresponding datasets. This unfortunate situation can occur due to class imbalance and insufficient samples in the datasets used for training the model. However, the proposed model's overall performance is reasonable and satisfactory based on statistical analysis.

D. RADAR PLOTS

We also draw radar plots for the generated best CNN models in Figure 9 to more clearly illustrate the various classification metrics for each of the six datasets. For COVID-19x$_1$ dataset, both COVID-19 and the normal class produce similar accuracy 98.2%, while the COVID-19 class generates better results of precision 98.9% than normal class 97.7%. The normal class generates a higher value for recall and F1-score with 99.2% and 98.4% respectively, compared to 96.9% and 97.9% for COVID-19 class. In case of the COVID-19x$_2$ dataset, the pneumonia class achieves better results in all cases of classification metrics compared to the COVID-19 and normal classes. It generates 98.9% accuracy, 98.8% precision, 98.7% recall and 98.7% F1-score value compared to 98.2%, 97.3%, 98.6%, 97.9% for COVID-19 class and 98.9%, 98.8%, 98.7%, 98.7% for normal class. In COVID-19-CT, the COVID-19 class performs better than the normal class with respect to its recall and F1-score, which are 95.8%, 95.3%, respectively. The normal class generates 94.9% recall and 95.4% F1-score value. The precision value of the COVID-19 class is 94.9% while 95.9% for the normal class. In case of accuracy, both classes generate similar results of 95.4%. Again, in the COVID-19-Radiography dataset, among four classes, the normal class produced the best results compared to the other three classes. The normal class gives corresponding results of 98.3%, 98.3%, 98.1%, 98.2% as accuracy, precision, recall and F1-score, respectively. For the same dataset, the produced accuracy, precision, recall and F1-score are 98.5%, 95.4%, 95.6% and 95.5% respectively, for the COVID-19 class, whereas the values are 98.1%, 97.6%, 97.8%, 97.7% for the class lung_opacity and 98.5%, 88.1%, 88.8%, 88.4% for the viral pneumonia. The pneumonia dataset generates equal accuracy for the bacterial and viral classes as 79.4%. For the remaining precision, recall and F1-score, the bacterial type performs well, which are 82%, 82.7%, 84.3% respectively than 67.4%, 72.7%, 70.1% for viral class. Finally, in the case of the skin cancer dataset, the benign and malignant classes achieve equal accuracy of 83.4%. But for precision and F1-score, the class benign gives better results with 92.2% and 85.8% compared to the 72.8% and 80% for the malignant class. For recall, benign produces 80.2% and far enough from 88.7% in the malignant class.

E. DISCUSSION ON ACHIEVED TOP RANKED CNN MODEL ARCHITECTURES

The evolution of the best CNN models of six datasets are further demonstrated in Figure 10 with respect to their training and validation samples over 20 generations. The initial training accuracy for the COVID-19x$_1$ dataset is 94.76% in the first generation, and it steadily improved to 97.89% for the following 19 generations. The validation dataset showed a similar pattern, with an initial accuracy of 94.61% in the first generation, which continued to increase and reached 97.78% in the 20th generation. Again, the training started with 94.06% accuracy and ended with 96.88% for COVID-19x$_2$ dataset. On the other hand, for the same dataset, the optimal CNN model starts and finishes the validation accuracy with 93.89% and 96.48%, respectively, over an entire generation. The best CNN model of the COVID-19-CT dataset starts training with a 93.40% accuracy and stops with an accuracy of 95.19% at the last generation. For validation, the model produced an accuracy of 93.19% that continued to increase and gave 94.92% accuracy at the maximum generation. The best CNN model of the COVID-19-Radiography dataset starts training with a 95.22% accuracy and ends with a 96.92% accuracy at the last generation. For validation, the model provides an accuracy of 95.09%, which keeps increasing, and 96.88% in the last generation. Similarly, in case of the pneumonia dataset, there is a noticeable improvement in the accuracy of both the training and validation set with an increase in the number of generations. The training accuracy begins at 74.11% and finally reaches an accuracy of 77.09% at generation 20. Similarly, the validation accuracy starts at 73.91% and increases to 76.98% at generation 20. The accuracy of the training and validation sets for the skin cancer dataset increases steadily across generations. The best CNN architecture starts with a training accuracy of 79.95% in the first generation and improves to 83.67% in the last generation. Similarly, the validation accuracy begins at 79.81% and improves to 83.17% in the last generation. Hence, it is evident that the accuracy for both the training and

