Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Artificial Intelligence In Medicine 119 (2021) 102156

Contents lists available at ScienceDirect

Artificial Intelligence In Medicine

journal homepage: www.elsevier.com/locate/artmed

Hybrid COVID-19 segmentation and recognition framework (HMB-HCF) using deep learning and genetic algorithms

Hossam Magdy Balaha [a,*], Magdy Hassan Balaha [b], Hesham Arafat Ali [a]

[a] Computer Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Egypt
[b] Obstetrics and Gynecology, Faculty of Medicine, Tanta University, Egypt

ARTICLE INFO

Keywords:
Classification
Convolutional neural network (CNN)
COVID-19
Data augmentation (DA)
Deep learning (DL)
Genetic algorithms (GA)
Optimization
Transfer learning (TL)

ABSTRACT

COVID-19 (Coronavirus) went through a rapid escalation until it became a pandemic disease. The normal and manual medical infection discovery may take few days and therefore computer science engineers can share in the development of the automatic diagnosis for fast detection of that disease. The study suggests a hybrid COVID-19 framework (named HMB-HCF) based on deep learning (DL), genetic algorithm (GA), weighted sum (WS), and majority voting principles in nine phases. Its segmentation phase suggests a lung segmentation algorithm using X-Ray images (named HMB-LSAXI) for extracting lungs. Its classification phase is built from a hybrid convolutional neural network (CNN) architecture using an abstractly-designed CNN (named HMB1-COVID19) and transfer learning (TL) pre-trained models (VGG16, VGG19, ResNet50, ResNet101, Xception, DenseNet121, DenseNet169, MobileNet, and MobileNetV2). The hybrid CNN architecture is used for learning, classification, and parameters optimization while GA is used to optimize the hyperparameters. This hybrid working mechanism is combined in an overall algorithm named HMB-DLGA. The study experiments implemented the WS approach to evaluate the models' performance using the loss, accuracy, F1-score, precision, recall, and area under curve (AUC) metrics with different pre-defined ratios. A collected, combined, and unified X-Ray dataset from 8 different public datasets was used alongside the regularization, dropout, and data augmentation techniques to limit the overall overfitting. The applied experiments reported state-of-the-art metrics. VGG16 reported 100% WS metric (i.e., 0.0097, 99.78%, 0.9984, 99.89%, 99.78%, and 0.9996 for the loss, accuracy, F1, precision, recall, and AUC respectively) concerning the highest WS. It also reported a 99.92% WS metric (i.e., 0.0099, 99.84%, 0.9984, 99.84%, 99.84%, and 0.9996 for the loss, accuracy, F1, precision, recall, and AUC respectively) concerning the last reported WS result. HMB-HCF was validated on 13 different public datasets to verify its generalization. The best-achieved metrics were compared with 13 related studies. These extensive experiments' target was the applicability verification and generalization.

1. Introduction

The novel coronavirus (COVID-19) appeared in late December 2019 in Wuhan, China which later (day-by-day) aroused a worldwide concern [1]. With the rapid escalation of that virus, the World Health Organization (WHO) declared on 16th March 2020 that the COVID-19 became a pandemic disease. Some countries reported a drop in the infection rate in July and August. A second infection wave started in October and currently, most are reporting rising cases [2]. This coronavirus until 4th January 2021 has already infected 218 countries (and territories) where the total number of cases is 85,801,672. The total number of deaths until that day is 1,855,931 which is about 2.16% of the total cases [3]. Most of the death cases occurred with very old people whose immunity is low (or faced health problems) [4]. USA, India, Brazil, Russia, UK, and France are the top six infected countries according to the published total number of cases [2].

The reported symptoms of the COVID-19 include fever, cough, and shortness of breath. It starts with a fever and dry cough and after 6 to 7 days, it can lead to the third symptom. The symptoms approximately last for two weeks in mild cases and three to six weeks for severe (or critical) cases [5]. A hospital-based study was performed on 138 patients and conducted that fever, fatigue (muscle pain), and dry cough were the most common symptoms [6].

Some acute respiratory syndromes have been recognized as being

* Corresponding author.
E-mail addresses: hossam.m.balaha@mans.edu.eg (H.M. Balaha), magdy.balaha@med.tanta.edu.eg (M.H. Balaha), h_arafat_ali@mans.edu.eg (H.A. Ali).

https://doi.org/10.1016/j.artmed.2021.102156
Received 15 February 2021; Received in revised form 8 August 2021; Accepted 17 August 2021
Available online 28 August 2021

caused by other strains of the coronavirus family; Severe Acute Respiratory Syndrome (SARS) and the Middle East Respiratory Syndrome (MERS). Imaging is a critical component of the diagnosis, monitoring, and follow-up in the coronavirus-related syndromes [7]. Imaging features in the COVID-19 are variable and nonspecific [8,9]. Investigators are making every effort to further characterize these imaging features, but the information is still limited.

X-Ray images datasets are one of the common public and available datasets. Many datasets (i.e., databases) of patients suffering from COVID-19 became public and available to be used since March 2020. This leads to the demand to try to discover and differentiate the lung radiological patterns (X-Ray and Computerized Tomography (CT)) in both the non-corona and the corona-infected people. Chest CT can produce fast imaging features, however, findings are highly sensitive and yet are non-specific [10]. There are two problems; the low specificity and the need for earlier diagnosis. This challenged the scientists and computer science (CS) engineers to try to find both earlier and accurate diagnoses. While the classical infection discovery might take a few days, CS engineers might share in the development of the automatic medical imaging recognition of the disease [11].

Pattern recognition (PR) is one of the major research fields that comprise the process of recognizing, identifying the different features, and extracting the differences (i.e., patterns) from the different types of inputs such as images [12]. A significant advancement in the recognition process was achieved with the integration of deep learning (DL) approaches; based on the Artificial Neural Networks (ANN) [13]. DL is used in detection [14], classification [15,16], and learning [14,17,18]. DL has many categories including Convolutional Neural Networks (CNNs), Deep Belief Networks (DBNs), Recurrent Neural Networks (RNNs), and stacked auto-encoders [19–21].

CNN is used in identifying, analyzing, and classifying visual imagery [22]. Convolutions and parallel processing are the main dependencies of CNNs. A CNN consists of multiple convolutional layers and fully-connected (FC) layers. It is easier and faster to train and has fewer parameters compared with the traditional ANNs, especially while working with images and videos. After training, CNN could reach convergence, generalization, acceptable results, and accurate decisions [23].

As imaging has paramount importance, yet it had low specificity. Hence, some DL approaches and systems have been proposed [24,25]. However, until now, there are no validated nor verified systems, that diagnose the COVID-19 automatically with the help of the computer vision (CV) approach. One of the great challenges is finding a suitable, official, and published dataset. The size of the dataset, in this case, is very important for segmentation, feature extraction, recognition, and classification to achieve generalization [26].

The major objective of the current study is to provide a hybrid framework that performs segmentation, classification, and recognition on the lung images of the COVID-19 disease with the help of CNNs and genetic algorithms (GA). CNNs are used in the learning and parameters optimization process while GA is used in the hyperparameters optimization process.

The contributions of the current study can be summarized as follows:

- Proposing a hybrid COVID-19 framework named HMB-HCF.
- Suggesting a lung segmentation algorithm using X-Ray images named HMB-LSAXI.
- Designing an abstract CNN model named HMB1-COVID19.
- Suggesting a combined DL and GA algorithm for learning and optimization named HMB-DLGA.
- Including a hybrid hierarchy model using HMB1-COVID19 and transfer learning pre-trained CNN models.
- Studying the effects of regularization, optimization, dropout, and data augmentation techniques through the different reported experiments.
- Reporting state-of-the-art performance metrics compared with other related works and approaches.

The paper is organized as follows: in the next section, we describe the previous related studies and works. In Section 3, The Proposed Hybrid COVID-19 Framework (HMB-HCF), we present and discuss the different internal phases of the proposed framework in detail. In Section 4, we report the experimental results and their discussion. Finally, Section 5 provides a conclusion for the study and presents future work.

2. Related work

Recently, there have been extensive researches in the field of recognition using DL and particularly for COVID-19 virus recognition. The studies are utilizing different tools, approaches, and variable datasets to facilitate the recognition of it. Until now, there are no verified public CS tools for the quantification of COVID-19 infection [11].

Bukhari et al. [27] used the transfer learning DL approach for COVID-19 recognition. They used ResNet50 [28] as their base model. Their CNN architecture was trained and tested on a dataset that consisted of 278 images. It was provided by the University of Montreal and the National Institutes of Health. Their used digital dataset of Chest X-Rays was divided into three groups labeled: “Normal”, “Pneumonia”, and “COVID-19”. They divided the dataset into 80% and 20% for training and testing respectively. The images were resized into (224, 224) pixels. They applied dropout [29] and data augmentation as well with different configurations: horizontal flipping, random rotation, random zooming, random lighting, and random wrapping [30]. Their analysis of the data was 98.18% and 98.19% for accuracy and F1-score respectively.

Gozes et al. [31] proposed a system that outputs a lung abnormality localization map and measurements by accepting the thoracic CT images as inputs. Their system applied 3D and 2D analysis. They used different datasets for their subsystems. Their experiments reached 0.996 Area Under Curve (AUC), 98.2% sensitivity, and 92.2% specificity.

Apostolopoulos et al. [32] proposed a CNN architecture with the help of transfer learning (TF) for COVID-19 detection. Their used dataset was a collection of 1428 X-Ray images (224 “COVID-19”, 700 common “Pneumonia”, and 504 “Normal”). They collected their used dataset from the different available X-Ray images on public medical repositories. They applied the Rectified Linear Unit (ReLU) [33] as the hidden activation functions, dropout technique to avoid overfitting, and Adam optimizer [34] in the training process. Their experiments with TF reported an overall accuracy of 97.82%.

Chowdhury et al. [35] provided an artificial intelligence (AI) approach. They created a public dataset using three public datasets by collecting images from some of the recently published articles. It contained a mixture of 190 COVID-19, 1345 viral Pneumonia, and 1341 Normal chest x-ray images. They applied data augmentation in the process of training and validation. They used four different pre-trained deep CNNs: AlexNet, ResNet18, DenseNet201 [36], and SqueezeNet [37]. These applied two different training schemes: (Normal and COVID-19 Pneumonia); and (Normal, Viral, and COVID-19 Pneumonia). Their reported classification accuracy, sensitivity, specificity, and precision for the two schemes were (98.3%, 96.7%, 100%, and 100%); and (98.3%, 96.7%, 99%, and 100%) respectively.

Abbas et al. [38] applied their previously developed CNN, named Decompose, Transfer, and Compose (DeTraC), for the classification problem of COVID-19. Their experimental results showed the DeTraC capability in detecting the COVID-19 cases from a comprehensive image dataset. They reported 93.10% and 100% for accuracy and sensitivity respectively.

Wang et al. [39] collected 453 CT images of pathogen-confirmed COVID-19 cases along with those previously diagnosed with typical viral pneumonia. They used 217 images for training the Inception model [40]. Their reported internal total accuracy was 82.9% with a specificity of 80.5% and sensitivity of 84%. They used an external testing dataset that reported a total accuracy of 73.1% with a specificity of 67% and sensitivity of 74%.

Abraham et al. [41] investigated the effectiveness of using multi-CNNs to detect COVID-19 from X-Ray images automatically. They used the correlation-based feature selection (CFS) technique and Bayesnet classifier for the COVID-19 prediction. They tested the used approach using two public datasets. The first dataset consisted of 453 COVID-19 images and 497 non-COVID images. Their approach achieved an AUC of 0.963 and an accuracy value of 91.16%. The second dataset consisted of 71 COVID-19 images and 7 non-COVID images. Their approach achieved an AUC of 0.911 and an accuracy value of 97.44%.

Islam et al. [42] introduced a DL technique based on the combination of a CNN and long short-term memory (LSTM) [43] to diagnose the X-Ray COVID-19 images automatically. The CNN was used for the feature extraction while the LSTM was used for the detection. They used a 4575 X-Ray images with 1525 COVID-19 images. Their experimental results achieved an accuracy of 99.4%, AUC of 99.9%, a specificity of 99.2%, a sensitivity of 99.3%, and an F1-score of 98.9%.

Polsinelli et al. [44] proposed a light CNN design. It was based on the SqueezeNet model. Their proposed modified SqueezeNet CNN achieved an accuracy of 85.03%, a sensitivity of 87.55%, specificity of 81.95%, a precision of 85.01%, and an F1-score of 86.20%. Aslan et al. [45] developed a COVID-19 diagnosis model using Multilayer Perceptron and CNN. Their architecture achieved a higher accuracy of 96.30%.

Bahgat et al. [46] suggested an optimized transfer learning approach for COVID-19 named OTLD-COVID-19. They used DenseNet (121, 169, and 201), Xception, MobileNet (V1, V2, V3 Small, and V3 Large), EfficientNetB0, and ResNet (50 V2, 101 V2, and 152 V2) pre-trained CNN models. They also used Manta-Ray Foraging Optimizer for the hyper-parameters optimization. They collected the used CT images from 8 different public datasets. Their best-reported accuracy was 98.47%.

Jain et al. [47] used the pre-trained CNN model, Xception, model for detecting Chest X-rays images. It reported the highest accuracy 97.97%. They used 6432 chest X-Ray images. Wang et al. [48] suggested a DL algorithm using CT images for COVID-19 detection. They modified the inception TF model to establish their algorithm. They applied their approach to the collected 1065 CT images.

3. The proposed hybrid COVID-19 framework (HMB-HCF)

The current section covers the detailed discussion of the proposed hybrid COVID-19 framework named HMB-HCF. It presents the whole process from the dataset (X-Ray images) acquisition to the exporting of the final results passing through data pre-processing, learning, recognition, optimization, and image augmentation as shown in Fig. 1.

The HMB-HCF consists of nine phases: (1) Images Acquisition Phase, (2) Pre-Processing Phase, (3) Segmentation Phase, (4) Data Augmentation Phase, (5) Splitting Phase, (6) Training, Classification, and Optimization Phase, (7) Prediction and Evaluation Phase, (8) Output and Export Phase, and (9) Hybrid Model Deployment Phase.

