Journal of Ambient Intelligence and Humanized Computing (2024) 15:451–464
https://doi.org/10.1007/s12652-022-03901-1

ORIGINAL RESEARCH

Evolving deep convolutional neural networks by IP-based marine predator algorithm for COVID-19 diagnosis using chest CT scans

Bing Liu[1] · Xuan Nie[1] · Zhongxian Li[1] · Shihong Yang[2] · Yushu Tian[3]

Received: 8 December 2021 / Accepted: 4 May 2022 / Published online: 24 May 2022

Abstract
This paper proposes an optimal structured deep convolutional neural network (DCNN) based on the marine predator algorithm (MPA) to construct a novel automatic diagnosis platform that may help radiologists identify COVID-19 and non-COVID-19 patients based on CT scan categorization and analysis. The goal is met with the help of three modifications based on the regular MPA. First, a novel encoding scheme based on Internet Protocol (IP) addresses is proposed, followed by introducing an Enfeebled layer to build a variable-length DCNN. Finally, the learning process divides big datasets into smaller chunks that are randomly evaluated. The proposed model is compared to the COVID-CT and SARS-CoV-2 datasets to undertake a complete evaluation. Following that, the performance of the developed model (DCNN-IPMPA) is compared to that of a typical DCNN and seven variable-length models using five well-known comparison metrics, as well as the receiver operating characteristic and precision-recall curves. The results show that the DCNN-IPMPA outperforms other benchmarks, with a final accuracy of 97.21% on the SARS-CoV-2 dataset and 97.94% on the COVID-CT dataset. Also, timing analysis indicates that the DCNN processing time is the best among all benchmarks as expected; however, DCNN-IPMPA represents a competitive result compared to the standard DCNN.

Keywords DCNNs · Chest CT scans · COVID-19 · Internet protocol address · Marine predator algorithm

1 Introduction

Due to the widespread of COVID-19 and its variations in all nations, the timely and exact diagnosis of COVID-19 is essential. The initial stage in the COVID-19 diagnosis is to take the reverse transcription-polymerase chain reaction (RT-PCR) (Hu et al., 2021a); however, in a few situations, the RT-PCR test of a subject is negative, yet the associated patient obtains difficulty in breathing (Wu et al., 2021a, b). The chest CT scan is advised as a second-line diagnostic in this situation. If the lungs are heavily infected with COVID-19, its diagnosis by thorough observation can be easily made. However, when the disease is in its early stages, a visual diagnosis of this condition is accompanied by some doubt. COVID-19 lung damage has recently been identified using X-rays and CT scans (Khishe et al., 2021). However, the diagnostic accuracy is dependent on the expert's assessment (Zhang et al., 2022). Since changes in pixels' value may be better identified by image processing techniques as a quantitive method, this research aims to automatically and more accurately identify the contaminated CT images (Cao et al., 2022; Liu et al., 2022a, b). Deep learning (DL) algorithms can automatically identify this condition to address the mentioned problem (Saffari et al., 2020).

CT Scans reveal the bones, blood vessels, and other internal organs. As a result, they enable clinicians to assess the size and structure of internal organs. Compared to X-rays, CT Scans may isolate a specific body location without damaging nearby areas, and they can also reveal detailed images of the patient's body (Shah et al., 2021). This data can be evaluated to determine the exact location of the problem. As a result, deep learning techniques have made extensive use of the potential of chest CT scans for early detection and diagnosis of COVID-19 (Afshar et al., 2021; Dai et al., 2020; Shah et al., 2021). Grewal et al., for example, used

✉ Yushu Tian
yushutian389@outlook.com

[1] School of Software, Northwestern Polytechnical University, Xi'an, Shaanxi Province, China

[2] School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an, Shaanxi Province, China

[3] Guiyang Fourth People's Hospital, Guiyang City, Guizhou province, China

DenseNet and RNNs to interpret brain CT images (Grewal et al., 2018). Song et al. developed three deep neural models for lung cancer detection (SAE, DNN, and CNN) (Song et al., 2017). They discovered that the DCNN architecture surpasses the other models regarding accuracy. Gonzales et al. employed deep learning to investigate the neural system's ability to identify lung diseases and predict acute respiratory problems (González et al., 2018).

During the pandemic, it was discovered that CT Scan-based analysis methods accurately diagnosed COVID-19 disease. Zhao et al., for example, examined the association between chest CT scan data and pneumonia in COVID-19 patients (Bernheim et al., 2020). Due to the scarcity of acceptable COVID-19 CT Scan datasets, Zhao et al. constructed a freely accessible dataset consisting of 363 healthy persons and 349 COVID-19 occurrences from 216 patients (Jiang et al., 2019). Gozes et al. investigated the detection and tracking of COVID-19-positive cases. The accuracy of this improved model has been evaluated to be 95% (Gozes et al., 2020). Zheng et al., on the other hand, proposed a deep learning system for COVID-19 detection in 3D CT images (Zheng et al., 2020). Ai et al. demonstrated 97 percent sensitivity when using a pre-trained UNet to predict infection in 630 patients' CT scans (Ai et al., 2020).

COVID-CT was perhaps the first freely released data-set of this type. Following the presentation, this data set was used by a variety of researchers (Breban et al., 2013; Liu et al., 2020; Ozturk et al., 2020). Previously, the best F1-score, accuracy, and AUC were 86.5%, 85%, and 94% (Breban et al., 2013). Soares et al. (Angelov and Almeida Soares, 2020) presented a dataset of 2482 SARS-CoV-2 CT scans. Nonetheless, the aforementioned techniques need considerable processing time (He et al., 2020). As a result, a precise and rapid COVID-19 detector is necessary. Following a comprehensive investigation of the methods recently published for identifying and diagnosing COVID-19, it is clear that the DCNN is one of the most widely used techniques (Liu et al., 2017); thus, this research recommends utilizing DCNN's extraordinary potential as a detector of COVID-19 scans.

DCNN design complexity has been reduced by metaheuristics (Khishe and Mosavi, 2020a, b; Mosavi et al., 2017; Qiao et al., 2021), which develop an architecture without the assistance of a human designer (Rastogi and Choudhary, 2019). Numerous metaheuristics have been effectively studied and applied, including genetic programming (GP) (Suganuma et al., 2017), particle swarm optimization (PSO) (Mosavi and Khishe, 2017; Wang et al., 2018; J. Wu et al., 2021a, b), Position-transitional particle swarm optimization (Luo et al., 2020), sine–cosine algorithm (SCA) (Wang et al., 2020; C. Wu et al., 2021a, b), chimp optimization algorithm (ChOA) (Hu et al., 2021b), salp swarm algorithm (Khishe and Mohammadi, 2019), and genetic algorithms (GAs) (Liu et al., 2022a, b). However, learning from large data sets is impracticable due to the high cost of computing and the time of the learning process (Yuan et al., 2020). Perhaps, the EvoCNN was the first model that employed metaheuristic algorithms to evolve DCNN structures (Stanley and Miikkulainen, 2002). The EvoCNN helps save time by learning the classifier ten epochs rather than 25,600 epochs like the LEIC does.

Regardless of the advantages of different metaheuristic algorithms, the No-Free-Lunch (NFL) theorem (Webb et al., 2011) states that no metaheuristic technique can resolve all optimization problems satisfactorily. To address various optimization problems, scientists have attempted to develop new metaheuristics or improve existing ones. To overcome the difficulty of constructing DCNN structures without human intervention, we employ powerful nature-inspired techniques called Marine Predator Algorithms (MPA) (Mirjalili, 2016). A novel flexible encoding strategy is proposed in order to circumvent the limitations of fixed-length encodings.

As a result, the fundamental objective of the proposed model is to develop and implement a complementary MPA capable of discovering optimal DCNN structures automatically. To summarize, this article makes the following contributions:

1. Develop a novel variable-length MPA method that effectively encodes a DCNN architecture using a unique candid solution encoding technique.
2. Develop a method for learning variable-length DCNN structures independent of the fixed-length encoding constraint imposed by existing MPA; a new layer dubbed Enfeebled will be presented to construct a variable-length predator.
3. Develop a technique for evaluating fitness that uses a portion of the dataset rather than the complete dataset to accelerate the development process.