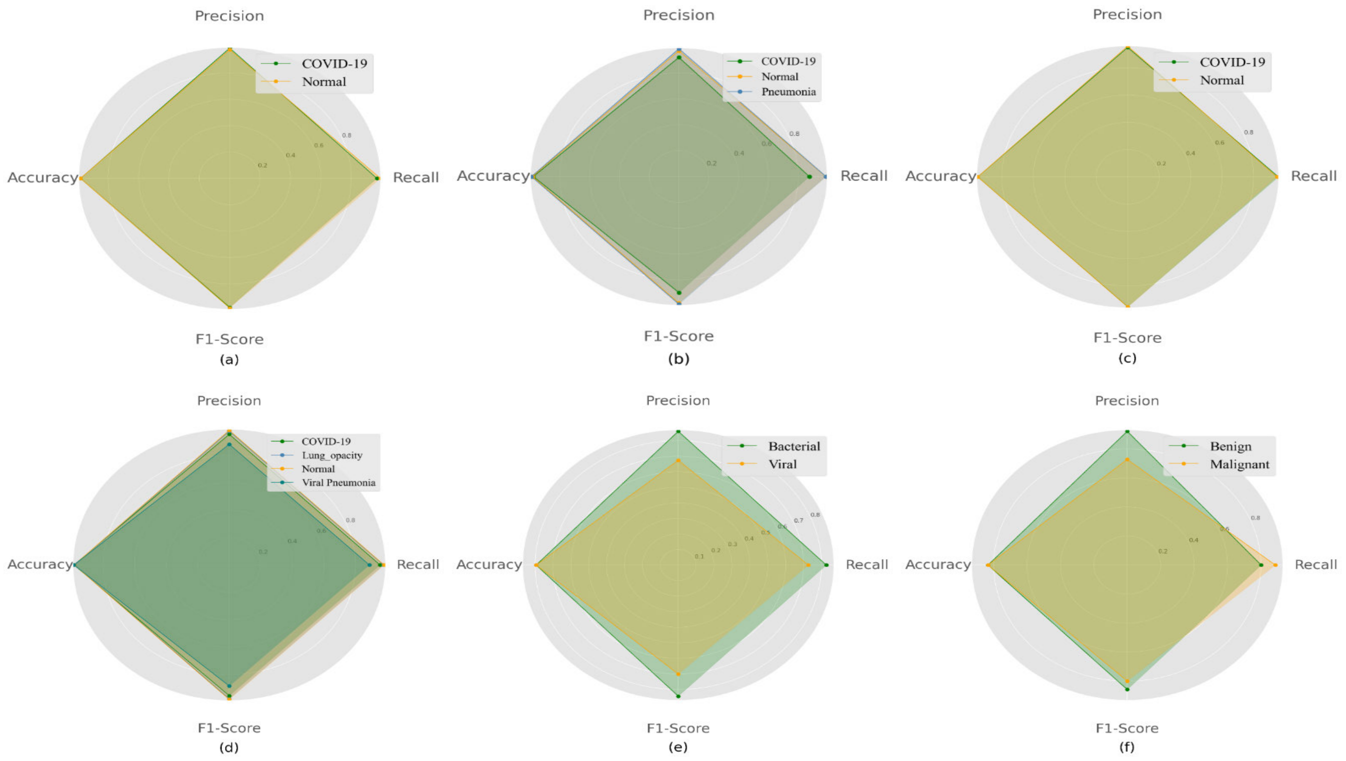

FIGURE 9. Class-wise accuracy, precision, recall, and F1-score for the (a) COVID-19x$_1$ (b) COVID-19x$_2$(c) COVID-19-CT (d) COVID-19-Radiography (e) Pneumonia (f) Skin Cancer dataset.

validation datasets are consistently improving as the number of generations increases.