Basically, the first phase, Images Acquisition Phase, is responsible for retrieving the input image dataset. The images may vary from an environment to another. Hence, the second phase, Pre-Processing Phase, exists. This phase is responsible for removing the unrequired elements

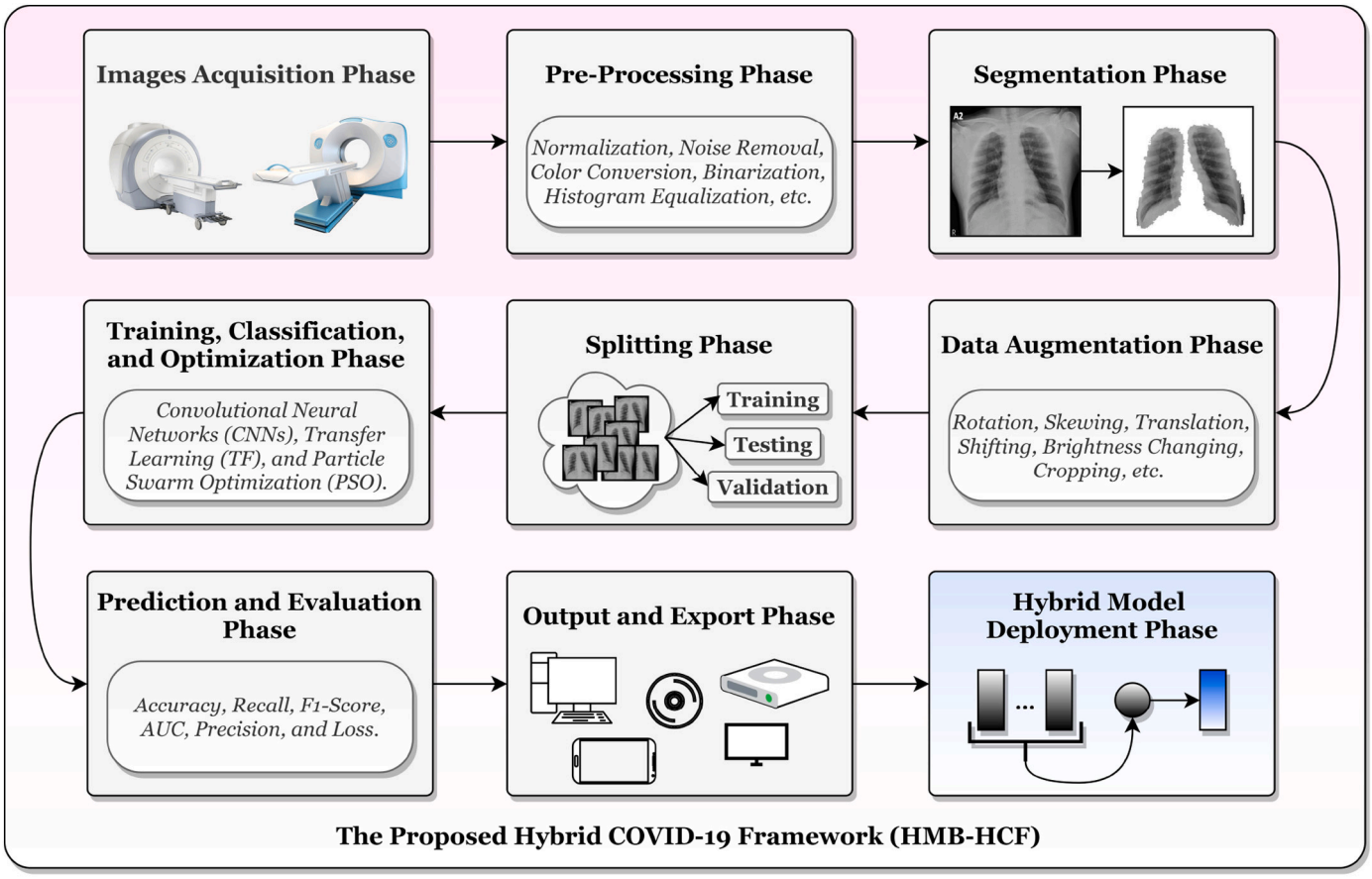


Fig. 1. The proposed hybrid COVID-19 framework (HMB-HCF).

from the scanned images such as noise, annotations, and markers. It includes a set of methods such as normalization, noise filtration, cropping, and skewing correction. If the dataset is well-prepared, the second phase can be bypassed.

The image may contain unrequired body elements such as arms that may interfere with the COVID-19 detection process. These elements should be removed to avoid any inferences. That's why the third phase, Segmentation Phase, is used. Public and accurate COVID-19 datasets are too few compared to the number of affected people. Hence, different techniques can be used to increase the diversity of the data. This happens in the fourth phase, Data Augmentation Phase. If the original images are required only to be used in the training and evaluation process, the fourth phase can be bypassed.

The dataset shall be split into training, testing, and validation subsets. The training and validation subsets are used to modify the model's parameters such as weights in the training phase and evaluate the performance after each epoch. The test subset will be used to measure the overall performance. This occurs in the fifth phase, Splitting Phase. In the sixth and seventh phases, the models will be trained, parameters and hyperparameters will be optimized, and the performance metrics will be calculated and reported.

After the training phase, the eighth phase, Output and Export Phase, exists. In this phase, the final results should be exported to be used later, either in another system or individually. It includes graph generation, final parameters and hyperparameters storage, and statistics generation. The most suitable weight initializer, optimizer, and regularizer should be selected according to a criterion that will be discussed and concluded later.

In the last phase, Hybrid Model Deployment Phase, a hybrid model is constructed from the optimized models and the decision is taken based on it. The internal details of each phase are discussed in the following subsections.

3.1. First phase: images acquisition phase

The first phase is responsible for retrieving the input image dataset. There are different ways of getting data from public ones to private patients' data. The main source of getting lung images is the X-Ray scanners. They are rotating X-Ray devices that create cross-sectional body images [49]. The drawbacks of that approach are (1) the difficulty of retrieving the images from the official organizations and (2) the time consumption to annotate (i.e., label) the retrieved images from professional doctors.

The second source is the public and accurate lung X-Ray datasets. The major advantage of the second approach is the ability to perform comparisons between different systems and published works.

The dataset must be annotated correctly to be used in any supervised learning approach such as machine learning (ML) and neural networks supervised techniques. If the dataset is not annotated, the decisions of a set of professional doctors can be obtained and their maximum decisions are taken. Fig. 2 shows a flowchart of the major sources of retrieving lung X-Ray scans and annotating them.

The flowchart starts by checking if the dataset is composed of X-Ray scans or not. If the choice is "No", then the right path (i.e., branch) is

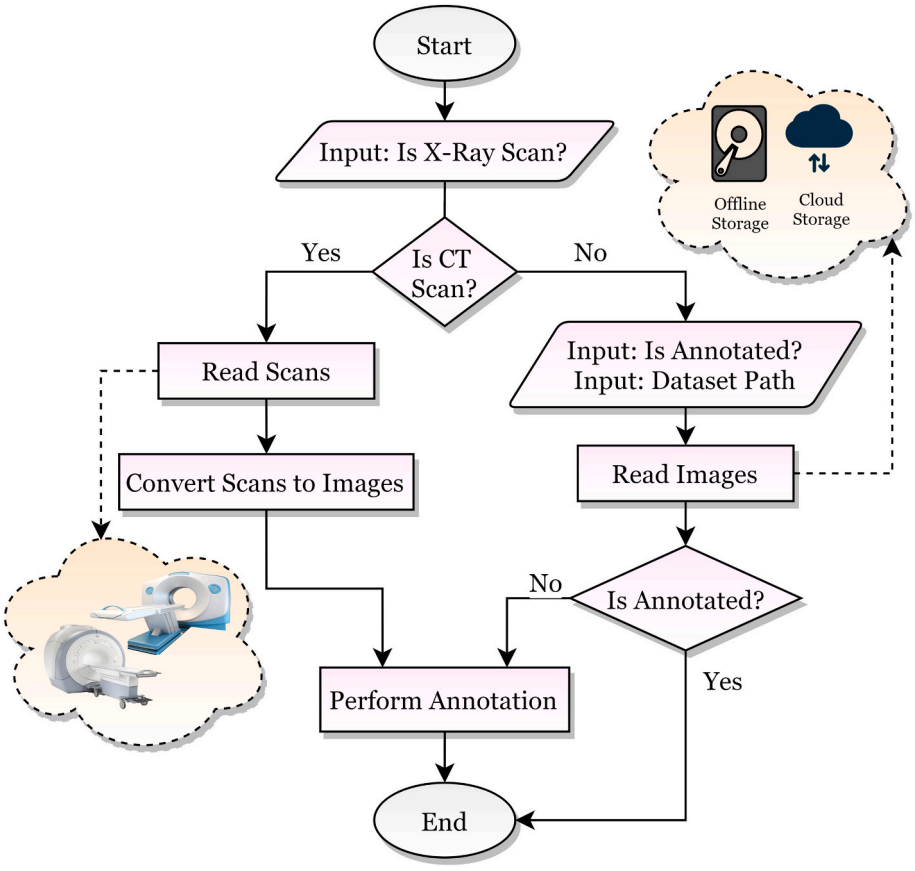


Fig. 2. The major sources of retrieving lung X-ray scans and their annotations.

Table 1
A simple annotation example.

Doctor no.	First sample	Second sample	Third sample
1	"COVID-19"	"Not-COVID-19"	"Pneumonia-Bacterial"
2	"COVID-19"	"COVID-19"	"Pneumonia-Viral"
3	"COVID-19"	"Not-COVID-19"	"Pneumonia-Viral"
4	"COVID-19"	"Not-COVID-19"	"COVID-19"
5	"COVID-19"	"Not-COVID-19"	"Pneumonia-Viral"
Decision	"COVID-19"	"Not-COVID-19"	"Pneumonia-Viral"

followed. It reads the stored images. The images can be retrieved from different storage resources such as hard disks or cloud storage. After that, it checks if they are annotated or not. If the images are not annotated, the annotation process is executed on each of the images. If the choice is "Yes", the left path is followed. It reads the X-Ray scans and converts them to images. After that, the annotation process comes out. The annotation process of a single image is the maximization of the professional doctors' decisions (i.e. majority voting principle) following Eq. (1).

$$\text{Decision} = \underset{c \in C}{max}(\text{Count}(c_1), \ldots, \text{Count}(c_N)) \quad (1)$$

where N is the number of used categories (i.e., classes) such as "COVID-19", "Not-COVID-19", and "Pneumonia", c is a class from the used C classes. Table 1 shows an example on the annotation process. It shows five doctors annotating three samples.

It is worth mentioning that, the words "categories", "labels", and "classes" are synonyms and refer to the same meaning in the current study. Table 2 shows some of the available online lung X-Ray images datasets that can be used.

The "Pneumonia (virus) vs COVID-19" [50] dataset contains 70 COVID-19 and 1493 Pneumonia X-Rays. The "Covid-19 Xray images using CNN" [51] dataset contains 140 COVID-19 and 144 Normal X-Rays. The "COVID-19 X-ray Images5" [52] dataset contains 91 COVID-19 and 1352 Normal X-Rays.

The "COVID-19 Patients Lungs X Ray Images 10,000" [53] dataset contains 70 COVID-19 and 28 Normal X-Rays. The "COVID-19 Chest X Rays" [54] dataset contains 69 COVID-19 and 79 Normal X-Rays. The "COVID-19 Dataset" [55] dataset contains 25 COVID-19 and 25 Normal X-Rays.

The "Curated Chest X-Ray Image Dataset for COVID-19" [56] dataset contains 1281 COVID-19, 3270 Normal, 1656 Pneumonia-Viral, and 3001 Pneumonia-Bacterial X-Rays. It was obtained by collating 15 publically available datasets and removing the duplicates based on the image similarities. The "Chest X-ray (Covid-19 & Pneumonia)" [57] dataset contains 576 COVID-19, 1583 Normal, and 4273 Pneumonia X-Rays. The "COVID19 with Pneumonia and Normal Chest Xray (PA) Dataset" [58] dataset contains 2313 COVID-19, 2313 Normal, and 2313 Pneumonia X-Rays.

The "COVID-19 Xray Dataset (Train & Test Sets)" [59] dataset contains 94 Pneumonia and 94 Normal X-Rays. The "COVID19-xray" [60] dataset contains 1161 COVID-19 X-Rays. The "Chest Xray for covid-19 detection" [61] dataset contains 174 COVID-19 and 174 Normal X-Rays.

The "covid_19_2020" [62] dataset contains 3594 COVID-19, 5583 Normal, 2.313 Pneumonia (common), 3001 Pneumonia-Bacterial, and 1656 Pneumonia-Viral X-Rays. The "COVID-19 Detection X-Ray Dataset" [63] dataset contains 189 COVID-19, 2231 Normal, 1624 Pneumonia-Bacterial, and 1029 Pneumonia-Viral X-Rays. The "COVID-19 & Normal Posteroanterior(PA) X-rays" [64] dataset contains 140 COVID-19, and 140 Normal X-Rays. The "COVID-19 Radiography Dataset" [65] dataset contains 3616 COVID-19, 10,192 Normal, 6012 Lung Opacity, and 1345 Pneumonia-Viral X-Rays.

The "Covid-GAN and Covid-Net mini Chest X-ray" [66] dataset contains 972 COVID-19, 2083 Normal, and 4489 Pneumonia X-Rays. The "COVID19_Pneumonia_Normal_Chest_Xray_PA_Dataset" [67] dataset contains 1525 COVID-19, 1525 Normal, and 1525 Pneumonia X-

Table 2
The available online lung X-Ray images datasets.

No.	Name	Classes	Size	Link
1	"Pneumonia (virus) vs COVID-19" [50]	2	1563	https://www.kaggle.com/muhammadmasdar/pneumonia-virus-vs-covid19
2	"Covid-19 Xray images using CNN" [51]	2	284	https://www.kaggle.com/akkinasrikar/covid19-xray-images-using-cnn
3	"COVID-19 X-ray Images5" [52]	2	1443	https://www.kaggle.com/uddiptadas/covid19-xray-images5
4	"COVID-19 patients lungs X ray images 10,000" [53]	2	98	https://www.kaggle.com/nabeelsajid917/covid-19-x-ray-10000-images
5	"COVID-19 chest X rays" [54]	2	148	https://www.kaggle.com/rupeshs/covid19-chest-x-rays
6	"COVID-19 dataset" [55]	2	50	https://www.kaggle.com/syedrz/covid19-dataset
7	"Curated chest X-ray image dataset for COVID-19" [56]	4	9208	https://www.kaggle.com/unaissait/curated-chest-xray-image-dataset-for-covid19
8	"Chest X-ray (Covid-19 & pneumonia)" [57]	3	6432	https://www.kaggle.com/prashant268/chest-xray-covid19-pneumonia
9	"COVID19 with pneumonia and normal chest Xray (PA) dataset" [58]	3	6939	https://www.kaggle.com/amanullahasraf/covid19-pneumonia-normal-chest-xray-pa-dataset
10	"COVID-19 Xray dataset (train & test sets)" [59]	2	188	https://www.kaggle.com/khoongweihao/covid19-xray-dataset-train-test-sets
11	"COVID19-xray" [60]	1	1161	https://www.kaggle.com/anaselmasry/covid19xray
12	"Chest Xray for covid-19 detection" [61]	2	348	https://www.kaggle.com/fusicfenta/chest-xray-for-covid19-detection
13	"covid_19_2020" [62]	5	16,147	https://www.kaggle.com/tikoboss/covid-19-2020
14	"COVID-19 detection X-ray dataset" [63]	4	5073	https://www.kaggle.com/darshan1504/covid19-detection-xray-dataset
15	"COVID-19 & Normal Posteroanterior(PA) X-rays" [64]	2	280	https://www.kaggle.com/tarandeep97/covid19-normal-posteroanteriorpa-xrays
16	"COVID-19 radiography dataset" [65]	4	21,165	https://www.kaggle.com/preetviradiya/covid19-radiography-dataset
17	"Covid-GAN and Covid-Net mini chest X-ray" [66]	3	7544	https://www.kaggle.com/yash612/covidnet-mini-and-gan-enerated-chest-xray
18	"COVID19_Pneumonia_Normal_Chest _Xray_PA_Dataset" [67]	3	4575	https://www.kaggle.com/asraf047/covid19-pneumonia-normal-chest-xray-pa-dataset
19	"Chest X-ray images" [68]	2	5856	https://www.kaggle.com/tolgadincer/labeled-chest-xray-images
20	"Chest Xray images pneumonia and Covid-19" [69]	3	6118	https://www.kaggle.com/masumrefat/chest-xray-images-pneumonia-and-covid19

Rays. The "Chest X-ray Images" [68] dataset contains 1583 Normal, and 4273 Pneumonia X-Rays. The "Chest Xray Images PNEUMONIA and Covid-19" [69] dataset contains 262 COVID-19, 1583 Normal, and 4273 Pneumonia X-Rays.