The following is the organization of the paper. Section 2 summarizes the DCNN, MPA algorithms, and the CT Scan datasets. Section 3 discusses the proposed approach for variable-length encoding. Section 4 summarizes and discusses the simulation results. Section 5 is the concluding section.

2 Background and materials

This section describes the background knowledge of the LeNet-5 DCNN, MPA, and CT Scan datasets.

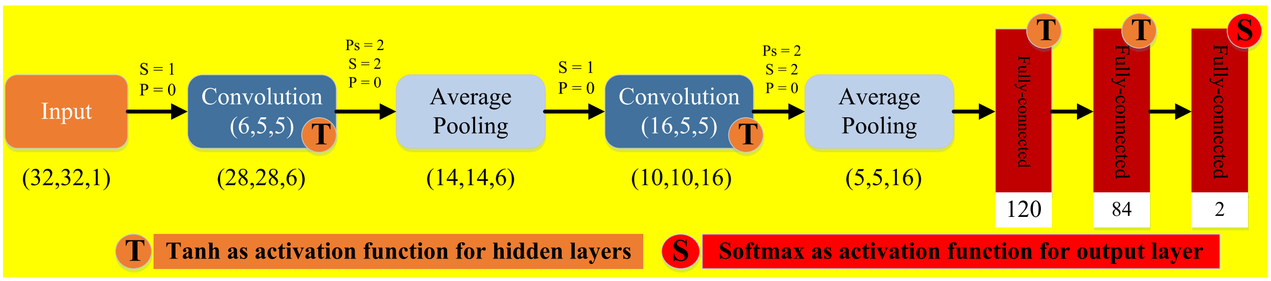


Fig. 1 The DCNN architecture

2.1 Deep convolutional neural network

DCNN is a straightforward yet successful model among the emerging deep learning-based techniques (Le Cun, 2015). We chose this network since its structure is straightforward and needs minimal parameters. As illustrated in Fig. 1, the DCNN is composed of three major components: convolution, subsampling, and fully connected layers. Additionally, feature maps, i.e.,FM_{ij}^{k} and sub-sampling operation, are respectively formulated by Eqs. (1) and (2).

$$FM_{ij}^{k} = \tanh((W^{k} \times x)_{ij} + b_{k}) \tag{1}$$

$$\alpha_{j} = \tanh(\beta \sum_{N\times N} \alpha_{i}^{n\times n} + b) \tag{2}$$

where b and β are respectively bias values and learning constants, and also $\alpha_{i}^{n\times n}$ are inputs (Liu et al., 2022a, b). The last fully-connected layer then performs classification. This layer has one neuron since our problem is a binary task (COVID and non-COVID).

2.2 Marine predator algorithm

In line with most metaheuristic algorithms, MPA is categorized as a population-based algorithm. Thus, the initial solution is regularly disseminated across the searching space as initial testing:

$$X_{0} = X_{\min} + rand(X_{\max} - X_{\min}) \tag{3}$$

where $X_{\min}$ and $X_{\max}$ signify the lower and upper bound of variables, respectively, and *rand* represents a uniformly distributed random vector that fluctuates from 0 to 1. It is worth stating that the best solution obtained so far in MPA is appointed as a leading predator for constructing a matrix named Elite, as expressed in Eq. (3). It is also essential to indicate that based on the information of the prey position, these matrix arrays will manage searching and finding the Prey.

$$\mathbf{Elite} = \begin{bmatrix} X_{1,1}^{I} & X_{1,2}^{I} & \cdots & X_{1,d}^{I} \\ X_{2,1}^{I} & X_{2,2}^{I} & \cdots & X_{2,d}^{I} \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1}^{I} & X_{n,2}^{I} & \cdots & X_{n,d}^{I} \end{bmatrix}_{n\times d} \tag{4}$$

Here, X^{I} signifies the best searching agent vector reproduced n times for constructing the **Elite** matrix. In which n represents the search agents count, d represents dimension numbers. The **Elite** is updated if the best searching agent is replaced with the better agent. **Prey** is another matrix that has a similar dimension as **Elite**. **Prey** is a matrix in which the searching agents update their position. This means that the initial **Prey** in which the best searching agent creates the Elite is produced by the initialization. The **Prey** is expressed in Eq. (5).

$$\mathbf{Prey} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}_{n\times d} \tag{5}$$

$X\ i,j$ In Eq. (5), signifies the *j*th dimension of *i*th searching agents. It is worth noting that the entire optimization procedure gets primarily and precisely associated with **Elite** and **Prey** matrices.

2.2.1 The MPA's optimization outline

The MPA's optimization procedure consists of three stages taking into consideration the various ratio of velocity, and together with simulating the complete life of both predator and Prey, these stages are (I) High-Velocity Ratio (HVR) or once a predator gets moving slower than Prey, (II) Unit-Velocity Ratio (UVR) or once both Prey and predator would have practically the identical speed, and (III) Low-Velocity Ratio (LVR) once the Prey gets moving slower than predator. The above three stages could further be described as follows:

- **Stage 1:** HVR or once the Prey gets moving slower than the predator. This setup takes place in the early iterations of optimization, where the exploration matters. The mathematical model of this rule is expressed as below:

$$\begin{aligned}&\textit{While}\ \text{Iteration} < \frac{1}{3}\times \text{maximum (Iteration)}\\ &\mathbf{stepsize_i} = \mathbf{R_{BM}} \otimes (\mathbf{Elite_i} - \mathbf{R_{BM}} \otimes \mathbf{Prey_i})\ \text{i} = 1, \ldots, \text{n}\\ &\mathbf{Prey_i} = \mathbf{Prey_i} + P.\mathbf{R} \otimes \mathbf{stepsize_i}\end{aligned} \tag{6}$$

Here $\mathbf{R_{BM}}$ represents a normally distributed random vector signifying the BM. The symbol ⊗ indicates entry-wise multiplications. In which the multiplication of $\mathbf{R_{BM}}$ by Prey mimics the prey motion. P signifies a constant number and its value set to 0.5, and $\mathbf{R}$ is a vector of uniform random numbers ranging from 0 to 1.

- **Stage 2**: In this stage, the Prey and predator move very similarly in pace. Here both exploitation and exploration are very crucial. Accordingly, 50% of the searching agents are allocated for exploration, and the remaining 50% are given for exploitation. The predator in this stage is accountable for exploration and the Prey for exploitation. Depending on the rule, in the UVR (v ≈1), the safest policy for a predator is BM if the Prey goes in LF. Accordingly, this research deems predator moves in BM while Prey moves in LF, as Eq. 7.

$$\begin{aligned}&\text{while}\ \frac{1}{3}\mathit{Maximum}(\mathit{Iteration}) < \text{Iteration} < \frac{2}{3}\mathit{Maximum}(\mathit{Iteration})\\ &\text{For: the first}\ \frac{1}{2}\times \text{searching agents}\\ &\mathbf{stepsize_i} = \mathbf{R_{LF}} \otimes (\mathbf{Elite_i} - \mathbf{R_{LF}} \otimes \mathbf{Prey_i})\ \text{i} = 1, \ldots, \text{n/2}\\ &\mathbf{Prey_i} = \mathbf{Prey_i} + P.\mathbf{R} \otimes \mathbf{stepsize_i}\end{aligned} \tag{7}$$

Here $\mathbf{R_{LF}}$ represents a random number vector depending on the LF. $\mathbf{R_{LF}}$ and **Prey** multiplication mimic the prey movement in LF, whereas adding the prey position to the step size imitates the prey movement. The MPA supposes the following expressions for the other half of the populations:

$$\begin{aligned}&\mathbf{stepsize_i} = \mathbf{R_{BM}} \otimes (\mathbf{R_{BM}} \otimes \mathbf{Elite_i} - \mathbf{Prey_i})\ \text{i} = \text{n/2}, \ldots, \text{n}\\ &\mathbf{Prey_i} = \mathbf{Elite_i} + P.\text{CF} \otimes \mathbf{stepsize_i}\\ &\text{CF} = (1 - \frac{\mathit{Iteration}}{\mathit{Maximum}(\mathit{Iteration})})^{2\times\frac{\mathit{Iteration}}{\mathit{Maximum}(\mathit{Iteration})}}\end{aligned} \tag{8}$$

Here CF represents a parameter that can be adaptive to manage the predator movements step size. $\mathbf{R_{BM}}$ and **Elite** multiplication mimic predator movement in BF, whereas prey modifies its position depending on the predator's movement in BM.