Further, we investigated the hyper-parameter settings corresponding to each layer of the CNN model through the scatter plots for each dataset. We have considered the top ten CNN model architectures and their associated hyper-parameter settings generated by the proposed model. The scatter plots for each layer type, such as Convo, Pool, and FC of the CNN model concerning each dataset, are presented in Figure 11. In the case of COVID-19x$_1$ dataset, among the achieved top ten architectures, eight have the same number as five Convo layers, while two have only six. It can be observed in Figure 11(a) that the filter sizes and the number of filters for the Convo layer vary mostly from 3 to 7 and 120 to 250, respectively. On the other hand, eight architectures attained three Pool layers, and the remaining two CNNs attained four Pool layers. Their associated hyper-parameters, such as Pool kernel size, varies from 2 to 3, and Pool type is obtained as max Pool, shown in Figure 11(b). Finally, the top ten generated architectures attained the less number of FC layers (2 or 3) in which the number of neurons in the layer mostly varies from 160 to 198 as shown in Figure 11(c). For COVID-19x$_2$ dataset, it can be observed that eight of the top ten architectures include five Convo layers, while two have only six Convo layers along with the variation of 3 to 6 filter sizes and 110 to 250 number of filters which are presented in Figure 11(a). However, six CNN model architectures achieved three Pool layers, and the four CNNs achieved two Pool layers. Their corresponding hyper-parameters, such as Pool kernel size, which ranges from 2 to 3, and Pool type, which is mostly max Pool, are depicted in Figure 11(b). Finally, as shown in Figure 11(c), the six CNN model has two FC layer, and four have one FC layer, with mostly 112 to 275 neurons per layer. In the context of the COVID-19-CT dataset, it is evident that most of the top ten architectures (eight out of ten) consist of five Convo layers. However, two architectures contain six and seven Convo layers, respectively, along with a varying range of filter sizes (ranging from 3 to 4) and filter numbers (ranging from 139 to 242), as illustrated in Figure 11(a). Additionally, seven CNN models produced three Pool layers, while the remaining three employ two Pool layers. The hyper-parameters associated with the Pool layers, such as the kernel size (ranging from 2 to 3) and the predominant use of average pooling, are depicted in Figure 11(b). Finally, Figure 11(c) demonstrates that eight CNN models incorporate two FC layers, while the remaining two have three FC layers. The number of neurons per FC layer is predominantly within the 178 to 258 range. The COVID-19-Radiography dataset reveals an exciting pattern with most CNN architectures, comprising eight out of ten models, consisting of five Convo layers. However, two architectures stand out by having only six Convo layers. These models exhibit a wide range of filter sizes, mostly among 3, 5 and 6, and generated 111 to