3.2. Second phase: pre-processing phase

The second phase is responsible for enhancing the images and removing the unrequired elements from them such as noise, annotations, and markers. Different methods can be used such as normalization, binarization, histogram equalization, noise filtration, cropping, and skewing correction. If the dataset is well-prepared, this phase can be bypassed. Eq. (2) shows the used normalization method. The value of 255.0 refers to the maximum pixel value in any of the image channels.

$$\text{Data}_{\text{Normalized}} = \frac{\text{Data}}{255.0} \tag{2}$$

The remaining used methods in the current paper discussion are combined with the proposed lung segmentation algorithm using X-Ray images discussion in the next subsection.

3.3. Third phase: Segmentation phase

The third phase segments only the lungs by neglecting the shoulders, arms, and external bones as the COVID-19 remains in the lungs themselves. This paper suggests a lung segmentation algorithm using X-Ray images named HMB-LSAXI and its pseudocode is shown in Algorithm 1. It is worth mentioning that the current subsection's following paragraphs present the base details of the suggested algorithm and more information can be retrieved from the used references.

Algorithm 1. Proposed lung segmentation algorithm using X-ray images, HMB-LSAXI, pseudocode.

3.3.1. Algorithm line 1

The HMB-LSAXI algorithm starts by reading the X-Ray colored lung image using the input image path "*imgPath*"'. A colored image has 3 channels ($ch = 3$), Red, Green, and Blue (RGB), while a grayscale image has only one channel ($ch = 1$).

3.3.2. Algorithm line 2

The image is converted to a grayscale image after that. There are different methods for that conversion such as average, channel-dependent luminance perception, gamma compression, and linear approximation [70–72]. The last method, linear approximation, is the used method in the algorithm. It implements Eq. (3) for each pixel.

$$G_r = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{3}$$

where G_r is the resultant grayscale pixel value, and R, G, and B are the red, green, and blue values respectively.

3.3.3. Algorithm line 3

Gaussian blurring is applied after that to remove the unrequired noise [73,74]. It follows Eq. (4) to get the kernel that will be applied to each pixel.

$$\text{Gauusian}_{\text{kernel}} = \frac{1}{2 \times \pi \times \sigma^2} \times e^{\left(\frac{x_k^2 + y_k^2}{-\frac{x_k^2+y_k^2}{2 \times \sigma^2}}\right)} \tag{4}$$

where σ is the standard deviation of the distribution and equals 1 using Gaussian. x_k and y_k are the values of the coordinates of the kernel. The used kernel size is (15×15).

3.3.4. Algorithm line 4

The blurred image is binarized [75] after that by setting the pixels values to 255 if they are less than or equal to the mean value and 0 otherwise as shown in Eq. (5).

Input: *imgPath* // X-Ray Lung Image Path
Output: *lungs*, *ratio* // Processed X-Ray Lung Image, Lungs Size Percentage in the Image

1 $img = \text{ReadImage}(imgPath)$ // Read the Colored RGB Image using the Image Path
2 $grayImg = \text{Convert2Gray}(img)$ // Convert the Colored Image to a Grayscale Image
3 $blurredImg = \text{GaussianBlur}(grayImg, 15 \times 15)$ // Apply Gaussian Blur on the Grayscale Image with a Kernel Size of 15×15
4 $bwImg = \text{Binarize}(blurredImg)$ // Apply Binarization by Setting the Pixels' Values that are Less than or Equal to the Mean to 255 and to 0 Otherwise
5 $kernel1 = \text{CreateOnesKernel}(2 \times 2)$ // Create a Matrix (Kernel) Filled with Ones with a Size of 2×2
6 $erodedImg = \text{ApplyErode}(bwImg, kernel1, iterations = 7)$ // Apply Erode Algroithm on the Binary Image using the Created Kernel for 7 Iterations
7 $floodedImg = \text{ApplyFloodFill}(erodedImg)$ // Apply Flood Fill Algroithm on each Row and Column on the Endpoints
8 $kernel2 = \text{CreateOnesKernel}(15 \times 15)$ // Create a Matrix (Kernel) Filled with Ones with a Size of 15×15
9 $dilatedImg = \text{ApplyDilate}(floodedImg, kernel2, iterations = 3)$ // Apply Dilate Algroithm using the Created Kernel for 3 Iterations
10 $contours = \text{FindLargest2Contours}(dilatedImg)$ // Find the Image Contours, Sort them according to their Sizes in a Descending Order, Extract the First Largest Two Sized Contours, Fill in these Contours, and Apply Bitwise XOR between the Filled Contours and the Dilated Image, *dilatedImg*
11 Set $foreground = contours$
12 $kernel3 = \text{CreateEllipsedKernel}(5 \times 5)$ // Create an Ellipsed Matrix (Kernel) with a Size of 5×5
13 $background = \text{ApplyDilate}(foreground, kernel3, iterations = 5)$ // Apply Dilate Algroithm using the Created Kernel for 5 Iterations
14 $markers, markerImg = \text{Watershed}(img, foreground, background)$ // Apply Watershed Algorithm using the Original, Foreground, and Background Images
15 $first, second, lungs = \text{ExtractLungs}(markers, markerImg)$ // Extract the Lungs (Left, Right, and Both) using the Markers and Marked Images
16 $ratio = \text{CalculateLungsRatio}(lungs)$ // Calculate the Lungs Size Percentage in the Image

$$\text{bw}_{(i,j)} = \begin{cases} 0 & \text{if } \text{gray}_{(i,j)} > \text{mean}_{\text{gray}} \\ 255 & \text{Otherwise} \end{cases} \quad (5)$$

where i and j are the image row and column respectively. The image mean value, $\text{mean}_{\text{gray}}$, is calculated using Eq. (6). The foreground pixels will be at an intensity value of 255 and the background pixels will be at an intensity value of 0.

$$\text{mean}_{\text{gray}} = \frac{\sum_{i=1}^{w}\sum_{j=1}^{h}\text{gray}_{(i,j)}}{w \times h} \quad (6)$$

where w and h are the width and height sizes of the image respectively.

3.3.5. Algorithm line 6

The generated binary image may contain numerous imperfections that can be removed using the morphological image processing algorithms [76]. Erosion is one of the basic operators in the area of mathematical morphology [77]. It is applied to the binarized images, but there are alternative versions that handle the grayscale images. The basic effect of it on a binary image is to erode the boundaries of regions of foreground pixels. These areas of the foreground pixels shrink in size and the holes or gaps within these areas become larger. Thus, it is useful in removing small white noises and detaching connected objects [78].

The image is eroded after binarization for 7 iterations using a structuring element (i.e., kernel) sized (2 × 2) and filled with ones (in Algorithm Line 5). Each iteration is performed individually. Technically, the value of a pixel is set to the minimum value of all of the pixels covered by the kernel when aligned at that pixel as shown in Eq. (7).

$$\text{eroded}_{(i,j)} = \min_{(i_k,j_k)\in K} \text{bw}_{(i+i_k,j+j_k)} \quad (7)$$

where i_k and j_k are the kernel's row and column respectively and K is the kernel matrix.

3.3.6. Algorithm line 7

A flood fill algorithm is used after eroding the binary image [79]. It fills a connected component with a specific color (i.e., black in our case) [80]. In our case, the image boundaries are required to be converted to black pixels if they were white. So the background is completely black and the foreground that contains the lungs is completely white. Hence, a loop is applied to each row on the endpoints of it. If the endpoint is white, the flooding with black is applied on the row. After that, another loop is applied to each column on the endpoints to perform the same operation.

3.3.7. Algorithm line 9

Similar to erosion, dilation is another basic operator in the area of mathematical morphology and it is the reverse process of erosion. The basic effect of it on a binary image is to enlarge the elements in size and

Table 3
The used data augmentation (DA) techniques and their configurations.

Technique	Configurations
Rotation range	±15°
Width shift range	±10%
Height shift range	±10%
Zoom range	±20%
Shear range	±20%
Horizontal flipping	Yes

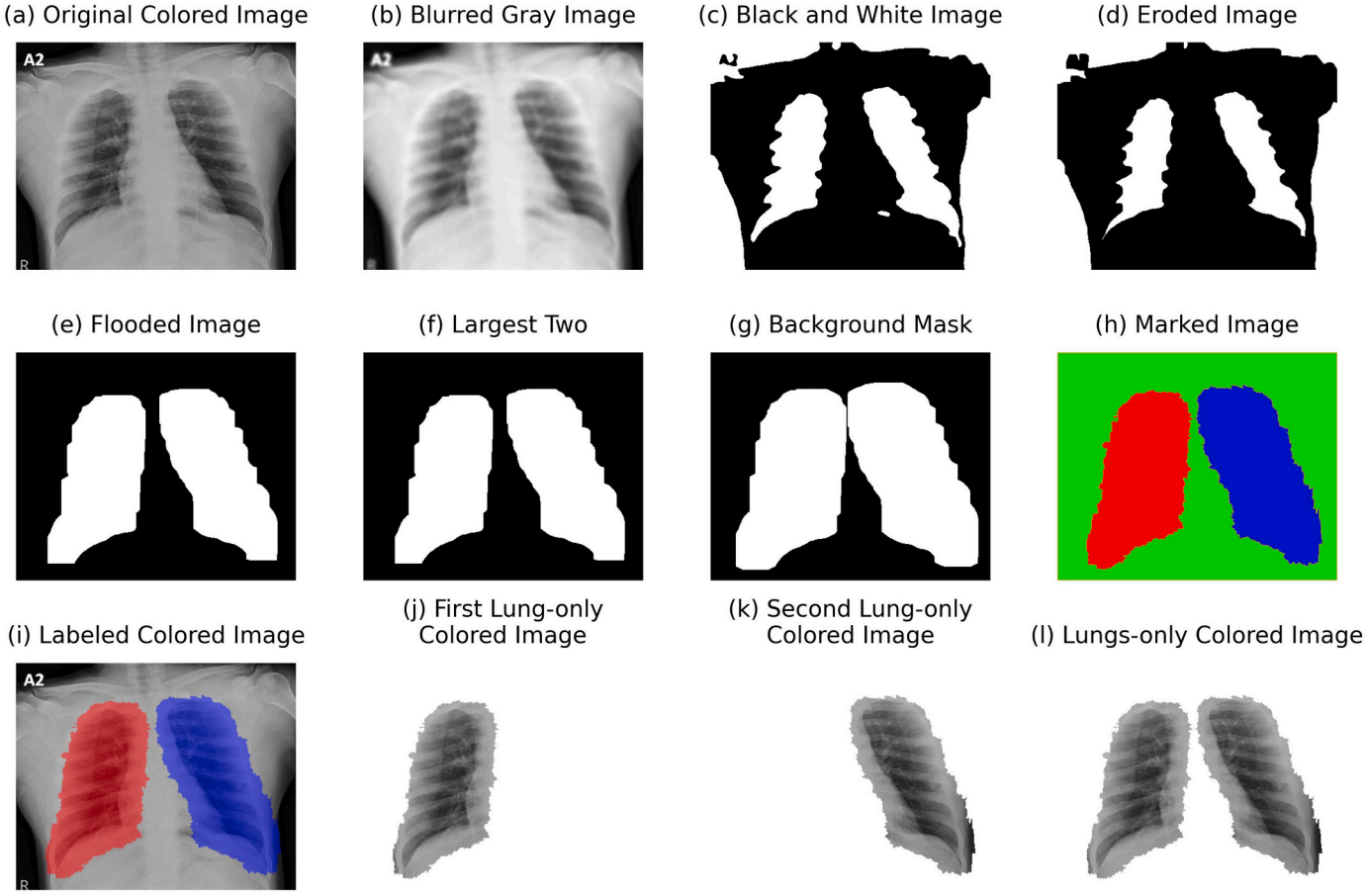


Fig. 3. Graphical illustration of the HMB-LSAXI major steps using a sample X-ray lung scan.

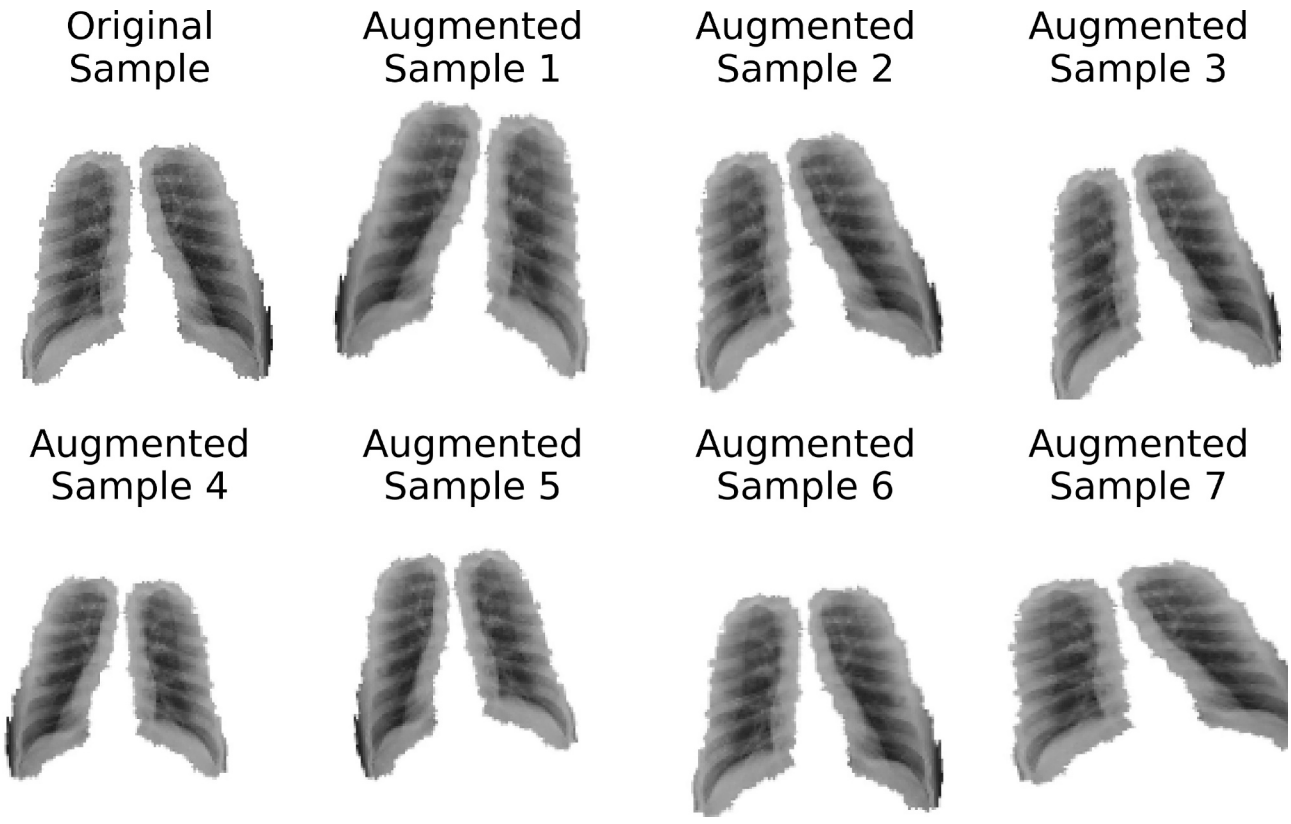


Fig. 4. Graphical illustration of some of the data augmentation (DA) techniques using a sample X-Ray lung scan.

connecting broken objects [81]. The image is dilated after that for 3 iterations using a structuring element sized (15 × 15) and filled with ones (in Algorithm Line 8). Each iteration is performed individually. Technically, the value of a pixel is set to the maximum value of all of the pixels covered by the kernel when aligned at that pixel as shown in Eq. (8).

$$\text{dilated}_{(i,j)} = \max_{(i_k,j_k)\in K} \text{bw}_{(i+i_k,j+j_k)} \tag{8}$$

3.3.8. Algorithm line 10

The next step is to find the contours of the dilated image. Contours can be explained as a curve joining process of all of the connected points along the boundary that have the same intensity [82]. The contours are very useful for object detection and recognition [83]. Different contours can be extracted from the image but the target is to find the largest two contours that represent the lungs. Hence, the contours are sorted according to their areas in descending order and the first two contours are extracted. Before extraction, a check must be applied to check if the number of contours is less than two. If so, the process should terminate which indicates that the lungs are not found. Another check can be applied after extraction to check if the two extracted contours are near in size or not. If not, the process should terminate too. The size check condition is shown in Eq. (9).

$$\text{condition} = \begin{cases} \text{True} & \text{if } \dfrac{\text{Area}(\text{contour}_2) \times 100}{\text{Area}(\text{contour}_1)} \geq 50\% \\ \text{False} & \text{Otherwise} \end{cases} \tag{9}$$

where contour_1 and contour_2 are the first and second largest contours respectively.