- **Stage 3:** This is where the Prey is moving slower than the predator, which can be introduced as follows:

$$\begin{aligned}&\text{While Iteration} > \frac{2}{3}\ \text{Maximum (Iteration)}\\ &\mathbf{stepsize_i} = \mathbf{R_{LF}} \otimes (\mathbf{R_{LF}} \otimes \mathbf{Elite_i} - \mathbf{Prey_i})\ \text{i} = 1, \ldots, \text{n}\\ &\mathbf{Prey_i} = \mathbf{Elite_i} + P.\text{CF} \otimes \mathbf{stepsize_i}\\ &\text{CF} = (1 - \frac{\mathit{Iteration}}{\mathit{Maximum}(\mathit{Iteration})})^{2\times\frac{\mathit{Iteration}}{\mathit{Maximum}(\mathit{Iteration})}}\end{aligned} \tag{9}$$

2.2.2 *The effect of fish aggregating devices and Eddy formation*

One more thing that triggers a change of behavior in marine predators is environmental or ecological problems, for instance, the Eddy formation or Fish Aggregating Devices (FADs) effects (Filmalter et al., 2011), which is deemed as local optima avoidance operator. Stagnation in local optima can be avoided by considering these longer jumps during the simulation. Therefore, the FADs impact is precisely given as:

$$\mathbf{Prey_i} = \begin{cases}\mathbf{Prey_i} + \text{CF} \times [\mathbf{X_{min}} + \mathbf{R} \otimes (\mathbf{X}_{max} - \mathbf{X_{min}})] \otimes \mathbf{U}\ \text{if r} \le \mathit{FADs}\\ \mathbf{Prey_i} + [\mathit{FADs}(1-r) + r](\mathbf{Prey}_{r1} - \mathbf{Prey}_{r2})\ \text{if r} > \mathit{FADs}\end{cases} \tag{10}$$

Here, the value of $\mathit{FADs} = 0.2$, which is considered the FADs probability impact on the process of optimization. $\mathbf{U}$ signifies the binary vector in conjunction with arrays as well as 0 and 1. This is created by producing a random vector ranging from 0 to 1 and modifying its array to 1 if it is greater than 0.2 and 0 if it is smaller than 0.2. r represents the uniform random number, which ranges from 0 to 1. Both $\mathbf{X_{min}}$ and $\mathbf{X_{max}}$ are considered vectors containing the lower and upper bound of the dimensions. r_1 and r_2 signify random indexes of the **Prey** matrix.

2.2.3 Memory of marine

Depending on the emphasized regions, marine predators hold an excellent memory in recapping where they have had successful hunting. The ability is mimicked through memory preservation within the MPA. The **Elite** matrix will be updated by modifying the **Prey** and executing FADs impact. The suitability of every searching agent of the existing iteration is assessed to its corresponding in an earlier iteration, and the existing one supersedes the searching agent if it is a better one.

2.3 Datasets

COVID-CT (Yang et al., 2020) and SARS-CoV-2 [20] are two CT Scan datasets used in this study, whose details are discussed in the following subsections.

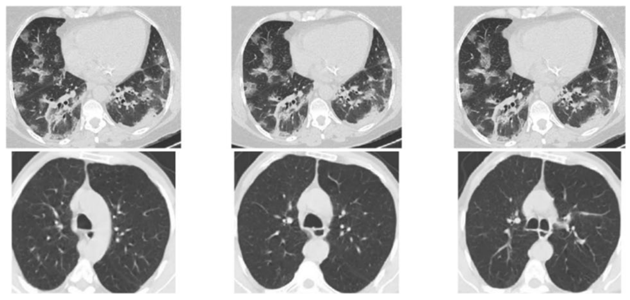


Fig. 2 Typical CT images from the SARS-CoV-2 CT-Scan dataset

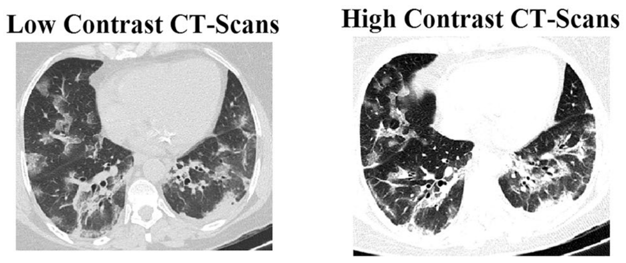


Fig. 3 The representation of two images with different sizes and contrast

2.3.1 SARS-CoV-2 CT-Scan dataset

SARS-CoV-2 was established at hospitals in Sao Paulo. This dataset contains 2482 CT scans of 120 patients, 1252 of whom are SARS-CoV-2 patients, and 1230 of whom are not infected but have a variety of respiratory ailments. Typical, COVID-19 positive and normal samples are depicted in Fig. 2. The first raw contains some COVID-19-infected instances, whereas the second raw contains regular (non-infected) instances. It should be noted that the SARS-CoV-2 has no extraordinary uniformity in terms of contrast and image sizes, as shown in Fig. 3.

2.3.2 COVID-CT dataset

The COVID-CT dataset[1] (Yang et al., 2020) contains CT scans of people infected with COVID-19. This collection includes 349 photos of 216 patients. Two more datasets, LUNA and MedPix, were employed to provide images of non-COVID and healthy individuals. This dataset segment contains 463 photographs of 55 healthy individuals. Similar to the first dataset, the second contains no specified image

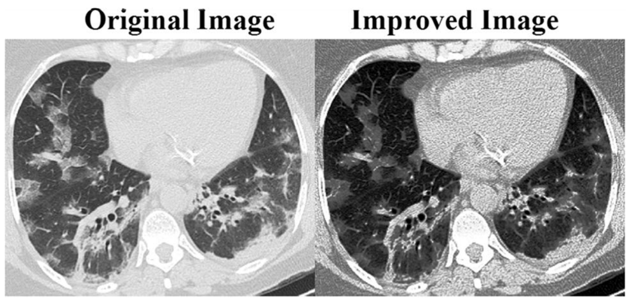


Fig. 4 An illustration of the standard image (left image) and enhanced image (right image)

Table 1 The datasets' specification

Dataset	Number	SARS-CoV-2 CT-Scan [20]	COVID-CT [22]
COVID-19	Patients	60	216
	Images	1252	349
Non-COVID-19	Patients	60	55
	Images	1230	463

contrast and size adjustment. As a result, as described in Perumal and Velmurugan (2018), we begin by enhancing the quality and size of the low-contrast images. Fig. Figure 4 illustrates both the unaltered and improved versions. The datasets utilized in this study are listed in Table 1.

3 Internet protocol marine predator algorithm

In order to evolve a DCNN, this section describes the IP-based MPA (IPMPA) technique in detail. The IPMPA algorithm is structured in Algorithm 1. The population will be initialized in three steps: encoding candid solutions, updating positions, and checking for termination.

```
Algorithm 1 Framework of IPMPA
Initialize the predators with a candid solution encoding strategy
            $X_i^t \leftarrow$ empty
while termination criterion is not satisfied
        update predators' position of by Eq. 3;
        evaluate the predators' fitness;
        update $X_i^t$;
end while
```

[1] https://github.com/UCSD-AI4H/COVID-CT.