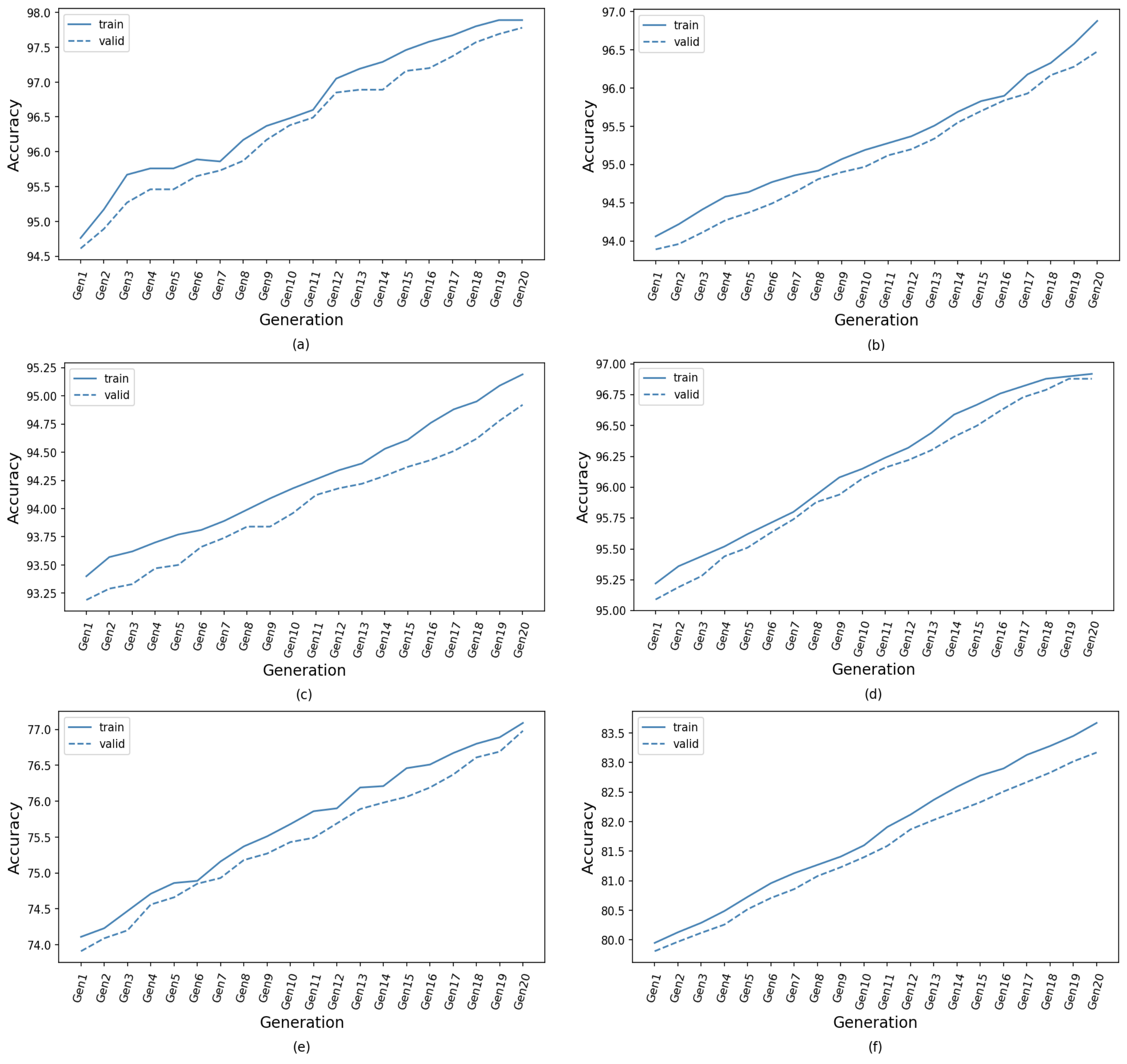

FIGURE 10. The evolution of the six best CNN networks in (a) COVID-19x$_1$, (b) COVID-19x$_2$, (c) COVID-19-CT, (d) COVID-19-Radiography, (e) Pneumonia and (f) Skin Cancer dataset.

244 filters, as represented in Figure 11(a). Moving on to the Pool layers, five CNN architectures incorporate three Pools, while the four models employ two Pool layers, and one model has one Pool layer. The hyper-parameters associated with these Pool layers, such as mostly 2 kernel size and max pooling, are depicted in Figure 11(b). In terms of FC layers, Figure 11(c) illustrates that eight CNN models have two FC layers, while the remaining two models employ three FC layers. The number of neurons in FC primarily ranges from 112 to 275. The top generated CNN architectures for the pneumonia dataset exhibit some consistent patterns with five CNN architectures consisting of eight Convo layers, three having seven Convo, and two composed of six Convo layers, as presented in Figure 11(a). The filter sizes are mostly 3, 5 and 6, with the number of filters varying from 92 to 244. In case of Pool layers, all ten CNN architectures incorporate three Pool layers along with mostly kernel size 3 and max Pool being the most common type, illustrated in Figure 11(b). Additionally, in Figure 11(c), eight CNN models are composed of two FC layers, and two have three FC layers in which the number of neurons ranges from 98 to 228. The top ten CNN architectures for the skin cancer dataset

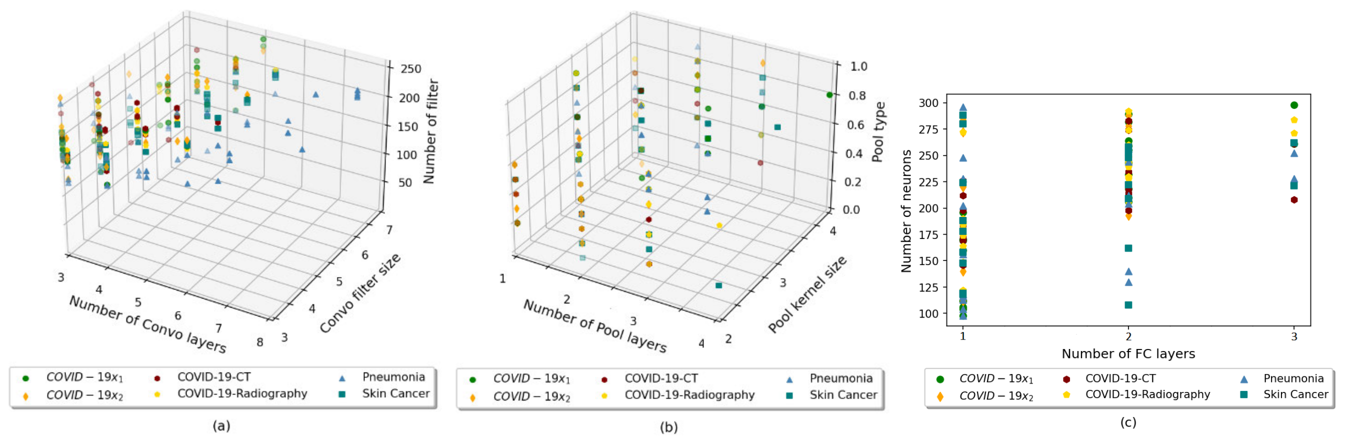

FIGURE 11. The hyper-parameters configuration for the top ten best CNN models are provided in three categories, namely, (a) Convolution, (b) Pooling, and (c) Fully-Connected layer for each of the six datasets.

TABLE 11. The obtained CNN architectures for each of the six medical image datasets in our experiment.