3.3.9. Algorithm line 14

The extracted two contours are filled and a bitwise XOR operation is applied between the flooded binary image and the filled contours to refine the image and create the masks for the Watershed algorithm [84]. The Watershed algorithm is a marker-based interactive algorithm used for segmentation [85]. It is useful when extracting overlapping objects

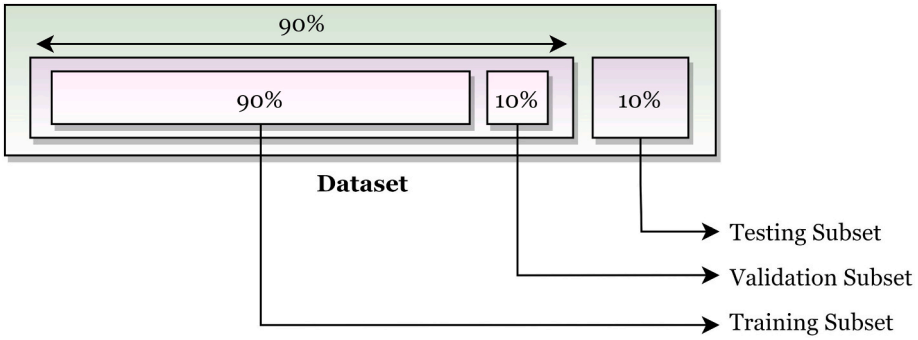


Fig. 5. Graphical illustration of the split process.

Table 4
The used models' number of parameters summarization.

#	Model	Number of parameters
1	HMB1-COVID	321,572
2	MobileNetV2	2,263,108
3	MobileNet	3,232,964
4	DenseNet121	7,041,604
5	DenseNet169	12,649,540
6	VGG16	14,716,740
7	VGG19	20,026,436
8	Xception	20,869,676
9	ResNet50	23,595,908
10	ResNet101	42,666,372

in images [86]. By applying the Watershed algorithm, the left and right colored lungs can be extracted from the background. The background is set to be white intensity in Algorithm Line 13: So, the only remaining elements in the colored image are the extracted lungs. The last check should be applied to ensure if the Watershed algorithm extracted the lungs successfully or not. If the sum of non-white pixels of the lungs to the size of the image is >10% or not. If true, we can consider the process succeeded to extract the lungs (in Algorithm Line 15 and Algorithm Line 16). The outputs of the algorithm are the extracted lungs and the computed lungs ratio.

Fig. 3 shows a graphical illustration of the HMB-LSAXI major steps using a sample X-Ray lung scan.

3.4. Fourth phase: data augmentation phase

Data augmentation (DA) is used to increase the diversity of relevant images artificially and avoid overfitting [87]. It is an essential part of training the models especially if there are few records or images. There are different DA techniques such as flipping, rotation, shifting, cropping, and brightness changing [30]. Flipping can be performed in horizontal, vertical, or both directions. Rotation can be applied circularly from 1 to 360 degrees. The image can be scaled outward (larger) or inward (smaller). A region of interest (ROI) can be cropped from the image and this is also a DA technique. Translation can be applied by shifting the image in X-, Y-, or both directions. Noise can help in that too by applying Gaussian noise [88].

There are also advanced DA techniques such as Generative Adversarial Networks (GANs) [89]. GAN is used in generative modeling using DL methods such as CNN. There are different derivatives of GANs such as Auxiliary Classifier Generative Adversarial Network (ACGAN) [90], Bidirectional Generative Adversarial Network (BiGAN) [91], and Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks (CCGAN) [92].

DA in the current paper is applied in two locations. The first is applied before training and splitting to equalize the number of images in each category while the second is applied in the training process to avoid overfitting and reach better generalization.

Using DA, before training and splitting, helps to equalize the number of records in each category. For example, there are three categories with a number of images 1000, 500, and 100 respectively. There is a large gap between the number of images in each category. Hence, we suggest applying DA before continuing to the next phase. This occurs by finding the maximum category with the maximum number of images (i.e., the first category in our example). For other categories, a ratio is calculated using Eq. (10) to determine the number of augmented images required for each image (e.g., each image in the third category should have 4 more augmented images in addition to the original one).

$$\text{Ratio}_{ci} = \left\lceil \frac{max_{c \in C} \text{Count}(X_c)}{\text{Count}(X_{ci})} \right\rceil - 1 \tag{10}$$

where X_c is the images in a c category and X_{ci} is the images in the ci category. The used configurations in the DA process are shown in Table 3. Fig. 4 shows a graphical illustration of some of the DA techniques applied on a sample X-Ray lung scan.

3.5. Fifth phase: splitting phase

The dataset shall be split into training, testing, and validation subsets after shuffling it. The training subset is used to update the model parameters. The validation subset is used to measure the performance of the model after each epoch. The test subset is used to measure the model's overall performance after the learning process. In the current study, the dataset is first split into 90% for training and validation and 10% for testing. The 90% portion is split again into 90% for training and 10% for validation. Fig. 5 shows a graphical illustration of the split process.

3.6. Sixth phase: training, classification, and optimization phase

The current subsection presents the used and suggested architectures using CNNs and TF and their parameters optimization process using well-known optimizers. It can be summarized in two points:

Table 5
Tabular representation of the proposed CNN architecture (HMB1-COVID19).

Block number	Layer description	Number of filters	Filter size	Pool size	Stride size	Padding	Other info
	Input layer						Image size (w, h, ch)
First block	Convolutional	32	(3, 3)		(1, 1)	1	10^{-4} L2 regularization
First block	Batch normalization						
First block	Convolutional	32	(3, 3)		(1, 1)	1	10^{-4} L2 regularization
First block	Batch normalization						
First block	Max-pooling			(2, 2)	(2, 2)	0	
First block	Dropout						25% dropout ratio
Second block	Convolutional	64	(3, 3)		(1, 1)	1	10^{-4} L2 regularization
Second block	Batch normalization						
Second block	Convolutional	64	(3, 3)		(1, 1)	1	10^{-4} L2 regularization
Second block	Batch normalization						
Second block	Max-pooling			(2, 2)	(2, 2)	0	
Second block	Dropout						25% dropout ratio
Third block	Convolutional	128	(3, 3)		(1, 1)	1	10^{-4} L2 regularization
Third block	Batch normalization						
Third block	Convolutional	128	(3, 3)		(1, 1)	1	10^{-4} L2 regularization
Third block	Batch normalization						
Third block	Max-pooling			(2, 2)	(2, 2)	0	
	Dropout						25% dropout ratio
Third block	FC flatten						
	FC dense	N					SoftMax function

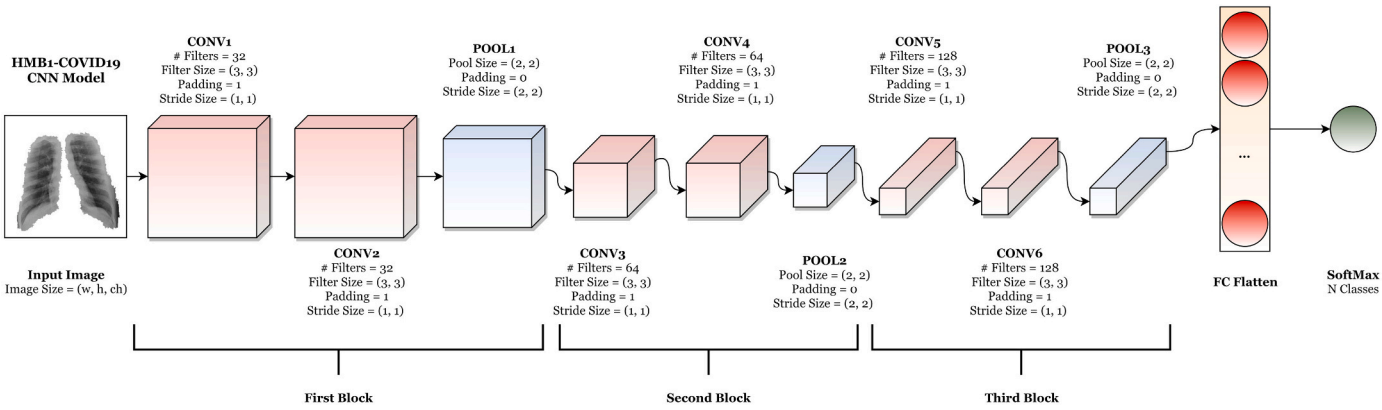


Fig. 6. Graphical representation of the proposed CNN architecture (HMB1-COVID19).

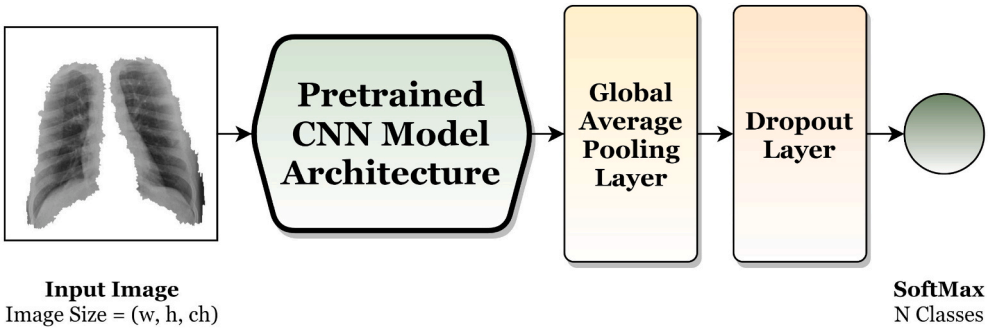


Fig. 7. Graphical representation of the pre-trained CNN design.

- Proposes a designed CNN model with its internal structure.
- Describes the TF approach and the used pre-trained TF models.

It is worth mentioning that, the major purpose of tackling both approaches (i.e., abstract CNN and TF) is the environment complexity and hardware availability. The suggested abstract CNN architecture contains fewer parameters compared to the used pre-trained CNN models and hence it can be faster (i.e., less time), and consume less memory and processing. It can be suitable for low-hardware devices such as mobiles and tablets. However, the pre-trained CNN models are more complex and hence can produce higher performance results. They can be suitable for modern and high-hardware devices such as servers. Table 4 summarizes approximately the number of parameters for each used model where the input image size is set to (64,64,3) with four output classes. The table is sorted in ascending order concerning the number of parameters.

The working mechanism of CNNs and their cascaded layers, and the CNN different parameters (i.e., weights) optimization and initializer techniques are summarized in Appendix 1: convolutional neural networks.

3.6.1. *The proposed abstract CNN model*

The authors proposed an abstractly designed CNN architecture named HMB1-COVID19, after try-and-error trials between different designs. The suggested architecture passed through three levels of incrementing (i.e., doubling) the number of kernels to extract more features where each level has two convolutional layers followed by max-pooling and dropout layers. A design summary of the suggested architecture is shown in Table 5 and Fig. 6.

In Table 5, the columns represent the block number, the layer type, the number of convolutional filters, the size of the convolutional filter, the size of pooling kernel, the stride size, the padding value, and other information respectively. The empty cells represent non-applicable (NA) values.

The proposed model uses (3, 3) filter size, padding of 1, and stride of (1, 1) for all convolutional layers. It uses (2, 2) pooling size, padding of 0, and stride of (2, 2) for all max-pooling layers. The dropout layers have initial dropout ratios of 25%. L2 regularization with an initial value of 10^{-4} is applied inside the convolutional layers. The output dense layer has *N* classes (i.e., the number of used categories in the dataset, and it will be defined in the experiments section).

3.6.2. *Transfer learning (TF)*

The transfer learning (TF) approach can be used instead of building and learning the CNN from scratch. TF is a machine learning concept that targets the gained experience and knowledge while handling a related task [93]. This approach is employed in DL by using the stored pre-trained models as the starting initial points. This grants swift progress and performance improvement. A pre-trained model can be obtained commonly by (1) selecting a related task where there are relationships in the inputs and outputs or (2) choosing a pre-trained source model from the shared, public, and available models. The current study follows the second path [94].

There are many shared, public, and available pre-trained CNN models such as VGG16 [95], VGG19 [96], ResNet50 [97], MobileNetV2 [98], Xception, NASNet, Large NasNet [99], DenseNet201 [100], InceptionV3 [101], and InceptionResNetV2 [102]. They were pre-trained on different databases such as the ImageNet [103]. It is a very

large scale hierarchical image database (i.e., over 14 million images). It was designed for visual object recognition software research and CV tasks [104].

The pre-trained CNN models, that are used in the current study, use the ImageNet pre-trained weights to initiate them. Each pre-trained model consists of a set of internal layers and can be controlled (i.e., to update the layer weights or freeze them). By decreasing the number of trainable layers, the number of trainable parameters will decrease, but it may lead to a noticeable decrease in the performance [105].

The number of controlled internal layers to be updated is denoted as the TF learning ratio in the current study. A global average pooling layer, a dropout layer, and an FC layer are added after the pre-trained model's last layer as shown in Fig. 7. Similar to pooling layers, the global average pooling layers are used to reduce the dimensionality but in an extreme style where the output layer size is $(1 \times 1 \times ch_{in})$ where ch_{in} is the number of input channels.

3.7. Seventh phase: prediction and evaluation phase

After and through learning the models, it is required to evaluate their performances to judge if they can generalize and be used or not.

Accuracy, Precision, Recall (i.e., Sensitivity), F1-score, and AUC values are widely-used performance measures (i.e., metrics) for the learning and production phases. Accuracy is the most intuitive performance metric and is defined as the ratio of correctly predicted observations to the total number of observations. Hence, as a common practice, the best architecture is selected according to the highest accuracy [106].

Precision is defined as the ratio of correctly predicted positive observations of the total predicted positive observations. Recall (i.e., sensitivity) is the ratio of correctly predicted positive observations to the total number of observations in a specific category (i.e., class). It helps when false-negative observations are high. The F1-score is the weighted average of precision and recall [107].

Eqs. (11), (12), (13), and (14) demonstrate the used accuracy, precision, recall, and F1-score formulas respectively to evaluate the models' performances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN + \varepsilon} \tag{11}$$

$$\text{Precision} = \frac{TP}{TP + FP + \varepsilon} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN + \varepsilon} \tag{13}$$

$$\text{F1}_{\text{Score}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

where ε is a very small number that is added in the denominators to avoid the division by zero, and *TP*, *TN*, *FP*, and *FN* are the true positive, true negative, false positive, and false negative values respectively.