Table 2 The information of the DCCN's layers with an example

Layer type	Parameter	Range	No. bits	Example
Convolutional	Filter size	[1, 8]	3	3 (011)
	No. feature maps	[1, 128]	7	5 (000 0101)
	Stride size	[1, 4]	2	3 (11)
	Summary		12	011 000 0101 11
Pooling	Kernel size	[1, 4]	2	3 (11)
	Stride size	[1, 4]	2	3 (11)
	Type:1(max),2(ave)	[1, 2]	1	1 (1)
	Place holder	[1, 128]	6	8 (00 1000)
	Summary		11	11 11 1 00 1000
Fully-connected	No. Neurons	[1, 2048]	11	128 (00,010,000,000)
	Summary		11	00,010,000,000
Enfeebled	Place holder	[1, 2048]	11	128 (00,010,000,000)
	Summary		11	00,010,000,000

3.1 Predator encoding strategy

The IP address structure of the network affects the IPMPA encoding strategy. Three layers comprise the DCNN structure, each with its own parameters. The following table summarizes the various layers. We can establish a network IP address with a fixed length and split it to allow different DCNN layers.

The IPMPA uses the IP address as the basis for its encoding technique. Three different types of layers comprise the DCNN architecture: convolutional, pooling, and fully connected. As illustrated in Table 2, each layer type has its own unique set of values and ranges for parameter settings.

A length must first be calculated for an IP-based encoding system. Three key parameters for convolutional layers, namely the number of feature maps, the filter size, and the stride size, are listed in the Parameter column of Table 2. Second, parameter ranges are adjusted to [1,8], [1,128], and [1,4], respectively, based on the sizes of the datasets. The decimal values 3, 5, and 3 can be converted to digitized strings 011, 000 0101, and 11, respectively, where the digitized string is finished with zeros until it reaches the appropriate bit length, as illustrated in the example value column of Table 2. Eventually, the summary row of the convolutional layer in Table 2 presents the total bit count of 12 and the sample digitized string 011 000 0101 01 obtained by connecting the strings of the three previously discussed parameters. The total number of bits and the sample binary string can be obtained using the same technique as with convolutional layers from pooling and fully-connected layers, as shown in Table 2. Because a layer's maximum bit count is twelve and an IP address is eight bits (one byte), a twelve-bit IP address will require 2 bytes.

As specified in Table 3, all DCNN layers must have CIDR-compliant subnets. Since DCNN layers are classified into three types, each type requires three subnets to be completely supported. The length of the IP address 0.0 plus the bit numbers of the convolution layer constitute the subnet mask (12 bits). 0.0/4 (0.0–15.255) is the subnet mask. The initial IP address for the pooling layer is 16.0, calculated by adding one to the IP address for the final convolution layer. Like the convolution layer, the subnet mask has a length of five, resulting in 16.0/5. As a result, subnets 16.0–23.255 correspond to the pooling layer. A similar issue arises with the fully-connected layer subnet 24.0/5, which spans the range 24.0–31.255. The subnets' arrangement is depicted in Table 3.

To adjust DCNN topologies with varying lengths, several layers in the IP-coded candid solution vector are disabled during initialization. To compensate, the Enfeebled layer and subnet are introduced. When all three DCNN layers are used, the Enfeebled subnet has an IP address range of 32.0/5, as demonstrated in Table 3. A binary string representing a particular layer is padded with zeros until it reaches the length of 2 bytes, at which point the subnet mask is implemented, and the string is converted to an IP address by splitting it by full stops as shown in Table 4. As a result, the 2-byte digitized strings 0000 0011 and 100 001 can be found in

Table 3 Four utilized subnets for four types of DCNN's layers

Layer type	Convolution	Fully-connected	Pooling	Enfeebled
Subnet (CIDR)	0.0/4	16.0/5	24.0/5	32.0/5
IP range	0.0–15.255	16.0–23.255	24.0–31.255	32.0–39.255

Table 4 IP addresses' examples

Layer Type	Convolution	Fully-connected	Pooling	Enfeebled
Binary	(0000) 011 000 1000 01	(00,000)011 11,111,001	(00,000)01 01 0 00 1011	(00,000)01,111,111,001
IP address	6.33	27.249	18.139	35.249

Table 5 The list of parameters

Parameters	Parameter definition	Value
Max-length	Max. length of DCNN layers	13
N	Population size	40
Max-totally-connected	Max. fully-connected layers (at least one fully-connected layer is required)	3
k	The training epoch number before evaluating the trained DCNN	10
No. batch	The batch size for evaluating the DCNN	200
Optimizer	The type of optimizer	Adam

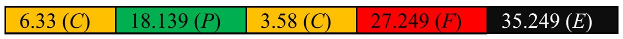


Fig. 5 An IP address for five DCNN layers

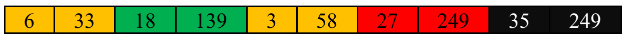


Fig. 6 The illustration of an encoded candid solution vector

Table 2. In order to obtain the IP address 6.33, the first byte was converted to 6 and the second to 33.

Before computing search agents, each layer must be converted to a two-byte IP address. First, some properties such as maximum length and maximum completely linked length must be stated. These are detailed in Table 5. The array that stores position data will be twice the size of a single byte, with each byte representing one dimension of the candid solution.

Consider the following candid solution vector illustration to better understand DCNN encoding and its variable-length architecture. In the DCNN architecture depicted in Fig. 5, the Pooling (P), Convolution (C), Fully Connected (F), and Enfeebled (E) layers can all be expressed by IP addresses. The size of the relevant candid solution vector is depicted in Fig. 6.

Following many MPA modifications, the 9th and 10th dimensions of the candid candidate solution may change to 18 and 139, respectively, converting the 3rd IP address indicating an Enfeebled layer to a Pooling layer and so changing the candid solution's DCNN architecture to five layers. In summary, as illustrated in this example, IPMPA-encoded searching agents can represent DCNN topologies with various lengths, namely 3, 4, and 5.

3.2 Initializing population

Population size is determined after initialization and created until the population size is met. Each empty vector of predators contains a network interface, including the individual's IP address and subnet information. Each subsequent layer may contain a convolution, a pooling, or an enfeebled layer. Any four layer types may be employed before the final fully connected layer. The last layer is formed of a layer that is fully connected. Each layer's subnets will be issued a random IP address.

3.3 Fitness assessment

Prior to fitness evaluation, a weight initialization method must be chosen, and the Xavier weight initialization approach (Postel, 1980) is chosen since it has been demonstrated to be effective and is utilized in a large number of deep learning systems. For the first half of the training dataset, individuals will be trained using the settings for their previously decoded DCNN architecture (as described in Algorithm 2). After batch evaluating the partially trained DCNN on the second segment of the training dataset, a sequence of accuracy rates is created. Following that, we calculate the candid solution's fitness by averaging the accuracy rate of each participant.

Algorithm 2 The Evaluation of Fitness Function

```
Initialize
        The training epoch number ($k$)
        The Population ($P$)
        The batch size ($batch\_size$);
        The training set ($D_{train}$)
        The fitness evaluation dataset ($D_{fitness}$)
for candid solution $c$ in P
    $i = 1$
    $i = i + 1$
    while $i \le k$
        Train the DCNN's connection weights described by candid solution $c$;
    end while
$accuracy$ = Batch-evaluate the trained DCNN on the $D_{fitness}$ dataset with the $batch\_size$
store each batch's accuracy;
Calculate the mean ($accuracy$);
Fitness value ← mean ($accuracy$);
$P$ ← calculate the fitness value of the candid solution $c$
end for
return $P$
```

The next step is to alter the position of the candid solution by applying the coefficient values for each byte included inside it. Each interface in the candid solution vector should have a unique IP address given to it to ensure that they all have access to the same subnet. The restrictions of each interface, such as the second interface's ability to act as a convolution layer, pooling layer, or enfeebled layer, may change depending on its position in the overall solution vector.