Dataset	CNN Architecture with Hyper-parameters
COVID-19x$_1$	Convo ($c_k = 3, c_n = 110$) + Convo ($c_k = 3, c_n = 132$) + Max Pool($p_k = 3$) + Convo ($c_k = 4, c_n = 221$) + Convo ($c_k = 6, c_n = 194$) + Max Pool ($p_k = 2$) + Convo ($c_k = 3, c_n = 248$) + Max Pool ($p_k = 3$) + FC ($f_n = 196$) + FC ($f_n = 282$) + FC ($f_n = 2$)
COVID-19x$_2$	Convo ($c_k = 3, c_n = 118$) + Convo ($c_k = 4, c_n = 158$) + Max Pool($p_k = 3$) + Convo ($c_k = 3, c_n = 166$) + Convo ($c_k = 6, c_n = 210$) + Max Pool ($p_k = 3$) + Convo ($c_k = 3, c_n = 244$) + Avg Pool ($p_k = 2$) + FC ($f_n = 220$) + FC ($f_n = 3$)
COVID-19-CT	Convo ($c_k = 3, c_n = 122$) + Convo ($c_k = 4, c_n = 181$) + Avg Pool ($p_k = 2$) + Convo ($c_k = 4, c_n = 198$) + Convo ($c_k = 3, c_n = 229$) + Avg Pool ($p_k = 2$) + Convo ($c_k = 3, c_n = 252$) + Max Pool ($p_k = 2$) + FC ($f_n = 212$) + FC ($f_n = 274$) + FC ($f_n = 2$)
COVID-19-Radiography	Convo ($c_k = 3, c_n = 98$) + Convo ($c_k = 3, c_n = 114$) + Max Pool ($p_k = 3$) + Convo ($c_k = 4, c_n = 156$) + Convo ($c_k = 6, c_n = 196$) + Avg Pool ($p_k = 2$) + Convo ($c_k = 6, c_n = 234$) + Max Pool ($p_k = 2$) + FC ($f_n = 184$) + FC ($f_n = 242$) + FC ($f_n = 4$)
Pneumonia	Convo ($c_k = 3, c_n = 98$) + Convo ($c_k = 3, c_n = 142$) + Avg Pool ($p_k = 3$) + Convo ($c_k = 5, c_n = 190$) + Convo ($c_k = 4, c_n = 111$) + Convo ($c_k = 5, c_n = 219$) + Convo ($c_k = 4, c_n = 142$) + Convo ($c_k = 5, c_n = 201$) + Max Pool ($p_k = 3$) + Convo ($c_k = 6, c_n = 251$) + Max Pool ($p_k = 3$) + FC ($f_n = 228$) + FC ($f_n = 2$)
Skin Cancer	Convo ($c_k = 3, c_n = 109$) + Convo ($c_k = 3, c_n = 122$) + Avg Pool ($p_k = 3$) + Convo ($c_k = 5, c_n = 139$) + Convo ($c_k = 4, c_n = 187$) + Convo ($c_k = 5, c_n = 192$) + Avg Pool ($p_k = 2$) + Convo ($c_k = 6, c_n = 241$) + Max Pool ($p_k = 4$) + FC ($f_n = 188$) + FC ($f_n = 209$) + FC ($f_n = 2$)

incorporate six Convo layers, as shown in Figure 11(a). The commonly used filter sizes are 3, 5, and 6, with the number of filters varying between 108 and 248. In addition, eight CNN models are incorporated with three Pool layers, while two architectures include four Pool layers. Most of the Pool layers have a kernel size of 3, with a preference for max pooling and presented in Figure 11(b). Furthermore, two architectures have three FC layers, five have two FC layers, and three have one FC layer. The number of neurons in these FC layers ranges from 148 to 224, as depicted in Figure 11(c).

Finally, the obtained suitable CNN model architectures using the proposed method corresponding to each of the six medical datasets are provided in Table 11. Most of the obtained suitable CNN models for COVID-19x$_1$, COVID-19x$_2$, COVID-19-CT, COVID-19-Radiography, pneumonia, and skin cancer datasets are composed of 10, 9, 9, 10, 12 and 11 layers, respectively. For the COVID-19x$_1$ dataset, only five, three and two layers for convolutional, max-pooling, and fully-connected layers are suitable for the classification. On the other hand, for the COVID-19x$_2$ dataset, five convolutional layers, two max-pooling layers, one average pooling layer, and one fully-connected layer are sufficient. The optimal CNN architecture for the COVID-19-CT dataset comprises of four convolutional layers, two average pooling layers, one max pooling layer, and two fully connected (FC) layers. Similarly, it can be observed that five convolutional layers, two max-pooling layers, one average pooling layer, and two FC layers of CNN architecture are enough to classify the COVID-19 radiography dataset. The obtained CNN architecture consists of eight convolutional layers, one average pooling layer, two max-pooling layers, and one FC layer to classify the pneumonia dataset. Finally, the top-performing CNN model architecture for the skin cancer dataset comprises six convolutional layers, two average pooling layers, one max pooling layer, and two FC layers.