The more the accuracy, F1-score, recall, AUC, and precision, the better the DL learning model while the lower the loss, the better the DL learning model. To unify all of them to find the maximum values, the value of "1 / Loss" is computed instead of the "Loss" value. To integrate between them, the weighted sum (WS) method [108] is used. It multiplies each metric by a *w* weight and computes the sum of them as shown in Eq. (15).

$$\begin{aligned}\text{Metric}_{WS} = {} & w_1 \times \text{Accuracy} + w_2 \times \text{Precision} + \\ & w_3 \times \text{Recall} + w_4 \times \text{F1}_{\text{Score}} + w_5 \times \text{AUC} + \frac{w_6}{\text{Loss}}\end{aligned} \tag{15}$$

where w_1, w_2, w_3, w_4, w_5, and w_6 are the weights and their sum must be 1. As mentioned, the accuracy is the most intuitive performance metric [109]. Hence, its weight, w_1, is modified to be the highest among others. In the current study, the used values for w_1, w_2, w_3, w_4, w_5, and w_6 are set to 0.5, 0.1, 0.1, 0.1, 0.1, and 0.1 respectively when the Metric_{WS} is used.

It is worth mentioning that the weights can be equalized to be "100 / 6" for each and this depends mainly on the target indication. In other words, if the target indicator is required to be for a specific performance metric, the highest weight *w* can be for it. If the target indicator is required to be balanced between all of the performance metrics, equalized weights can be used. As seen, the first path is selected.

3.8. Eighth phase: output and export phase

This phase handles the storing process as follows (1) the training and learning history can be stored in different formats such as text files and Comma Separated Values (CSV) files, (2) the trained CNN model is exported to be used later (i.e., to be used in the production system or to be retrained again), and (3) different figures can be plotted such as the relation between the number of epochs and training accuracy.

3.9. Ninth phase: hybrid model deployment phase

Choosing between the different hyperparameters used in the training of CNN models such as parameters optimizers, parameters initializers, dropout ratios, batch sizes, and TF learning ratios, can lead to a non-deterministic polynomial-time hard (NP-hard) problem [110].

For example, 8 parameters optimizers, 6 parameters initializers, 60 dropout ratios, 3 batch sizes, and 20 TF learning ratios will lead to 172,800 combinations. If a single combination takes only 1 min to train (which is a very small time in our case), then 120 days are required to complete. This will be very hard and time-consuming to handle.

Determining the most suitable or optimal combination among the learning hyperparameters' different combinations to achieve the best performance is considered a hyperparameters optimization problem. It can be handled using soft computing algorithms such as genetic algorithms (GAs) [111].

Genetic Algorithm (GA), in the current study, is used to solve the problem of finding the best combinations faster than the grid search or native searching approaches [112,113]. The working mechanism of the genetic algorithms is summarized in Appendix 2: genetic algorithms (GAs).

Hence, the current subsection discussion can be summarized in two points:

- Proposes a deep learning and genetic algorithms optimization approach (HMB-DLGA) for the parameters and hyperparameters learning and optimization.
- Describes the hybrid hierarchy used in the proposed hybrid COVID-19 framework (HMB-HCF).

3.9.1. Deep learning and genetic algorithms optimization approach (HMB-DLGA)

The authors used the CNNs to extract the features and optimize the learning parameters and used the GA to optimize the learning hyper-parameters and select the best combination. Algorithm 2 presents the proposed learning and optimization approach.

Algorithm 2. Deep learning and genetic algorithms optimization approach (HMB-DLGA) pseudocode.

Input: $N_p, N_s, N_c, S_r, model$ // Population Size, Number of Iterations, Number of the Required Combinations, Dataset Split Ratio, CNN Model Name/Type (Abstract or Pre-trained)
Output: *combinations* // The Best N_c Combinations
Data: X, Y // Images Dataset, Corresponding Images Labels
Set O_s = [Adam, NAdam, AdaDelta, AdaGrad, AdaMax, SGD, FTRL, RMSProp] // The Deep Learning Parameters Optimizers
Set W_s = [He Normal, He Uniform, Glorot Normal, Glorot Uniform, LeCun Normal, LeCun Uniform, Random Normal, Random Uniform, Truncated Normal] // The Deep Learning Parameters Initializers
Set $B_s = [32, 64]$ // The Used Batch Sizes
Set $L_s = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]\%$ // The Tranfer Learning Model Learning Ratios
Set $D_s = [0, 1, 2, ..., 60]\%$ // The Used Dropout Ratios
Set H_s = [ReLU, Tanh, ELU, SELU, Exponential, Sigmoid] // The Hidden Activation Functions
Set R_s = [L1(10^{-2}), L1(10^{-3}), L1(10^{-4}), L1(10^{-5}), L2(10^{-2}), L2(10^{-3}), L2(10^{-4}), L2(10^{-5})] // The Regularizers
Set M_s = [Accuracy, Precision, Recall, F1, AUC, Loss] // The Required Performance Metrics
$population = \text{GetInitialPopulation}(N_p, O_s, W_s, B_s, L_s, D_s, H_s, R_s)$ // Get the Initial N_p Population
$trainX, validationX, testX, trainY, validationY, testY = \text{SplitDataset}(X, Y, S_r)$ // Applying Dataset Split using the S_r Ratio to Get: Training Images Subset, Validation Images Subset, Testing Images Subset, Training Images Labels, Validation Images Labels, Testing Images Labels
Set $n_s = 1$ // Initialize the Iteration Counter
// Find the Best Combinations using the Iterative Approach (Loops)
while $n_s \leq N_s$ **do**
    Set *chromosomesWithScores* = [] // Initialize the Chromosomes with Scores as an Empty List
    // Find the Population Fitness Score Values of the Current Population using the Iterative Approach (Loops)
    Set $n_p = 1$ // Initialize the Population Counter
    while $n_p \leq N_p$ **do**
        Set $chromosome = population[n_p]$ // Get the Chromosome at Index n_p
        $trainedModel = \text{TrainValidateCNN}(model, chromosome, trainX, trainY, validationX, validationY, M_s)$ // Train and Validate the CNN Model on the Training and Validation Subsets
        $metrics = \text{TestCNN}(trainedModel, testX, testY, M_s)$ // Test the CNN Model on the Testing Subset and Find the Required Performance Metrics (M_s)
        $fitnessScore = \text{GetFitnessScore}(metrics)$ // Find the Fitness Score Value of the Current Chromosome
        $chromosomesWithScores.append((chromosome, fitnessScore))$ // Append the Chromosome with its corresponding Fitness Score Value in the *chromosomesWithScores* List
        $\text{ExportLogCNN}(trainedModel, model, chromosome, fitnessScore, n_s, n_p)$ // Export the Trained CNN Model with the Corresponding Chromosome and Fitness Score, Log the Learning History, and Plot the Required Figures
        Set $n_p = n_p + 1$ // Increment the Population Counter
    $sortedChromosomes = \text{SortChromosomes}(chromosomesWithScores)$ // Sort the Chromosomes according to the Fitness Score Values
    $newPopulation = \text{ApplySelectionCrossoverMutation}(sortedChromosomes, N_p, O_s, W_s, B_s, L_s, D_s, H_s, R_s)$ // Apply Selection, Crossover, and Mutation to Get the New Population
    Set $population = newPopulation$ // Set the New Population to the Current Population
    Set $n_s = n_s + 1$ // Increment the Iteration Counter
Set $combinations = \text{ExtractTop}(sortedChromosomes, N_c)$ // Extract the Top N_c Chromosomes

It is worth mentioning that the current subsection's following paragraphs present the details of the suggested algorithm and the detailed technical information can be retrieved from the appendices and used references.

3.9.1.1. Algorithm inputs and outputs. Algorithm 2 accepts five inputs (1) N_p: the population size (i.e., the number of chromosomes in the population), (2) N_s: the number of GA iterations, (3) N_c: the number of required combinations to be returned upon completion, (4) S_r: the dataset split ratio that will be applied to the dataset (X images and Y labels) to get the training, testing, and validation subsets, and (5) *model*: the required CNN model to learn and optimize (i.e., it can be HMB1-COVID19 or any pre-trained CNN model in the current study). It returns the best N_c combinations among all of them.

3.9.1.2. Algorithm line 1 to algorithm line 7. The algorithm starts by setting the available ranges for parameters optimizers, parameters initializers, batch sizes, TF model learning ratios, dropout ratios, hidden activation functions, and regularizers respectively.

The current study uses Adam, Nadam, AdaDelta, AdaGrad, AdaMax, SGD, Ftrl, and RMSProp as the DL parameters optimizers. It uses He Normal, He Uniform, Glorot Normal, Glorot Uniform, LeCun Normal, LeCun Uniform, Random Normal, Random Uniform, and Truncated Normal as the parameters' initializers for HMB1-COVID19 and ImageNet for the pre-trained CNN models. The TF model learning ratio is

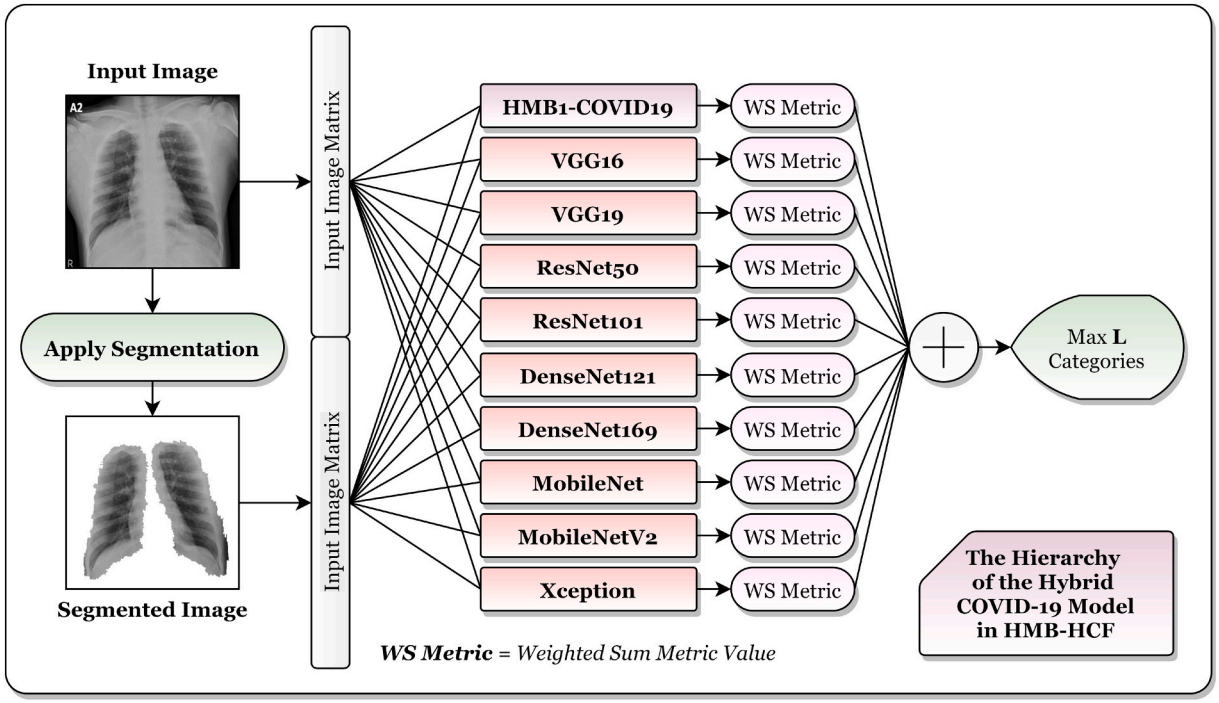


Fig. 8. The hierarchy of the hybrid COVID-19 model.

Table 6
Experiments common configurations summarization.

Configuration	Values
Dataset (X and Y)	Collected from 8 Resources
Categories	"Normal", "Pneumonia-Viral", "Pneumonia-Bacterial", and "COVID-19"
Dataset Size	13,711 (i.e., each category is 5439, 3631, 3001, and 1640 respectively)
Input Image Size	(64, 64, 3)
Abstract Models	HMB1-COVID19 (CNN Model)
Pre-trained Models	VGG16, VGG19, ResNet50, ResNet101, DenseNet121, DenseNet169, MobileNet, MobileNetV2, and Xception
DL Parameters Initializers W_s	He Normal, He Uniform, Glorot Normal, Glorot Uniform, LeCun Normal, LeCun Uniform, Random Normal, Random Uniform, and Truncated Normal
Pre-trained DL Parameters Initializers	ImageNet
Parameters Optimizers O_s	Adam, NAdam, AdaGrad, AdaDelta, AdaMax, RMSProp, Ftrl, and SGD
Hidden Activation Functions H_s	ReLU, Tanh, ELU, SELU, Exponential, and Sigmoid
Regularizers R_s	L1(10^{-2}), L1(10^{-3}), L1(10^{-4}), L1(10^{-5}), L2(10^{-2}), L2 (10^{-3}), L2(10^{-4}), and L2(10^{-5})
Output Activation Function	SoftMax
TF Learn Ratios L_s	[0, 5, 10, …, 95, 100]%
Batch Sizes B_s	32 and 64
Dropout Ratios D_s	[1, 2, 3, …, 95, 60]%
Number of Epochs	64
Performance Metrics M_s	Accuracy, Loss, Precision, F1-score, AUC, and Recall
Weighted Sum WS Ratios	Accuracy (0.5), Loss (0.1), Precision (0.1), F1-score (0.1), AUC (0.1), and Recall (0.1)
Number of GA Iterations N_s	15
Population Size N_p	10
GA Mutation Rate	0.25 (i.e., 25%)
Split Ratio S_r	90% to 10% (Fig. 5)
Data Augmentation	Yes (Table 3)
Training Environments	Google Colab and Toshiba Qosmio X70-A

ranged from 0% to 100% with a step of 5% and hence there are 21 values. The dropout ratio is ranged from 0% to 60% with a step of 1% and hence there are 61 values. The batch size is 32 or 64. ReLU, Tanh, ELU, SELU, Exponential, and Sigmoid are the used hidden activation functions. L1(10^{-2}), L1(10^{-3}), L1(10^{-4}), L1(10^{-5}), L2(10^{-2}), L2(10^{-3}), L2(10^{-4}), and L2(10^{-5}) are the used regularizers.

3.9.1.3. Algorithm line 8. After that, it defines the required performance metrics.

3.9.1.4. Algorithm line 9. All population chromosomes are initiated randomly concerning the available ranges (i.e., O_s: DL parameters optimizers, W_s: DL parameters initializers, H_s: hidden activation functions, D_s: dropout ratios, B_s: batch sizes, L_s: TF model learning ratios, and R_s: regularizers). They, the *population*, are constructed as a matrix where each row of the N_p rows is a solution (i.e., chromosome).

Each chromosome's genes: (1) the first gene is a randomly selected DL parameters' optimizer from O_s, (2) the second gene is a randomly selected batch size from B_s, (3) the third gene is a randomly selected dropout ratio from D_s, (4) the fourth gene is a randomly selected TF model learning ratio from L_s, (5) the fifth gene is a randomly selected DL parameters' initializer from W_s, (6) the sixth gene is a randomly selected hidden activation function from H_s, and (7) the seventh gene is a randomly selected regularizer from R_s.

If the model is a pre-trained CNN, the first 4 genes are used and if the model is the HMB1-COVID19 CNN model, all of them unless the fourth gene are used (i.e., the fourth gene is related to transfer learning). The reason behind this is that the pre-trained CNN model does not need parameters' initializers, regularizers, nor hidden activation functions as it depends initially on the ImageNet weights. For example, "[AdaDelta,32,50,80]". This example shows a randomly initialized chromosome for a pre-trained CNN model.