4 Results of the experiment and discussion

Generally, we incorporate IPMPA to increase the DCNN's diagnostic accuracy. For all tests, the number of predators (population size) is limited to 50, and the maximum number of iterations is limited to 200. DCNN is designed to learn at a rate of 11 at a batch size of 0.0002. Additionally, the epochs for each assessment are chosen between 1 and 10, and it is underlined that pictures must be downsampled to 31×31 before being fed to DCNNs. All experiments were carried out in MATLAB R2019b on a computer provided with an Intel Core i7-7700HQ processor running at a maximum frequency of 3.8 GHz, Windows 10, and 16 GB RAM. This paper compares IPMPA's performance to that of IPPSO (Wang et al., 2018), variable-length genetic algorithm (VLGA) (Qiongbing and Lixin, 2016), variable-length NSGA-II (VLNSGA-II) (Pal et al., 2021), variable-length brain storm optimization algorithm (VLBSO) (Cheng et al., 2021), IP-Modified PSO (IPMPSO) (Abbas, 2018),

Table 6 The setting parameters and initial values

Algorithms	Parameters	Value
VLBBO	P_c	1
	The range of migration probability	[0, 1]
	Max (I) and Max(E)	1
	Step size	1
	Mutation probability	0.005
VLGA and VLNSGA-II	Type	Real coded
	Crossover	Single-point (1)
	Selection	Roulette wheel
	Mutation	Uniform (0.01)
VLACO	τ_0	0.000001
	Q	20
	q_0	1
	P_g	0.9
	P_t	0.5
	a	1
	β	5
VLBSO	m	5
	P_{5a}	0.2
	P_{6b}	0.8
	P_{6biii}	0.4
	P_{6c}	0.5
	k	20

variable-length biogeography-based optimizer (VLBBO) (Khishe et al., 2021), and variable-length ant colony optimization (VLACO) (Liao et al., 2013) on the two datasets used in the study. The MPA and other benchmark models' parameters are summarized in Table 6.

4.1 Metrics for evaluation

This research employs several performance indicators, including sensitivity, accuracy, specificity, F1-score, and precision. Equations 11–15represent the metrics' equations.

$$Sensitivity(TPR) = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{11}$$

$$Specificity(TNR) = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{12}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{13}$$

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

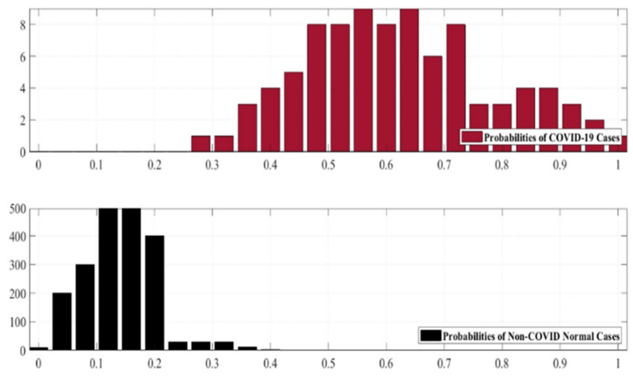


Fig. 7 The EPG for SARS-CoV-2 CT-Scan dataset

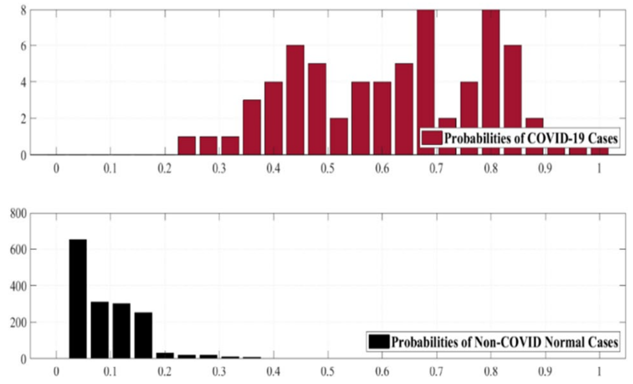
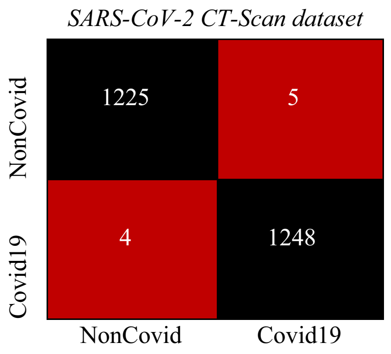


Fig. 8 The EPG for COVID-CT dataset

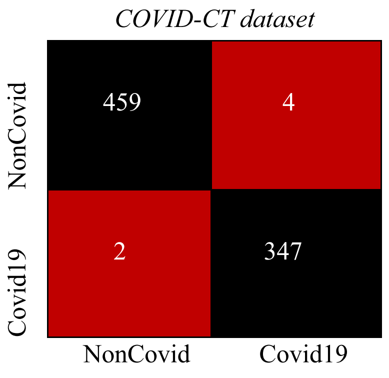


SARS-CoV-2 CT-Scan dataset	NonCovid	Covid19
NonCovid	1225	5
Covid19	4	1248

Fig. 9 The confusion matrix for the SARS-CoV-2 CT-Scan dataset

$$F_1 - score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{15}$$

TN denotes the number of true negative cases, TP denotes the number of true positive cases, FP denotes the number of false-positive cases, and FN denotes the number of false-negative cases.



COVID-CT dataset	NonCovid	Covid19
NonCovid	459	4
Covid19	2	347

Fig. 10 The confusion matrix for the COVID-CT dataset

4.2 Model performance analysis

As previously stated, while various resilient deep convolutional neural network designs have been developed recently, we choose the fundamental DCNN architecture due to the computational expense of this task. The first experiment assigns a probability rank to each image in the two datasets, SARS-CoV-2 CT-Scan and COVID-CT, reflecting the likelihood that each image is associated with COVID19 positive cases. This likelihood is then compared to a preset cut-off number in order to determine whether the inputs are infected. Notably, the planned model should obtain a probability of one for infected samples and zero for uninfected samples. Figures 7 and 8 illustrate the EPG distributions for SARS-CoV-2 and COVID-CT. Infected photos, by definition, have a higher probability than uninfected images. The confusion matrices for the SARS-CoV-2 CT-Scan and COVID-CT datasets are depicted in Figs. 9 and 10.

The specificity, sensitivity, precision, accuracy, and F1-Score of the DCNN-IPMPA and other benchmarks used in the second experiment, namely IPPSO, VLGA, VLNSGA-II, VLBSO, IPMPSO, VLBBO, and VLACO, are summarized in Tables 7 and 8. The results indicate that DCNN-IPMPA has the highest accuracy score, 97.21 percent for the SARS-CoV-2 dataset and 97.94 percent for the COVID-CT dataset. Additionally, the second-best result obtained utilizing DCNN-VLNSGA-II is 97.31 and 97.00 percent for SARS-CoV-2 CT-Scan and COVID-CT, respectively. Notably, the optimal outcome is highlighted in bold letters.