VI. CONCLUSION

This research paper introduces a method involving a two-phase evolutionary framework to design the most effective CNN model architectures for various medical image classification tasks. In this framework, each individual represents a CNN architecture, which includes different layer types and hyper-parameter configurations of those layers. During Phase I of the proposed evolutionary framework, the number of different types of layers (Convo, Pool, and FC) in the CNN architecture is determined using the DE algorithm. On the other hand, the hyper-parameters of the generated CNN architectures are tuned using the enhanced crossover, mutation, and elitism selection strategy of the GA algorithm in Phase II. The proposed method is evaluated on datasets of COVID-19, pneumonia, and skin cancer. The experimental results demonstrate superior performance compared to state-of-the-art models, including existing COVID-19 models, pre-trained models, and NAS-based CNN models in terms of precision, recall, F1-score and classification accuracy. Furthermore, we also highlight the effectiveness of the proposed model itself by representing the confusion matrix, statistical analysis, and radar plots of six datasets and discussed the configuration of the best-generated CNN models.

In future, a block or cell-based model can be designed using the proposed method. Moreover, this proposed approach can be applied to medical signal processing-based disease predictions.

REFERENCES

[1] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Deep learning for hyperspectral image classification: An overview," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6690–6709, Sep. 2019.

[2] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Feb. 2020.

[3] D. Das, R. Naskar, and R. S. Chakraborty, "Image splicing detection with principal component analysis generated low-dimensional homogeneous feature set based on local binary pattern and support vector machine," *Multimedia Tools Appl.*, vol. 82, pp. 25847–25864, Mar. 2023.

[4] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, pp. 1–14, Apr. 2015.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[11] A. Makwe and A. S. Rathore, "An empirical study of neural network hyperparameters," in *Evolution in Computational Intelligence: Frontiers in Intelligent Computing: Theory and Applications (FICTA 2020)*, vol. 1. Singapore: Springer, 2021, pp. 371–383.

[12] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[13] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, May 2022.

[14] X. Zhou, A. K. Qin, Y. Sun, and K. C. Tan, "A survey of advances in evolutionary neural architecture search," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 950–957.

[15] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[16] M. Ahmad, M. Abdullah, and D. Han, "A novel encoding scheme for complex neural architecture search," in *Proc. 34th Int. Tech. Conf. Circuits/Systems, Comput. Commun. (ITC-CSCC)*, Jun. 2019, pp. 1–4.

[17] C. White, W. Neiswanger, S. Nolen, and Y. Savani, "A study on encodings for neural architecture search," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20309–20319.

[18] X. Zheng, R. Ji, Q. Wang, Q. Ye, Z. Li, Y. Tian, and Q. Tian, "Rethinking performance estimation in neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11353–11362.

[19] Y. Sun, X. Sun, Y. Fang, G. G. Yen, and Y. Liu, "A novel training protocol for performance predictors of evolutionary neural architecture search algorithms," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 524–536, Jun. 2021.

[20] H. Tan, R. Cheng, S. Huang, C. He, C. Qiu, F. Yang, and P. Luo, "RelativeNAS: Relative neural architecture search via slow-fast learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 475–489, Jan. 2023.

[21] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–18.

[22] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[23] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[24] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.

[25] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[26] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1388–1397.

[27] H. Zhu, Z. An, C. Yang, K. Xu, E. Zhao, and Y. Xu, "EENA: Efficient evolution of neural architecture," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1891–1899.

[28] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[29] M. Suganuma, M. Kobayashi, S. Shirakawa, and T. Nagao, "Evolution of deep convolutional neural networks using Cartesian genetic programming," *Evol. Comput.*, vol. 28, no. 1, pp. 141–163, Mar. 2020.

[30] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, Apr. 2020.

[31] B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid differential evolution approach to designing deep convolutional neural networks for image classification," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2018, pp. 237–250.

[32] A. Ghosh, N. D. Jana, S. Mallik, and Z. Zhao, "Designing optimal convolutional neural network architecture using differential evolution algorithm," *Patterns*, vol. 3, no. 9, Sep. 2022, Art. no. 100567.

[33] A. Ghosh and N. D. Jana, "Artificial bee colony optimization based optimal convolutional neural network architecture design," in *Proc. IEEE 19th India Council Int. Conf. (INDICON)*, Nov. 2022, pp. 1–7.

[34] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm Evol. Comput.*, vol. 49, pp. 62–74, Sep. 2019.

[35] E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," *Proc. U.K. Workshop Comput. Intell.*, 2019, pp. 119–130.

[36] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002.

[37] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proc. Workshop Mach. Learn. High-Perform. Comput. Environ.*, Nov. 2015, pp. 1–5.