3.9.1.5. Algorithm line 10. The "X" and "Y" represent the images and the corresponding labels respectively. The "SplitDataset" function is responsible for splitting the whole dataset into train, test, and validation subsets (discussed in "Fifth Phase: Splitting Phase") using the split ratio

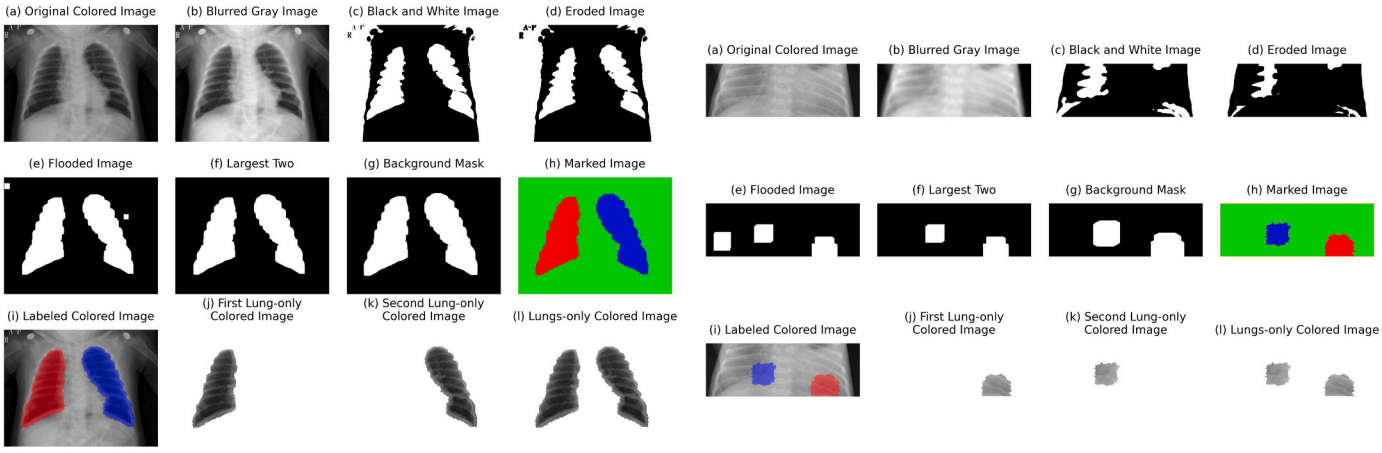


Fig. 9. Two samples with the segmentation steps. Left: a successful segmentation process. Right: a failed segmentation process.

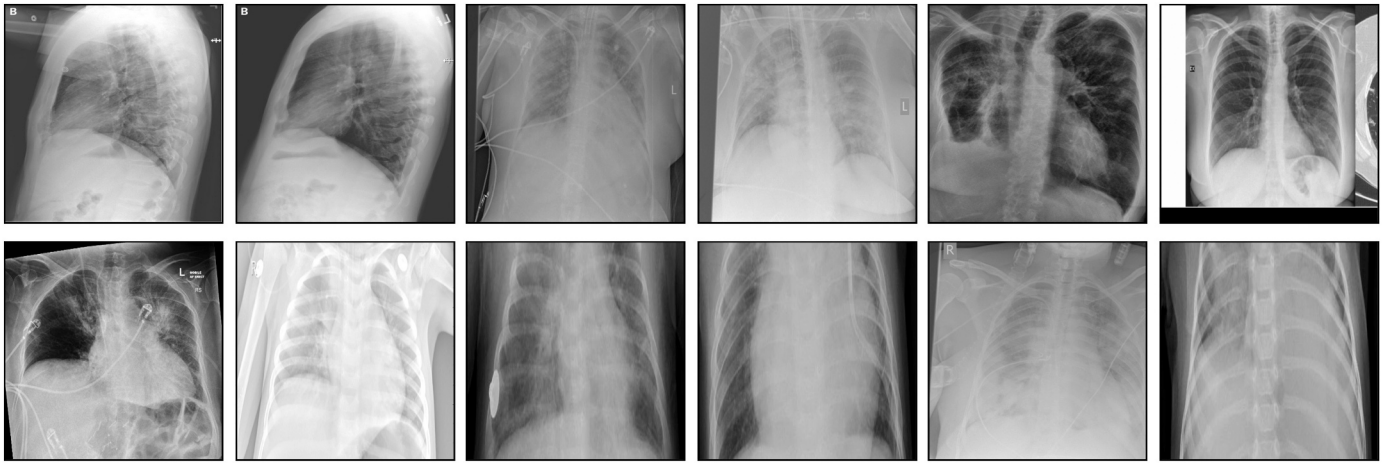


Fig. 10. Samples of the defected X-ray lungs images.

Table 7
EXP-SEG-1: first HMB-HCF learning and optimization experiment with segmentation and VGG16.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	SGD	32	0.4	15%	0.4120	83.91%	0.8367	86.29%	81.27%	0.9700	77.02%
2	SGD	32	0.3	30%	0.2210	90.67%	0.9085	91.62%	90.12%	0.9905	82.95%
3	SGD	32	0.1	40%	0.1305	95.55%	0.9552	95.57%	95.47%	0.9946	87.14%
4	SGD	32	0.0	40%	0.1099	96.14%	0.9615	96.34%	95.97%	0.9965	87.79%
5	SGD	32	0.0	40%	0.0669	97.86%	0.9772	97.75%	97.69%	0.9984	89.72%
6	SGD	32	0.1	40%	0.0732	97.69%	0.9780	97.88%	97.72%	0.9981	89.53%
7	SGD	32	0.4	75%	0.0513	98.18%	0.9827	98.35%	98.20%	0.9989	90.51%
8	SGD	32	0.1	90%	0.0563	98.34%	0.9836	98.36%	98.36%	0.9978	90.43%
9	SGD	32	0.4	90%	0.0185	99.62%	0.9963	99.63%	99.63%	0.9996	95.11%
10	SGD	32	0.4	90%	0.0464	98.55%	0.9857	98.57%	98.57%	0.9987	90.99%
11	SGD	32	0.4	75%	0.0269	99.03%	0.9905	99.05%	99.05%	0.9992	92.94%
12	SGD	32	0.4	90%	0.0174	99.36%	0.9936	99.36%	99.36%	0.9996	95.22%
13	SGD	32	0.4	90%	0.0176	99.57%	0.9958	99.58%	99.58%	0.9996	95.33%
14	SGD	32	0.4	90%	0.0245	99.30%	0.9934	99.36%	99.31%	0.9992	93.52%
15	SGD	32	0.1	90%	0.0372	99.30%	0.9931	99.31%	99.31%	0.9981	92.12%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

S_r. The training and validation subsets are used to learn and train the selected CNN model. The testing subset is used to test the performance of the trained CNN model after the learning process.

3.9.1.6. Algorithm line 12. An outer loop is executed and **Algorithm Line 13** to **Algorithm Line 26** are repeated for the given number of iterations N_s. In other words, the hyperparameters' optimization process

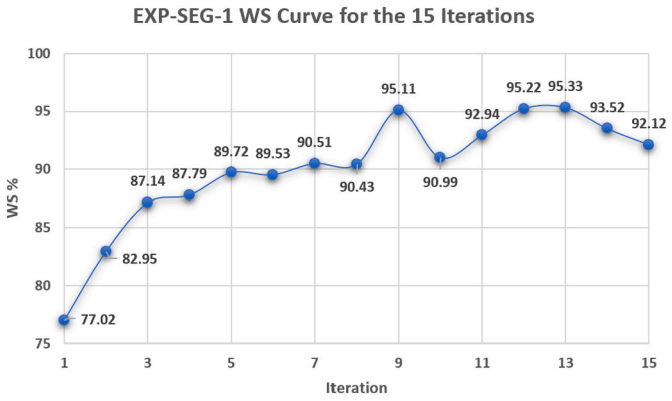


Fig. 11. EXP-SEG-1 WS curve for the 15 iterations.

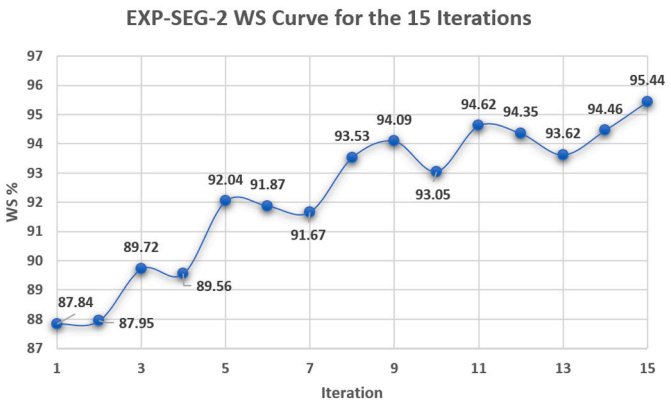


Fig. 12. EXP-SEG-2 WS curve for the 15 iterations.

using GA is applied for N_s times.

3.9.1.7. Algorithm line 15 to algorithm line 22. An inner loop is executed and **Algorithm Line 16** to **Algorithm Line 22** are repeated for the given population size N_p. This loop iterates on each chromosome to (1) train and validate the selected CNN model using the current chromosome on the train and validation subsets (in **Algorithm Line 17**), (2) test the trained CNN model on the test subset and report the required performance metrics stated in M_s and store them in the *metrics* variable (in **Algorithm Line 18**), (3) compute the fitness function to determine the corresponding score, *fitnessScore*, using the WS method defined in Eq. (15) (in **Algorithm Line 19**), (4) append in the chromosome with the corresponding fitness score in a list to be used after the completion of the inner loop (in **Algorithm Line 20**), and (5) export the trained CNN model to be used later in the production, the training history can be logged, and different figures and relations can be plotted and stored (in **Algorithm Line 21**). This learning process is applied to all of the N_p solutions and their corresponding fitness scores are stored.

3.9.1.8. Algorithm line 23. After that, they are sorted in descending order concerning the fitness scores.

3.9.1.9. Algorithm line 24. Selection, crossover, and mutation are applied to the sorted chromosomes. Each mutation is applied randomly if it is above or equal to 0.5 (i.e., a fractional number from 0 to 1 is generated randomly and if it is more than or equal to 0.5, the mutation is applied). The used mutation rate is set to 25% which means that only a single gene is selected randomly from the available options.

3.9.1.10. Algorithm line 27. Finally, the top N_c chromosomes concerning the fitness scores are extracted from the final sorted chromosomes and returned as outputs.

3.9.2. The hierarchy of the hybrid COVID-19 model

The authors used the best combinations of each trained CNN model and cascaded them as shown in Fig. 8.

The input image (i.e., X-ray lung image) is segmented and both of them are concatenated and passed to all of the trained CNN models including the suggested HMB1-COVID19 and other pre-trained models. These models are those reported from the previously discussed phases inside the proposed framework. The WS scores are summed together and the maximum L categories are reported. This will report more accurate results and decisions compared with the usage of only a single trained CNN model as shown in the experimental results in the next section.

It is worth mentioning that the suggested hybrid model can be partitioned. In other words, one or more models can be turned OFF and the remaining ones will handle the rest of the work.

The final prediction is based on the majority voting principle similar

Table 8
EXP-SEG-2: second HMB-HCF learning and optimization experiment with segmentation and VGG19.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	SGD	64	0.2	50%	0.1107	96.14%	0.9630	96.58%	96.03%	0.9973	87.84%
2	AdaGrad	32	0.3	85%	0.1151	96.41%	0.9642	96.55%	96.29%	0.9954	87.95%
3	SGD	64	0.3	85%	0.0670	97.80%	0.9783	97.88%	97.78%	0.9979	89.72%
4	SGD	64	0.3	85%	0.0717	97.75%	0.9769	97.75%	97.64%	0.9983	89.56%
5	SGD	32	0.3	85%	0.0340	98.87%	0.9891	98.94%	98.89%	0.9991	92.04%
6	SGD	32	0.3	85%	0.0378	99.09%	0.9896	98.96%	98.96%	0.9987	91.87%
7	SGD	64	0.2	85%	0.0376	98.77%	0.9878	98.78%	98.78%	0.9991	91.67%
8	SGD	64	0.2	85%	0.0243	99.25%	0.9931	99.36%	99.26%	0.9992	93.53%
9	SGD	32	0.3	85%	0.0213	99.25%	0.9926	99.26%	99.26%	0.9996	94.09%
10	SGD	32	0.3	85%	0.0264	99.09%	0.9910	99.10%	99.10%	0.9992	93.05%
11	SGD	32	0.4	85%	0.0195	99.36%	0.9936	99.36%	99.36%	0.9996	94.62%
12	SGD	32	0.4	85%	0.0213	99.57%	0.9958	99.58%	99.58%	0.9989	94.35%
13	SGD	32	0.3	85%	0.0235	99.20%	0.9921	99.21%	99.21%	0.9995	93.62%
14	SGD	64	0.2	85%	0.0194	99.14%	0.9915	99.15%	99.15%	0.9996	94.46%
15	SGD	32	0.2	85%	0.0170	99.46%	0.9947	99.47%	99.47%	0.9996	95.44%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 9
EXP-SEG-3: third HMB-HCF learning and optimization experiment with segmentation and ResNet50.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaMax	64	0.1	20%	7.1878	26.33%	0.2614	26.14%	26.14%	0.5535	26.55%
2	AdaMax	64	0.1	20%	2.7166	26.27%	0.2661	27.06%	26.20%	0.5492	26.65%
3	AdaMax	64	0.1	20%	2.6082	26.92%	0.2611	27.46%	24.92%	0.5613	26.96%
4	AdaMax	64	0.2	10%	2.0592	29.54%	0.2932	29.99%	28.69%	0.5513	29.13%
5	AdaGrad	64	0.4	30%	3.5762	26.33%	0.2614	26.14%	26.14%	0.5428	26.46%
6	AdaMax	64	0.2	10%	2.3876	31.58%	0.3120	32.23%	30.26%	0.5429	30.63%
7	Adam	64	0.4	30%	2.9762	26.33%	0.2614	26.14%	26.14%	0.5390	26.43%
8	AdaMax	64	0.1	20%	3.6475	26.11%	0.2620	26.20%	26.20%	0.5534	26.48%
9	AdaGrad	64	0.4	30%	3.9086	30.03%	0.2979	29.85%	29.72%	0.5260	29.24%
10	AdaMax	64	0.1	20%	3.5242	26.06%	0.2617	26.19%	26.15%	0.5537	26.45%
11	AdaMax	64	0.4	50%	2.0866	29.28%	0.2369	32.35%	18.89%	0.5247	27.43%
12	Nadam	64	0.3	30%	5.0350	26.33%	0.2614	26.14%	26.14%	0.5427	26.45%
13	Nadam	64	0.1	20%	3.5695	26.32%	0.2614	26.14%	26.14%	0.5570	26.60%
14	AdaMax	64	0.1	20%	2.9179	30.24%	0.3047	30.59%	30.36%	0.5499	29.80%
15	AdaMax	32	0.1	20%	4.0317	28.10%	0.2781	27.84%	27.78%	0.5411	27.83%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

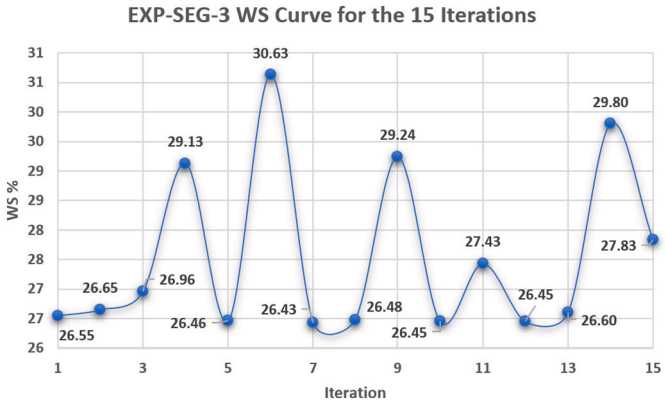


Fig. 13. EXP-SEG-3 WS curve for the 15 iterations.

to Eq. (1). In other words, the most repeated (i.e., common) decision among the models for a single input will be the final decision. It is worth mentioning that the user can retrieve the most common L decisions (i.e., categories).