While a variety of metrics can be used to compare pattern recognizers, only one metric can provide a comprehensive view of the performance of benchmark algorithms across a range of thresholds, i.e., precision-recall curve. This diagram illustrates the relationship between recall and precision rate. The precision-recall curves for DCNN-IPMPA and eight benchmarks on the SARS-CoV-2 and COVID-CT datasets are shown in Figs. 11 and 12. The Receiver Operating

Table 7 The reults of benchmark models for SARS-CoV-2 dataset

Model	precision (%)	F1-Score (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)
DCNN	93.85	92.74	92.46	93.43	92.31
DCNN-IPPSO	94.43	93.20	93.14	93.11	92.44
DCNN-VLBSO	93.62	92.21	93.03	92.75	91.33
DCNN-IPMPSO	95.31	94.91	94.77	94.30	94.70
DCNN-VLGA	93.43	94.33	95.02	93.44	93.02
DCNN-VLBBO	95.32	95.72	95.77	98.32	95.78
DCNN-VLACO	96.31	95.22	95.33	95.51	95.33
DCNN-VLNSGA-II	96.98	96.26	96.84	95.90	**95.81**
DCNN-IPMPA	**97.21**	**96.98**	**97.31**	**96.21**	95.76

Bold indicates the optimal outcome

Table 8 The reults of benchmark models for COVID-CT dataset

Model	precision (%)	F1-Score (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)
DCNN	94.24	93.38	93.55	93.87	93.21
DCNN-IPPSO	95.55	94.41	93.02	92.01	93.46
DCNN-VLBSO	95.40	95.22	95.75	94.11	95.34
DCNN-IPMPSO	97.25	96.01	96.46	95.72	95.86
DCNN-VLGA	94.70	94.11	96.12	95.16	94.17
DCNN-VLBBO	96.55	95.33	96.99	95.89	94.65
DCNN-VLACO	96.18	96.66	95.33	95.28	95.41
DCNN-VLNSGA-II	97.21	96.56	97.99	95.33	**95.91**
DCNN-IPMPA	**97.94**	**96.44**	**97.00**	**95.22**	95.43

Bold indicates the optimal outcome

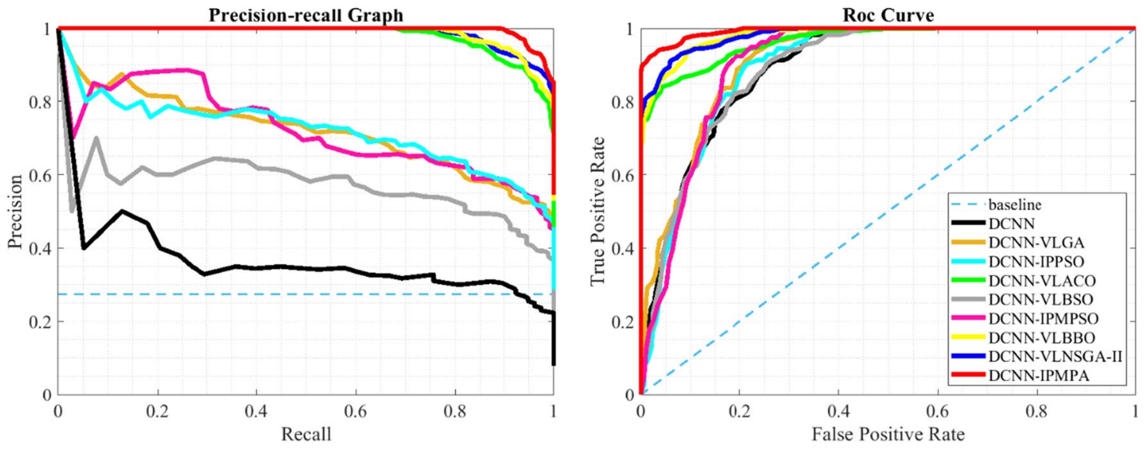


Fig. 11 The Precision-recall and ROC curves for the SARS-CoV-2 dataset

Characteristic (ROC) curve is another appropriate graph for illustrating the TPR as a function of the FPR. Thus, Figs. 11 and 12 depict the receiver operating characteristic (ROC) plots for DCNN-IPMPA and benchmarks, respectively. On both datasets, these figures demonstrate that DCNN-IPMPA surpasses other variable-length DCNNs.

When comparing metaheuristic algorithms, convergence speed is a critical metric to consider. In addition to the plots discussed above, Fig. 13 depicts the convergence curves of comparison benchmarks to enable comparisons.

The above figures show that the DCNN-IPMPA model surpasses all other comparable models. Additionally, it can

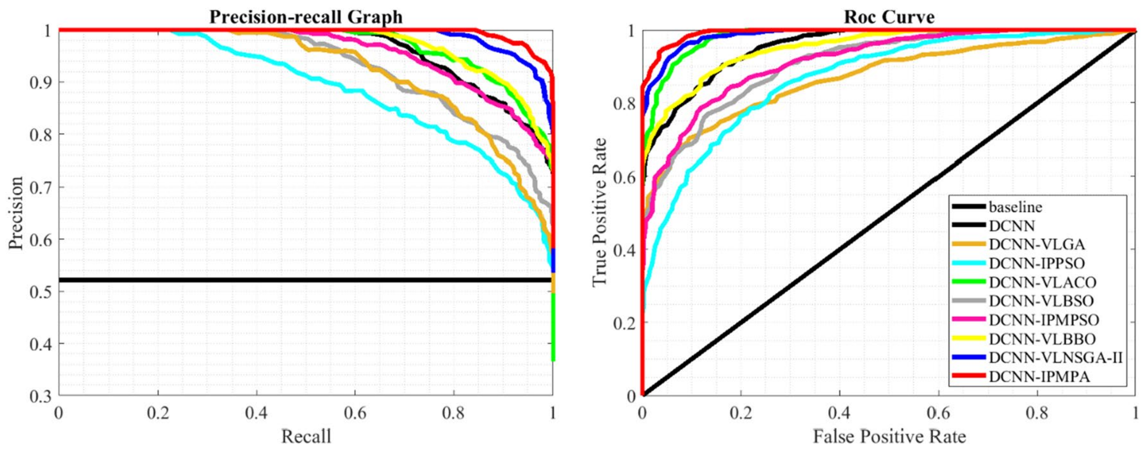


Fig. 12 The Precision-recall and ROC curves for the COVID-CT dataset

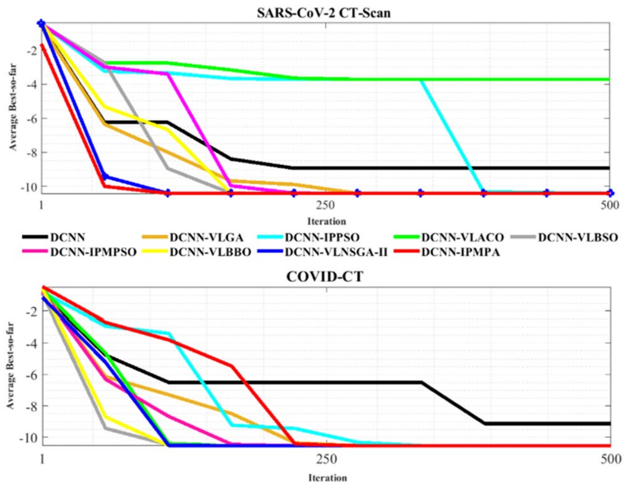


Fig. 13 The convergence curves for utilized benchmark algorithms

be stated that the canonical DCNN performs poorly compared to all other benchmarks.

4.3 The Analysis of time complexity

As previously stated, accuracy rate vs. processing time is a trade-off. In the preceding section, the accuracy of the proposed model was compared to that of various benchmarks. This section compares the processing time of the developed framework to ensure an accurate comparison. On an Intel Core i7-7700HQ CPU and an NVidia Tesla K20 GPU, we built the DCNN-IPMPA and comparison networks. The time values for the 2633 training images and 659 test images are shown in Table 9. Additionally, the best outcome is highlighted in bold type. Comparing the test and training times of benchmark networks implemented on GPU and CPU is shown in Table 9.

The results of Table 9 reveal that the DCNN-IPMPA outperforms other IP-based benchmarks by 61 s and 215 ms on the CPU and GPU, respectively. By considering the entire amount of processed images, i.e., 3392, it is concluded that DCNN-IPMPA requires less than 0.001 s for each image for testing and training. It must be noted that the DCNN processing time is the best among all benchmarks as expected; however, DCNN-IPMPA represents a competitive result compared to the standard DCNN.

4.4 Sensitivity analysis

This subsection assesses the sensitivity of the DCNN-control IPMPA's parameters. The parameter *FADs is deemed a* local optima avoidance operator for MPA. The second and third parameters (N_{Layer} and N_{batch}, respectively) address the network structure. The analytical results demonstrate the robustness and sensitivity of the parameters to input. Four different parameter values were used in the corresponding tests (Wu et al., 2021a, b). Numerous parameter combinations were established using an orthogonal array, as illustrated in Table 10. The model is trained for all potential parameter combinations. Additionally, Table 11 summarizes the mean square error (MSE) values obtained for each test. Figure 14 shows the trend in the parameter values concerning the data in Table 11. The result indicates that $N_{\text{Layer}}=5$, $FADs=0.2$, and $N_{\text{batch}}=10$ produce the best results.