[38] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[40] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[41] D. E. Goldberg, *Genetic Algorithms*. Noida, India: Pearson, 2013.

[42] M.-D. Yang, Y.-F. Yang, and Y.-P. Chen, "Evaluation of fitness functions of GA classification," in *Proc. Companion Publication Annu. Conf. Genet. Evol. Comput.*, Jul. 2014, pp. 1479–1480.

[43] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 4780–4789.

[44] L. Wang, Z. Q. Lin, and A. Wong, "COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images," *Sci. Rep.*, vol. 10, no. 1, pp. 1–12, Nov. 2020.

[45] I. D. Apostolopoulos and T. A. Mpesiana, "COVID-19: Automatic detection from X-ray images utilizing transfer learning with convolutional neural networks," *Phys. Eng. Sci. Med.*, vol. 43, no. 2, pp. 635–640, Jun. 2020.

[46] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. R. Acharya, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," *Comput. Biol. Med.*, vol. 121, Jun. 2020, Art. no. 103792.

[47] A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro, "CovidGAN: Data augmentation using auxiliary classifier GAN for improved COVID-19 detection," *IEEE Access*, vol. 8, pp. 91916–91923, 2020.

[48] A. I. Khan, J. L. Shah, and M. M. Bhat, "CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest X-ray images," *Comput. Methods Programs Biomed.*, vol. 196, Nov. 2020, Art. no. 105581.

[49] D. Ezzat, A. E. Hassanien, and H. A. Ella, "An optimized deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization," *Appl. Soft Comput.*, vol. 98, Jan. 2021, Art. no. 106742.

[50] Y. Song, S. Zheng, L. Li, X. Zhang, X. Zhang, Z. Huang, J. Chen, R. Wang, H. Zhao, Y. Chong, J. Shen, Y. Zha, and Y. Yang, "Deep learning enables accurate diagnosis of novel coronavirus (COVID-19) with CT images," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 18, no. 6, pp. 2775–2780, Nov. 2021.

[51] D. M. Ibrahim, N. M. Elshennawy, and A. M. Sarhan, "Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases," *Comput. Biol. Med.*, vol. 132, May 2021, Art. no. 104348.

[52] H. Louati, S. Bechikh, A. Louati, A. Aldaej, and L. B. Said, "Joint design and compression of convolutional neural networks as a bi-level optimization problem," *Neural Comput. Appl.*, vol. 34, pp. 15007–15029, Apr. 2022.

[53] A. Singh, K. K. Singh, M. Greguš, and I. Izonin, "CNGOD—An improved convolution neural network with grasshopper optimization for detection of COVID-19," *Math. Biosci. Eng.*, vol. 19, no. 12, pp. 12518–12531, 2022.

[54] Y. Zhou, X. Wang, M. Zhang, J. Zhu, R. Zheng, and Q. Wu, "MPCE: A maximum probability based cross entropy loss function for neural network classification," *IEEE Access*, vol. 7, pp. 146331–146341, 2019.

[55] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 291–322, Jan. 2022.

[56] C. Banerjee, T. Mukherjee, and E. Pasiliao, "An empirical study on generalizations of the ReLU activation function," in *Proc. ACM Southeast Conf.*, Apr. 2019, pp. 164–167.

[57] S. Y. Sen and N. Özkurt, "Convolutional neural network hyperparameter tuning with Adam optimizer for ECG classification," in *Proc. Innov. Intell. Syst. Appl. Conf. (ASYU)*, Oct. 2020, pp. 1–6.

[58] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, Mar. 2011.

[59] R. Lakshmi and K. Vivekanandan, "An analysis of recombination operator in genetic algorithms," in *Proc. 5th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2013, pp. 223–226.

[60] S. Akter, M. W. Murad, R. H. Chaity, M. Sadiquzzaman, and S. Akter, "Genetic algorithm with updated multipoint crossover technique and its application to TSP," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jun. 2020, pp. 1209–1212.

[61] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Found. Genet. Algorithms*, vol. 1, no. 1, pp. 69–93, 1991.

[62] W. El-Shafai and F. E. Abd El-Samie, "Extensive COVID-19 X-ray and CT chest images dataset," Mendeley Data, V3, 2020. Accessed: Dec. 29, 2020. [Online]. Available: https://data.mendeley.com/datasets/8h65ywd2jr/3

[63] E. Soares, P. Angelov, S. Biaso, M. H. Froes, and D. K. Abe, "Sars-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification," *MedRxiv*, p. 2020-04, 2020, doi: 10.1101/2020.04.24.20078584.

[64] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, M. B. I. Reaz, and M. T. Islam, "Can AI help in screening viral and COVID-19 pneumonia?" *IEEE Access*, vol. 8, pp. 132665–132676, 2020.