4. Experimental results and discussion

The performed and reported experiments can be divided into four categories:

- Experiments related to the segmentation process.
- Experiments related to the proposed framework with segmentation.
- Experiments related to the proposed framework without segmentation.
- Experiments Summarization, Generalization Validation, and Comparisons.

In the first three experimental categories, the used dataset is unified and collected from 8 sources. They are "COVID-19 Radiography Database" [114], "Pneumonia (virus) vs COVID-19" Dataset [50], "Covid-19 Xray images using CNN" Dataset [51], "COVID-19 X-ray Images5" Dataset [52], "COVID-19 Patients Lungs X Ray Images 10,000" Dataset

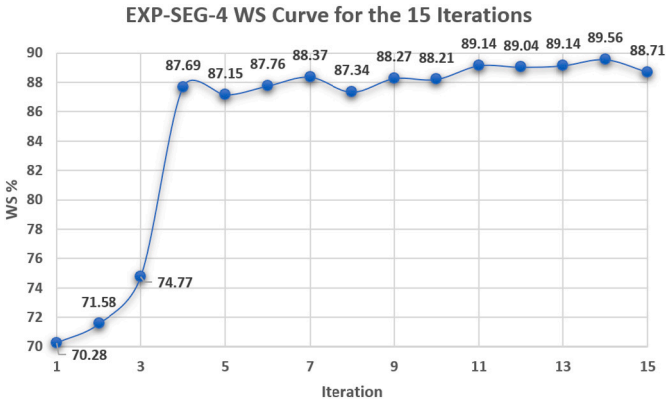


Fig. 14. EXP-SEG-4 WS curve for the 15 iterations.

Table 10
EXP-SEG-4: fourth HMB-HCF learning and optimization experiment with segmentation and ResNet101.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaDelta	32	0.2	100%	0.8164	75.71%	0.7619	79.49%	73.25%	0.9409	70.28%
2	AdaDelta	32	0.2	100%	1.2505	77.64%	0.7724	79.71%	74.96%	0.9493	71.58%
3	AdaDelta	32	0.2	100%	1.8553	81.45%	0.8123	82.41%	80.10%	0.9614	74.77%
4	SGD	32	0.2	100%	0.1355	96.25%	0.9626	96.29%	96.24%	0.9946	87.69%
5	SGD	32	0.2	100%	0.1588	95.71%	0.9575	95.91%	95.60%	0.9936	87.15%
6	SGD	32	0.2	100%	0.1145	96.14%	0.9618	96.24%	96.13%	0.9963	87.76%
7	SGD	32	0.1	100%	0.0996	96.73%	0.9680	96.82%	96.77%	0.9961	88.37%
8	SGD	64	0.2	100%	0.1275	95.71%	0.9584	95.91%	95.76%	0.9951	87.34%
9	SGD	64	0.2	100%	0.1143	96.78%	0.9684	96.92%	96.77%	0.9954	88.27%
10	SGD	32	0.2	100%	0.1118	96.73%	0.9666	96.73%	96.58%	0.9956	88.21%
11	AdaGrad	32	0.0	100%	0.0826	97.43%	0.9746	97.46%	97.46%	0.9977	89.14%
12	SGD	64	0.0	100%	0.0830	97.32%	0.9732	97.35%	97.30%	0.9974	89.04%
13	AdaGrad	32	0.0	100%	0.0817	97.43%	0.9743	97.51%	97.35%	0.9971	89.14%
14	SGD	32	0.0	100%	0.0663	97.59%	0.9759	97.66%	97.51%	0.9985	89.56%
15	AdaGrad	32	0.0	100%	0.0936	97.10%	0.9711	97.14%	97.09%	0.9957	88.71%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 11
EXP-SEG-5: fifth HMB-HCF learning and optimization experiment with segmentation and Xception.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaMax	32	0.3	15%	4.1383	49.22%	0.4936	49.59%	49.14%	0.7227	46.67%
2	Nadam	32	0.2	65%	1.8644	54.21%	0.5389	54.74%	53.09%	0.7933	51.26%
3	AdaMax	64	0.3	90%	1.7366	57.53%	0.5739	57.71%	57.09%	0.8351	54.39%
4	AdaMax	64	0.3	90%	1.8229	61.88%	0.6180	62.05%	61.56%	0.8398	57.93%
5	Nadam	32	0.0	90%	1.0670	69.06%	0.6930	69.90%	68.72%	0.8875	64.29%
6	Nadam	64	0.3	70%	1.7254	61.13%	0.6046	61.41%	59.58%	0.8254	57.02%
7	Nadam	64	0.3	70%	1.6139	61.61%	0.6090	61.76%	60.10%	0.8372	57.51%
8	AdaMax	32	0.3	90%	1.0881	66.70%	0.6670	67.03%	66.38%	0.8938	62.39%
9	AdaMax	64	0.3	90%	1.4792	59.41%	0.5934	60.17%	58.55%	0.8376	55.95%
10	AdaMax	32	0.3	90%	1.5132	66.11%	0.6608	66.45%	65.72%	0.8670	61.62%
11	AdaGrad	64	0.3	100%	0.0524	98.55%	0.9860	98.62%	98.57%	0.9988	90.74%
12	AdaGrad	64	0.3	100%	0.0552	98.07%	0.9806	98.20%	97.93%	0.9985	90.25%
13	AdaGrad	64	0.3	100%	0.0451	98.34%	0.9833	98.36%	98.31%	0.9990	90.88%
14	AdaMax	32	0.5	100%	0.0386	98.93%	0.9897	98.99%	98.94%	0.9986	91.73%
15	AdaMax	32	0.3	100%	0.0311	99.25%	0.9926	99.26%	99.26%	0.9989	92.61%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

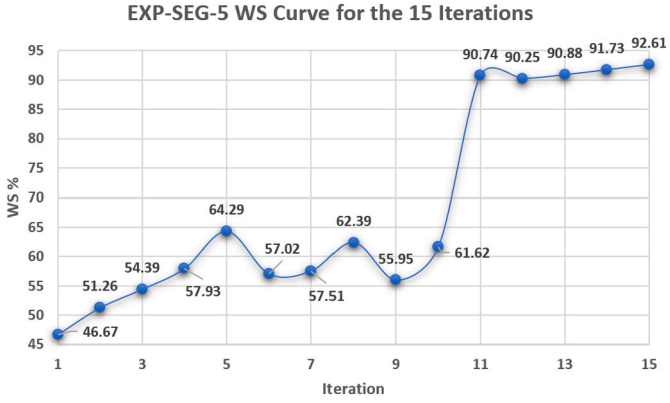


Fig. 15. EXP-SEG-5 WS curve for the 15 iterations.

[53], "COVID-19 Chest X Rays" Dataset [54], "COVID-19 Dataset" [55], and "Curated Chest X-Ray Image Dataset for COVID-19" [56]. The datasets details are presented in "First Phase: Images Acquisition Phase".

The collected images are combined by category together into 4 categories ("COVID-19", "Normal", "Pneumonia-Bacterial", and "Pneumonia-Viral"). The uncategorized and repeated images are removed. The total number of images is 13,711 and in each category is "1640", "5439", "3001", and "3631" respectively.

The following subsections discuss the different experiments with their reported results. However, Table 6 summarizes the common experiments configurations.

4.1. Segmentation experiments using HMB-LSAXI

As discussed in the hybrid hierarchy (Fig. 8), the input image is applied in two formats, segmented and non-segmented. The segmentation process used the proposed HMB-LSAXI algorithm.

By applying the segmentation on the overall dataset (i.e., the 13,711 images), only 7308 are segmented successfully. The computed

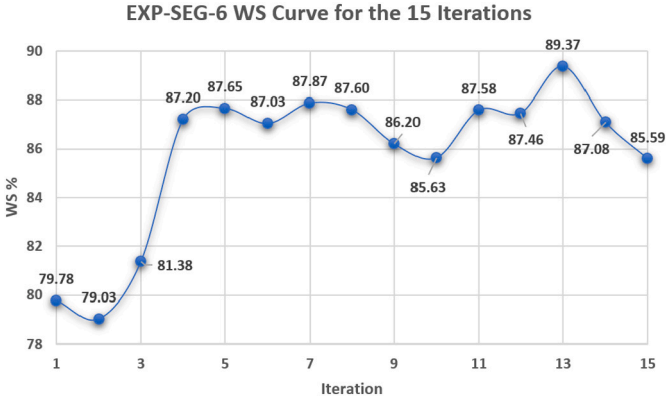


Fig. 16. EXP-SEG-6 WS curve for the 15 iterations.

Table 12
EXP-SEG-6: sixth HMB-HCF learning and optimization experiment with segmentation and DenseNet121.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	RMSProp	32	0.1	100%	0.5317	87.45%	0.8732	87.54%	87.12%	0.9670	79.78%
2	RMSProp	32	0.1	100%	0.7389	86.64%	0.8660	86.72%	86.49%	0.9594	79.03%
3	AdaMax	32	0.1	100%	0.4670	89.17%	0.8945	89.71%	89.19%	0.9743	81.38%
4	RMSProp	32	0.1	100%	0.1369	95.66%	0.9564	95.77%	95.52%	0.9952	87.20%
5	AdaMax	32	0.2	100%	0.1374	96.19%	0.9629	96.34%	96.24%	0.9936	87.65%
6	AdaMax	32	0.1	100%	0.1623	95.55%	0.9567	95.80%	95.55%	0.9938	87.03%
7	AdaMax	32	0.1	100%	0.1259	96.41%	0.9645	96.50%	96.40%	0.9942	87.87%
8	AdaMax	32	0.2	100%	0.1370	96.14%	0.9616	96.18%	96.13%	0.9948	87.60%
9	AdaMax	32	0.1	100%	0.2274	94.85%	0.9477	94.82%	94.73%	0.9904	86.20%
10	SGD	32	0.2	100%	0.2171	94.05%	0.9419	94.31%	94.07%	0.9890	85.63%
11	SGD	32	0.0	100%	0.1386	96.14%	0.9619	96.19%	96.19%	0.9930	87.58%
12	SGD	64	0.0	100%	0.1546	96.14%	0.9605	96.15%	95.95%	0.9933	87.46%
13	AdaMax	32	0.1	100%	0.0814	97.69%	0.9772	97.72%	97.72%	0.9974	89.37%
14	AdaMax	32	0.0	100%	0.1996	95.81%	0.9587	95.87%	95.87%	0.9913	87.08%
15	SGD	64	0.1	100%	0.1963	93.99%	0.9388	93.93%	93.83%	0.9922	85.59%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 13
EXP-SEG-7: seventh HMB-HCF learning and optimization experiment with segmentation and DenseNet169.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaGrad	64	0.3	85%	3.1203	38.55%	0.3865	39.08%	38.25%	0.6681	37.59%
2	AdaGrad	64	0.3	85%	4.0409	37.59%	0.3768	38.02%	37.35%	0.6451	36.57%
3	AdaGrad	32	0.3	85%	3.5347	54.74%	0.5463	55.11%	54.17%	0.7604	51.40%
4	AdaGrad	32	0.3	85%	2.3902	54.37%	0.5427	54.92%	53.64%	0.7810	51.32%
5	AdaGrad	32	0.3	85%	2.6333	59.52%	0.5927	59.84%	58.73%	0.7947	55.53%
6	AdaGrad	32	0.3	85%	3.0503	57.16%	0.5702	57.70%	56.40%	0.7754	53.47%
7	AdaGrad	32	0.3	85%	2.5901	55.82%	0.5592	56.45%	55.42%	0.7824	52.55%
8	AdaGrad	64	0.3	100%	0.4106	88.79%	0.8876	89.05%	88.48%	0.9751	81.02%
9	AdaGrad	64	0.3	100%	0.3227	90.94%	0.9093	91.01%	90.86%	0.9839	82.90%
10	AdaGrad	64	0.3	100%	0.2460	92.87%	0.9286	93.01%	92.71%	0.9886	84.59%
11	AdaGrad	64	0.3	100%	0.2439	92.23%	0.9218	92.45%	91.92%	0.9886	84.06%
12	Adam	64	0.3	100%	0.1236	95.87%	0.9576	95.78%	95.73%	0.9955	87.43%
13	AdaGrad	64	0.3	100%	0.3227	91.53%	0.9152	91.77%	91.28%	0.9818	83.35%
14	SGD	64	0.3	100%	0.3227	91.90%	0.9207	92.20%	91.95%	0.9818	83.70%
15	Adam	64	0.3	100%	0.2161	93.67%	0.9360	93.80%	93.40%	0.9892	85.27%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

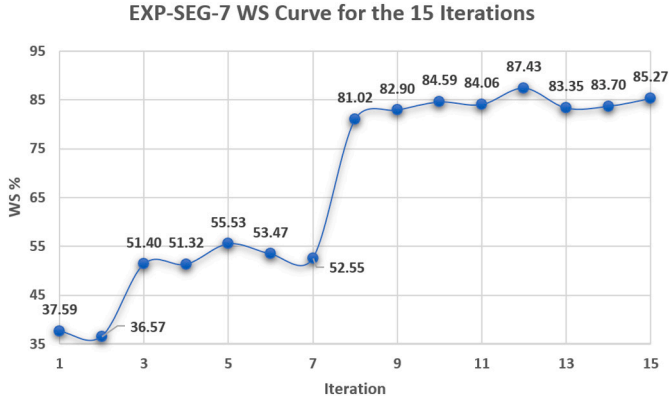


Fig. 17. EXP-SEG-7 WS curve for the 15 iterations.

segmentation success ratio is 53.30%. After tracing the defective images to determine the reasons for this low ratio, we analyzed the major reasons behind that.

They are (1) some images are sided X-Ray images, (2) some images are not clear (i.e., the lungs do not appear in them), (3) some images were previously manipulated, and (4) some images contain part of the lungs. Fig. 9 shows two X-Ray lungs images samples with the segmentation steps where the left represents a successful segmentation process and the right represents a failed segmentation process. Fig. 10 shows samples of the defected X-Ray lungs images.

It is worth mentioning that (1) for the experiments with segmentation, the defected images are removed from the dataset, and (2) for the experiments without segmentation, the whole dataset including the defected ones are used to validate if the classifier and optimizer can overcome the defection issue.

4.2. HMB-HCF learning and optimization experiments with segmentation

HMB1-COVID19 and the pre-trained CNN models are subjected to be optimized where each one at a time. This is applied to get the best combinations from each of them to be used later in the hybrid architecture. To distinguish between the experiments, the annotation "EXP-

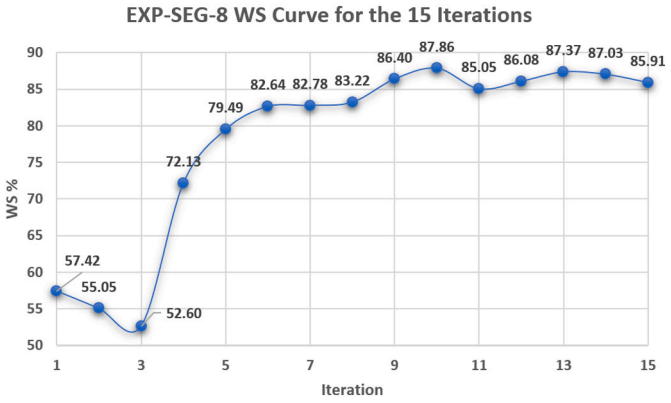


Fig. 18. EXP-SEG-8 WS curve for the 15 iterations.