5 Conclusion

This paper evaluates the MPA's effectiveness in determining the best structure of DCNNs without requiring manual effort to achieve both speed and accuracy. Three enhancements based on standard MPA were developed to accomplish the purpose. First, a novel encoding technique

Table 9 The comparison of test and training time of benchmark networks implemented on GPU and CPU

Model	CPU vs. GPU	Training time	Testing time	P value
DCNN	GPU	125 s	598 ms	0.0033
	CPU	1 h, 6 min, 1 s	54 s	0.0045
DCNN-IPPSO	GPU	1199 ms	824 ms	0.0075
	CPU	1 h, 14 min, 33 s	1 min, 55 s	0.0032
DCNN-VLBSO	GPU	746.2 ms	623 ms	0.0014
	CPU	1 h, 27 min, 52 s	2 min, 00 s	0.0002
DCNN-IPMPSO	GPU	2195 ms	928 ms	0.0001
	CPU	1 h, 16 min, 25 s	2 min, 11 s	0.0043
DCNN-VLGA	GPU	632 ms	415 ms	0.0056
	CPU	1 h, 58 min, 58 s	1 min, 48 s	0.0021
DCNN-VLBBO	GPU	744.6 ms	621 ms	0.0008
	CPU	2 h, 12 min, 3 s	1 min, 03 s	0.0008
DCNN-VLACO	GPU	1242 ms	822 ms	0.0014
	CPU	2 h, 5 min, 45 s	2 min, 02 s	0.0033
DCNN-VLNSGA-II	GPU	1101 ms	876 ms	0.0012
	CPU	2 h, 1 min, 1 s	2 min, 12 s	0.0004
DCNN-IPMPA	GPU	324 ms	215 ms	0.0007
	CPU	1 h, 8 min, 3 s	1 min, 01 s	**N/A**

Table 10 the parameters' specification

Level	N_{layer}	FADs	N_{batch}
1	3	0.1	6
2	4	0.2	8
3	5	0.3	10
4	6	0.4	12

Table 11 The MSE for various control parameter values

Experiments	Parameters			Result (MSE)
	N_{layer}	*FADs*	N_{batch}	
#1	1	4	1	0.0621
#2	1	3	2	0.0435
#3	1	2	3	0.0235
#4	1	1	4	0.0132
#5	2	1	2	0.0414
#6	2	2	1	0.0342
#7	2	3	4	0.0115
#8	2	4	3	0.0082
#9	3	4	1	0.0198
#10	3	3	4	0.0096
#11	3	1	2	0.0045
#12	**3**	**2**	**3**	**0.0018**
#13	4	2	4	0.0252
#14	4	3	3	0.0121
#15	4	4	2	0.0063
#16	4	1	1	0.0035

based on the IP address was proposed; then, an Enfeebled layer was presented to cover the dimensions of specified candid solution vectors. Finally, the learning process partitioned enormous datasets. The DCNN-IPMPA was applied to the SARS-CoV-2 and COVID-CT datasets to conduct a detailed comparison. The performance of the DCNN-IPMPA was compared to that of standard DCNNs, and DCNNs evolved using IPPSO, VLGA, VLNSGA-II, VLBSO, IP-MPSO, VLBBO, and VLACO using five well-known metrics: sensitivity, accuracy, specificity, F1-Score, and precision, as well as ROC and precision-recall curves. The study established that the presented system outperforms competing models, achieving a final accuracy of 97.21 percent on the SARS-CoV-2 dataset and 97.94 percent on the COVID-CT dataset. Timing analysis indicated that the DCNN processing time is the best among all benchmarks as expected; however, DCNN-IPMPA represents a competitive result compared to the standard DCNN. Numerous directions for future research study exist, including the application of DCNN-IPMPA to various image processing tasks. Researchers can explore the IPMPA in order to address multi-objective optimization problems. Additionally, a unique fitness function can be built to more accurately model the issue. As additional study directions, the use of oppositional-based learning, chaotic maps, orthogonal learning, and Gaussian walks can be investigated to optimize DCNN performance. Also, other DCNN structures or vision transformers can be used for future research studies.

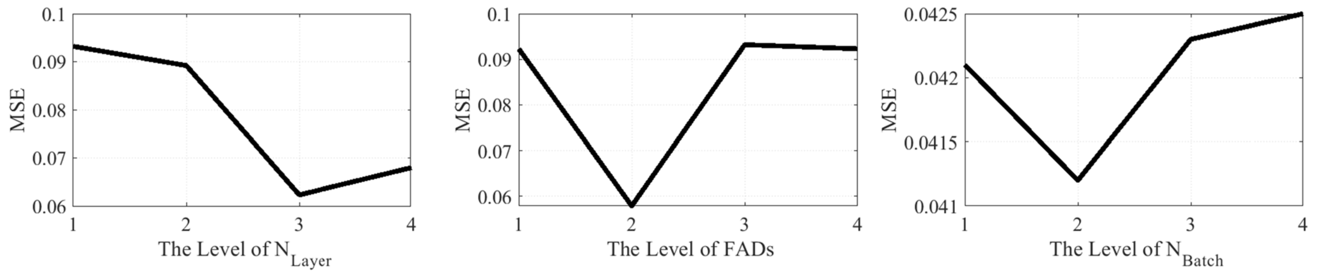


Fig. 14 The results of the control parameters

Funding Not Applicable.

Availability of data and material The resource images can be downloaded using the following links: https://github.com/UCSD-AI4H/COVID-CT; http://medicalsegmentation.com/covid19/

Declarations

Conflict of interest The authors declare that there is no conflict of interest.

References

Abbas SA (2018) Internet protocol (IP) steganography using modified particle swarm optimization (MPSO) algorithm. Diyala J Pure Sci 14:220–236

Afshar P, Heidarian S, Enshaei N, Naderkhani F, Rafiee MJ, Oikonomou A, Fard FB, Samimi K, Plataniotis KN, Mohammadi A (2021) COVID-CT-MD, COVID-19 computed tomography scan dataset applicable in machine learning and deep learning. Sci Data 8:1–8

Ai T, Yang Z, Hou H, Zhan C, Chen C, Lv W, Tao Q, Sun Z, Xia L (2020) Correlation of chest CT and RT-PCR testing in coronavirus disease 2019 (COVID-19) in China: a report of 1014 cases. Radiology 296:E32–E40

Angelov P, Almeida Soares E (2020) SARS-CoV-2 CT-scan dataset: a large dataset of real patients CT scans for SARS-CoV-2 identification. medRxiv.

Bernheim A, Mei X, Huang M, Yang Y, Fayad ZA, Zhang N, Diao K, Lin B, Zhu X, Li K (2020) Chest CT findings in coronavirus disease-19 (COVID-19): relationship to duration of infection. Radiology. https://doi.org/10.1148/radiol.2020200463

Breban R, Riou J, Fontanet A (2013) Interhuman transmissibility of Middle East respiratory syndrome coronavirus: estimation of pandemic risk. Lancet 382:694–699

Cao B, Zhao J, Liu X, Arabas J, Tanveer M, Singh AK, Lv Z (2022) Multiobjective evolution of the explainable fuzzy rough neural network with gene expression programming. IEEE Trans Fuzzy Syst.