[65] U. Sait, K. V. G. Lal, S. P. Prajapati, R. Bhaumik, T. Kumar, S. Shivakumar, and K. Bhalla, "Curated dataset for COVID-19 posterior-anterior chest radiography images (X-rays)," Mendeley Data, V3, 2021, doi: 10.17632/9xkhgts2s6.3.

[66] C. Fanconi. (2021). Skin cancer: Malignant vs benign. Kaggle Repository. [Online]. Available: https://www.kaggle.com/datasets/fanconic/skin-cancer-malignant-vs-benign

[67] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, Apr. 2016.

[68] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[69] D. Guo, X. Wang, K. Gao, Y. Jin, J. Ding, and T. Chai, "Evolutionary optimization of high-dimensional multiobjective and many-objective expensive problems assisted by a dropout neural network," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 4, pp. 2084–2097, Apr. 2022.

[70] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.*, vol. 45, no. 4, pp. 427–437, Jul. 2009.

[71] R. Artstein and M. Poesio, "Inter-coder agreement for computational linguistics," *Comput. Linguistics*, vol. 34, no. 4, pp. 555–596, Dec. 2008.

ARJUN GHOSH (Student Member, IEEE) received the B.Sc. degree (Hons.) in computer science (CS) from the Malda College, University of North Bengal, Siliguri, India, the M.Sc. degree in CS from the University of North Bengal, and the M.Tech. degree in computer science and engineering (CSE) from the Maulana Abul Kalam Azad University of Technology (formerly known as the West Bengal University of Technology), Kolkata, India. He is currently pursuing the Ph.D. degree with the Department of CSE, National Institute of Technology Durgapur, Durgapur, India. His research interests include deep learning, neural architecture search, computer vision, and evolutionary computing.

NANDA DULAL JANA (Member, IEEE) received the B.Tech. and M.Tech. degrees in computer science and engineering from the University of Calcutta, Kolkata, India, in 2004 and 2006, respectively, and the Ph.D. degree in computer science and engineering from the Indian Institute of Engineering Science and Technology, Shibpur, India, in 2017. Currently, he is an Assistant Professor with the Department of Computer Science and Engineering, National Institute of Technology Durgapur, India. He has authored over 50 research articles in peer-reviewed journals and international conferences, focusing on machine learning, neural architecture search, and evolutionary computing.

SWAGATAM DAS (Senior Member, IEEE) received the B.E.Tel.E., M.E.Tel.E., and Ph.D. degrees in electronics and telecommunication engineering from Jadavpur University, Kolkata, India, in 2003, 2005, and 2009, respectively. Currently, he is the Head of the Electronics and Communication Sciences Unit and an Associate Professor with the Indian Statistical Institute, Kolkata. He has authored over 300 research articles in peer-reviewed journals and international conferences, amassing more than 30,000 Google Scholar citations with an H-index of 81. His research interests include evolutionary computing and machine learning. Moreover, he actively participates in program and organizing committees for prestigious international conferences, including the Conference on Neural Information Processing Systems, AAAI Conference on Artificial Intelligence, International Conference on Artificial Intelligence and Statistics, ACM Multimedia, British Machine Vision Conference, IEEE Congress on Evolutionary Computation, and Genetic and Evolutionary Computation Conference. He is also the founding Co-Editor-in-Chief of *Swarm and Evolutionary Computation* and serves as an Associate Editor for several IEEE journals, including IEEE Transactions on Cybernetics, IEEE Transactions on Evolutionary Computation, *Pattern Recognition*, *Neurocomputing*, and *Information Sciences*. Furthermore, he is an Editorial Board Member of *Information Fusion*, *Progress in Artificial Intelligence*, *Applied Soft Computing*, *Engineering Applications of Artificial Intelligence*, and *Artificial Intelligence Review*. He received the 2012 Young Engineer Award from the Indian National Academy of Engineering and the 2015 Thomson Reuters Research Excellence India Citation Award for being India's most cited researcher in Engineering and Computer Science, from 2010 to 2014.

RAMMOHAN MALLIPEDDI (Senior Member, IEEE) received the master's and Ph.D. degrees in computer control and automation from Nanyang Technological University, Singapore, in 2007 and 2010, respectively. Currently, he is an Associate Professor with the Department of Artificial Intelligence, Kyungpook National University, Daegu, South Korea. He has coauthored articles in IEEE Transactions on Evolutionary Computation. His research interests include evolutionary computing, artificial intelligence, image processing, digital signal processing, robotics, and control engineering. Additionally, he serves as an Associate Editor for *Swarm and Evolutionary Computation*, an international journal by Elsevier, and regularly reviews articles for journals, including IEEE Transactions on Evolutionary Computation and IEEE Transactions on Cybernetics.