Cheng J, Chen J, Guo Y, Cheng S, Yang L, Zhang P (2021) Adaptive CCR-ELM with variable-length brain storm optimization algorithm for class-imbalance learning. Nat Comput 20:11–22

Dai W, Zhang H-W, Yu J, Xu H, Chen H, Luo S, Zhang H, Liang L, Wu X, Lei Y (2020) CT imaging and differential diagnosis of COVID-19. Can Assoc Radiol J 71:195–200

Filmalter JD, Dagorn L, Cowley PD, Taquet M (2011) First descriptions of the behavior of silky sharks, Carcharhinus falciformis, around drifting fish aggregating devices in the Indian Ocean. Bull Mar Sci. https://doi.org/10.5343/bms.2010.1057

González G, Ash SY, Vegas-Sánchez-Ferrero G, Onieva Onieva J, Rahaghi FN, Ross JC, Díaz A, San José Estépar R, Washko GR (2018) Disease staging and prognosis in smokers using deep learning in chest computed tomography. Am J Respir Crit Care Med 197:193–203

Gozes O, Frid-Adar M, Greenspan H, Browning PD, Zhang H, Ji W, Bernheim A, Siegel E (2020) Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection and patient monitoring using deep learning ct image analysis. arXiv Prepr. arXiv:2003.05037.

Grewal M, Srivastava MM, Kumar P, Varadarajan S (2018) Radnet: radiologist level accuracy using deep learning for hemorrhage detection in ct scans. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). IEEE, pp. 281–284.

He X, Yang X, Zhang S, Zhao J, Zhang Y, Xing E, Xie P (2020) Sample-efficient deep learning for covid-19 diagnosis based on ct scans. MedRxiv 3:034501

Hu T, Khishe M, Mohammadi M, Parvizi G-R, Karim SHT, Rashid TA (2021a) Real-time COVID-19 diagnosis from X-Ray images using deep CNN and extreme learning machines stabilized by chimp optimization algorithm. Biomed Signal Process Control: 102764.

Hu T, Khishe M, Mohammadi M, Parvizi GR, Taher Karim SH, Rashid TA (2021b) Real-time COVID-19 diagnosis from X-Ray images using deep CNN and extreme learning machines stabilized by chimp optimization algorithm. Biomed Signal Process Control 68:102764. https://doi.org/10.1016/j.bspc.2021.102764

Jiang J, Wang X, Duan F, Liu W, Bu L, Li F, Li C, Sun Z, Ma S, Deng C (2019) Study of the relationship between pilot whale (Globicephala melas) behaviour and the ambiguity function of its sounds. Appl Acoust 146:31–37

Khishe M, Mohammadi H (2019) Passive sonar target classification using multi-layer perceptron trained by salp swarm algorithm. Ocean Eng. https://doi.org/10.1016/j.oceaneng.2019.04.013

Khishe M, Mosavi MR (2020a) Chimp optimization algorithm. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2020.113338

Khishe M, Mosavi MR (2020b) Classification of underwater acoustical dataset using neural network trained by Chimp Optimization Algorithm. Appl Acoust. https://doi.org/10.1016/j.apacoust.2019.107005

Khishe M, Caraffini F, Kuhn S (2021) Evolving deep learning convolutional neural networks for early COVID-19 detection in chest X-ray images. Mathematics 9:1002

Le Cun Y (2015) LeNet-5, convolutional neural networks 20:14. URL http://yann.lecun.com/exdb/lenet.

Liao T, Socha K, de Oca MAM, Stützle T, Dorigo M (2013) Ant colony optimization for mixed-variable optimization problems. IEEE Trans Evol Comput 18:503–518

Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26

Liu Z, Xiao X, Wei X, Li J, Yang J, Tan H, Zhu J, Zhang Q, Wu J, Liu L (2020) Composition and divergence of coronavirus spike proteins and host ACE2 receptors predict potential intermediate hosts of SARS-CoV-2. J Med Virol 92:595–601

Liu X, Zhao J, Li J, Cao B, Lv Z (2022a) Federated neural architecture search for medical data security. IEEE Trans Ind Inform.

Liu Y, Tian J, Hu R, Yang B, Liu S, Yin L, Zheng W (2022b) Improved feature point pair purification algorithm based on SIFT during endoscope image stitching. Front Neurorobot 16:840594. https://doi.org/10.3389/fnbot

Luo X, Yuan Y, Chen S, Zeng N, Wang Z (2020) Position-transitional particle swarm optimization-incorporated latent factor analysis. IEEE Trans Knowl Data Eng.

Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst. https://doi.org/10.1016/j.knosys.2015.12.022

Mosavi MR, Khishe M (2017) Training a feed-forward neural network using particle swarm optimizer with autonomous groups for sonar target classification. J Circ Syst Comput. https://doi.org/10.1142/S0218126617501857

Mosavi MR, Khishe M, Akbarisani M (2017) Neural network trained by biogeography-based optimizer with chaos for sonar data set classification. Wirel Pers Commun. https://doi.org/10.1007/s11277-017-4110-x

Ozturk T, Talo M, Yildirim EA, Baloglu UB, Yildirim O, Rajendra Acharya U (2020) Automated detection of COVID-19 cases using deep neural networks with X-ray images. Comput Biol Med. https://doi.org/10.1016/j.compbiomed.2020.103792

Pal R, Chaudhuri TD, Mukhopadhyay S (2021) Portfolio formation and optimization with continuous realignment: a suggested method for choosing the best portfolio of stocks using variable length NSGA-II. Expert Syst Appl 186:115732

Postel J (1980) DoD standard internet protocol. ACM SIGCOMM Comput Commun Rev 10:12–51

Qiao W, Khishe M, Ravakhah S (2021) Underwater targets classification using local wavelet acoustic pattern and multi-layer perceptron neural network optimized by modified whale optimization algorithm. Ocean Eng 219:108415. https://doi.org/10.1016/j.oceaneng.2020.108415

Qiongbing Z, Lixin D (2016) A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. Expert Syst Appl 60:183–189

Rastogi S, Choudhary S (2019) Face recognition by using neural network. Acta Inform Malaysia 3:7–9

Saffari A, Zahiri SH, Khishe M, Mosavi SM (2020) Design of a fuzzy model of control parameters of chimp algorithm optimization for automatic sonar targets recognition. IJMT.

Shah V, Keniya R, Shridharani A, Punjabi M, Shah J, Mehendale N (2021) Diagnosis of COVID-19 using CT scan images and deep learning techniques. Emerg Radiol. https://doi.org/10.1007/s10140-020-01886-y

Song Q, Zhao L, Luo X, Dou X (2017) Using deep learning for classification of lung nodules on computed tomography images. J Healthc Eng.

Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10:99–127

Suganuma M, Shirakawa S, Nagao T (2017) A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 497–504.

Wang B, Sun Y, Xue B, Zhang M (2018) Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In: 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp. 1–8

Wang Y, Yuan LP, Khishe M, Moridi A, Mohammadzade F (2020) Training RBF NN using sine-cosine algorithm for sonar target classification. Arch Acoust. https://doi.org/10.24425/aoa.2020.135281

Webb GI, Keogh E, Miikkulainen R, Miikkulainen R, Sebag M (2011) No-Free-Lunch Theorem. Encyclop Mach Learn. https://doi.org/10.1007/978-0-387-30164-8_592

Wu C, Khishe M, Mohammadi M, Karim SHT, Rashid TA (2021a) Evolving deep convolutional neutral network by hybrid sine–cosine and extreme learning machine for real-time COVID19 diagnosis from X-ray images. Soft Comput. https://doi.org/10.1007/s00500-021-05839-6

Wu J, Khishe M, Mohammadi M, Karim SHT, Shams M (2021b) Acoustic detection and recognition of dolphins using swarm intelligence neural networks. Appl Ocean Res 115:102837

Yang X, He X, Zhao J, Zhang Y, Zhang S, Xie P (2020) COVID-CT-dataset: a CT scan dataset about COVID-19.

Yuan Y, He Q, Luo X, Shang M (2020) A multilayered-and-randomized latent factor model for high-dimensional and sparse matrices. IEEE Trans Big Data.

Zhang Z, Wang L, Zheng W, Yin L, Hu R, Yang B (2022) Endoscope image mosaic based on pyramid ORB. Biomed Signal Process Control 71:103261

Zheng C, Deng X, Fu Q, Zhou Q, Feng J, Ma H, Liu W, Wang X (2020) Deep learning-based detection for COVID-19 from chest CT using weak label. MedRxiv 395:497