Table 14
EXP-SEG-8: eigth HMB-HCF learning and optimization experiment with segmentation and mobileNet.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	RMSProp	64	0.2	35%	4.8663	61.77%	0.6162	61.77%	61.48%	0.8031	57.42%
2	RMSProp	64	0.2	35%	6.0402	59.36%	0.5889	58.93%	58.85%	0.7691	55.05%
3	RMSProp	32	0.2	35%	7.7115	56.51%	0.5634	56.41%	56.26%	0.7426	52.60%
4	RMSProp	64	0.2	100%	1.2859	78.66%	0.7849	78.92%	78.09%	0.9175	72.13%
5	AdaGrad	64	0.2	100%	0.3296	86.65%	0.8684	87.95%	85.80%	0.9799	79.49%
6	RMSProp	64	0.2	100%	0.4666	87.13%	0.8711	87.38%	86.86%	0.9730	82.64%
7	AdaGrad	64	0.2	100%	0.2534	90.72%	0.9047	91.10%	89.88%	0.9875	82.78%
8	SGD	64	0.2	100%	0.2553	91.21%	0.9113	91.35%	90.91%	0.9882	83.22%
9	SGD	64	0.2	100%	0.1635	94.80%	0.9486	94.91%	94.81%	0.9936	86.40%
10	Nadam	64	0.2	100%	0.1047	96.14%	0.9621	96.29%	96.13%	0.9976	87.86%
11	SGD	64	0.2	100%	0.2249	93.35%	0.9343	93.48%	93.38%	0.9902	85.05%
12	SGD	64	0.2	100%	0.1765	94.42%	0.9457	94.64%	94.49%	0.9936	86.08%
13	SGD	64	0.2	100%	0.1262	95.82%	0.9570	95.73%	95.68%	0.9954	87.37%
14	SGD	64	0.2	100%	0.1376	95.50%	0.9536	95.41%	95.31%	0.9946	87.03%
15	AdaMax	32	0.3	100%	0.2108	94.53%	0.9432	94.42%	94.22%	0.9878	85.91%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 15
EXP-SEG-9: ninth HMB-HCF learning and optimization experiment with segmentation and MobileNetV2.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	Adam	64	0.2	80%	2.6106	56.78%	0.5683	57.12%	56.56%	0.7817	53.30%
2	AdaMax	64	0.3	100%	0.8695	83.32%	0.8320	83.34%	83.07%	0.9444	76.18%
3	AdaMax	64	0.3	100%	0.5846	86.76%	0.8671	86.79%	86.65%	0.9648	79.21%
4	SGD	64	0.3	100%	0.4353	88.69%	0.8863	88.73%	88.53%	0.9725	80.89%
5	AdaMax	32	0.3	100%	0.4721	88.69%	0.8857	88.65%	88.50%	0.9713	80.84%
6	AdaMax	32	0.3	100%	0.3570	89.76%	0.8969	89.90%	89.48%	0.9813	81.88%
7	SGD	64	0.3	100%	0.5495	87.88%	0.8824	88.45%	88.03%	0.9642	80.24%
8	AdaMax	32	0.5	100%	0.3869	90.24%	0.9012	90.34%	89.91%	0.9761	82.18%
9	AdaMax	64	0.3	100%	0.4300	89.60%	0.8961	89.73%	89.48%	0.9744	81.66%
10	AdaMax	64	0.4	100%	0.4171	90.24%	0.9017	90.49%	89.85%	0.9764	82.17%
11	AdaMax	32	0.0	100%	0.5140	86.43%	0.8645	86.66%	86.25%	0.9675	79.02%
12	SGD	64	0.3	100%	0.4265	90.08%	0.9046	90.73%	90.20%	0.9726	82.14%
13	AdaMax	32	0.0	100%	0.5896	88.31%	0.8835	88.47%	88.24%	0.9630	80.46%
14	SGD	64	0.3	100%	0.4610	89.71%	0.8968	89.78%	89.59%	0.9732	81.71%
15	SGD	64	0.2	100%	0.4998	86.70%	0.8641	86.58%	86.25%	0.9685	79.16%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

SEG-xx" is used to denote the HMB-HCF learning and optimization experiments with segmentation where "xx" is the experiment number.

Table 7 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the VGG16 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 7, it is clear that the SGD parameters optimizer with a batch size of 32 was the top-1 in all iterations. The 40% dropout ratio was the best in 8 iterations. The 90% TF learn ratio was the best in 7 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0174, 99.62%, 0.9963, 99.63%, 99.63%, 0.9996, and 95.33% respectively while the last reported iteration metrics were 0.0372, 99.30%, 0.9931, 99.31%, 99.31%, 0.9981, and 92.12% respectively. The WS and TF learn ratio had a high positive correlation (0.86) while the WS and dropout ratio had negligible correlation (0.17). The dropout ratio and TF learn ratio had a low positive correlation (0.38). Fig. 11 shows the EXP-SEG-1 WS curve for the 15 iterations.

Table 8 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the VGG19 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 8, the SGD optimizer was the best in 14 iterations. The 32 batch size was the best in 9 iterations. The 30% dropout ratio was the best in 8 iterations. The 85% TF learn ratio was the best in 14 iterations.

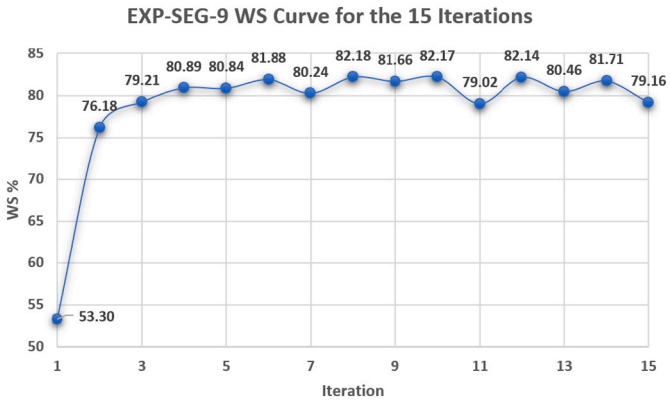


Fig. 19. EXP-SEG-9 WS curve for the 15 iterations.

The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0170, 99.57%, 0.9958, 99.58%, 99.58%, 0.9996, and 95.44% respectively while the last reported iteration metrics were 0.0170, 99.46%, 0.9947, 99.47%, 99.47%, 0.9996, and 95.44% respectively. The WS and TF learn ratio had a moderate positive correlation (0.50) while the WS and dropout ratio had negligible correlation (0.12). The dropout ratio and TF learn ratio had a low positive correlation (0.33). Fig. 12 shows the EXP-SEG-2 WS curve for the 15 iterations.

Table 9 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the ResNet50 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 9, the AdaMax optimizer was the best in 10 iterations. The 64 batch size was the best in 14 iterations. The 10% dropout ratio was the best in 8 iterations. The 20% TF learn ratio was the best in 8 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 2.0592, 31.58%, 0.3120, 32.35%, 30.36%, 0.5613, and 30.63% respectively while the last reported iteration metrics were 4.0317, 28.10%, 0.2781, 27.84%, 27.78%, 0.5411, and 27.83% respectively. The WS and TF learn ratio had a low negative correlation (−0.33) while the WS and dropout ratio had negligible correlation (0.03). The dropout ratio and TF learn ratio had a moderate positive correlation (0.69). Fig. 13 shows the EXP-SEG-3 WS curve for the 15 iterations.

Table 10 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the ResNet101 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 10, the SGD optimizer was the best in 9 iterations. The 32 batch size was the best in 12 iterations. The 20% dropout ratio was the best in 9 iterations. The 100% TF learn ratio was the best in 15 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0663, 97.59%, 0.9759, 97.66%, 97.51%, 0.9985, and 89.56% respectively while the last reported iteration metrics were 0.0936, 97.10%, 0.9711, 97.14%, 97.09%, 0.9957, and 88.71% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had a low negative correlation (−0.47). The dropout ratio and TF learn ratio could not be determined. Fig. 14 shows the EXP-SEG-4 WS curve for the 15 iterations.

Table 11 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the Xception pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

Table 16
EXP-SEG-10: tenth HMB-HCF learning and optimization experiment with segmentation and HMB1-COVID19.

Iteration	PO[a]	Batch size	DR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	RMSProp	32	0.47	ELU	Glorot normal	L1(10^{-5})	0.5693	81.79%	0.8146	83.31%	79.75%	0.9646	75.17%
2	RMSProp	32	0.47	ELU	Glorot normal	L1(10^{-5})	0.4683	86.88%	0.8668	87.88%	85.54%	0.9790	79.45%
3	AdaGrad	32	0.14	ReLU	Glorot normal	L1(10^{-4})	1.2240	79.04%	0.7889	81.63%	76.41%	0.9536	72.83%
4	SGD	64	0.17	ReLU	Glorot normal	L1(10^{-4})	2.2340	59.05%	0.5925	60.22%	58.34%	0.8273	55.62%
5	SGD	32	0.17	ReLU	Glorot normal	L2(10^{-2})	0.6212	81.89%	0.8166	83.80%	79.69%	0.9622	75.24%
6	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.6352	78.60%	0.7867	80.05%	77.39%	0.9515	72.58%
7	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.4542	84.96%	0.8475	86.12%	83.46%	0.9744	77.88%
8	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.5117	82.81%	0.8254	82.99%	82.11%	0.9684	76.05%
9	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.5731	82.76%	0.8257	83.21%	81.96%	0.9642	75.97%
10	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.4978	83.68%	0.8391	85.06%	82.82%	0.9696	76.92%
11	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.3687	88.34%	0.8826	88.89%	87.66%	0.9843	80.77%
12	SGD	32	0.17	ReLU	Glorot normal	L2(10^{-2})	0.6447	80.82%	0.8067	82.24%	79.20%	0.9623	74.40%
13	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.4143	86.85%	0.8675	87.32%	86.20%	0.9792	79.49%
14	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.4116	87.95%	0.8786	88.36%	87.38%	0.9805	80.38%
15	SGD	32	0.17	ReLU	Glorot normal	L1(10^{-5})	0.3124	91.38%	0.9140	91.77%	91.04%	0.9894	83.33%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

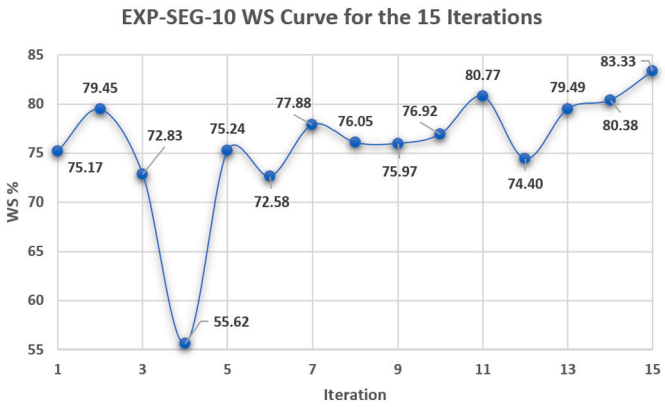


Fig. 20. EXP-SEG-10 WS curve for the 15 iterations.

From Table 11:, the AdaMax optimizer was the best in 8 iterations. The 64 batch size was the best in 8 iterations. The 30% dropout ratio was the best in 12 iterations. The 90% TF learn ratio was the best in 6 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0311, 99.25%, 0.9926, 99.26%, 99.26%, 0.9990, and 92.61% respectively while the last reported iteration metrics were 0.0311, 99.25%, 0.9926, 99.26%, 99.26%, 0.9989, and 92.61% respectively. The WS and TF learn ratio had a moderate positive correlation (0.67) while the WS and dropout ratio had a low positive correlation (0.32). The dropout ratio and TF learn ratio had negligible correlation (0.11). Fig. 15 shows the EXP-SEG-5 WS curve for the 15 iterations.

Table 12 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the DenseNet121 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 12, the AdaMax optimizer was the best in 8 iterations. The 32 batch size was the best in 13 iterations. The 10% dropout ratio was the best in 9 iterations. The 100% TF learn ratio was the best in 15 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0814, 97.69%, 0.9772, 97.72%, 97.72%, 0.9974, and 89.37% respectively while the last reported iteration metrics were 0.1963, 93.99%, 0.9388, 93.93%, 93.83%, 0.9922, and 85.59% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had negligible correlation (−0.04). The dropout ratio and TF learn ratio could not be determined. Fig. 16 shows the EXP-SEG-6 WS curve for the 15 iterations.

Table 13 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the DenseNet169 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 13, the AdaGrad optimizer was the best in 12 iterations. The 64 batch size was the best in 10 iterations. The 30% dropout ratio was the best in 15 iterations. The 100% TF learn ratio was the best in 8 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.1236, 95.87%, 0.9576, 95.78%, 95.73%, 0.9955, and 87.43% respectively while the last reported iteration metrics were 0.2161, 93.67%, 0.9360, 93.80%, 93.40%, 0.9892, and 85.27% respectively. The WS and TF learn ratio had a very high positive correlation (0.96) while the WS and dropout ratio had negligible correlation (0.00). The dropout ratio and TF learn ratio had negligible correlation (0.00). Fig. 17 shows the EXP-SEG-7 WS curve for the 15 iterations.

Table 14 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the MobileNet pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 14, the SGD optimizer was the best in 6 iterations. The 64 batch size was the best in 13 iterations. The 20% dropout ratio was the best in 14 iterations. The 100% TF learn ratio was the best in 12 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.1047, 96.14%, 0.9621, 96.29%, 96.13%, 0.9976, and 87.86% respectively while the last reported iteration metrics were 0.2108, 94.53%, 0.9432, 94.42%, 94.22%, 0.9878, and 85.91% respectively. The WS and TF learn ratio had a very high positive correlation (0.95) while the WS and dropout ratio had negligible correlation (0.17). The dropout ratio and TF learn ratio had negligible correlation (0.13). Fig. 18 shows the EXP-SEG-8 WS curve for the 15 iterations.

Table 15 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the MobileNetV2 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 15, the AdaMax optimizer was the best in 9 iterations. The 64 batch size was the best in 10 iterations. The 30% dropout ratio

Table 17
Tabular summary of the top-1 experiments with segmentation concerning the highest WS.

Experiment	Iteration	Model	PO[a]	Batch size	DR[a]	TLR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
EXP-SEG-1	VGG16	13	SGD	32	0.4	90%	NA	NA	NA	0.0176	99.57%	0.9958	99.58%	99.58%	0.9996	95.33%
EXP-SEG-2	VGG19	15	SGD	32	0.2	85%	NA	NA	NA	0.0170	99.46%	0.9947	99.47%	99.47%	0.9996	95.44%
EXP-SEG-3	ResNet50	6	AdaMax	64	0.2	10%	NA	NA	NA	2.3876	31.58%	0.3120	32.23%	30.26%	0.5429	30.63%
EXP-SEG-4	ResNet101	14	SGD	32	0.0	100%	NA	NA	NA	0.0663	97.59%	0.9759	97.66%	97.51%	0.9985	89.56%
EXP-SEG-5	Xception	15	AdaMax	32	0.3	100%	NA	NA	NA	0.0311	99.25%	0.9926	99.26%	99.26%	0.9989	92.61%
EXP-SEG-6	DenseNet121	13	AdaMax	32	0.1	100%	NA	NA	NA	0.0814	97.69%	0.9772	97.72%	97.72%	0.9974	89.37%
EXP-SEG-7	DenseNet169	12	Adam	64	0.3	100%	NA	NA	NA	0.1236	95.87%	0.9576	95.78%	95.73%	0.9955	87.43%
EXP-SEG-8	MobileNet	10	Nadam	64	0.2	100%	NA	NA	NA	0.1047	96.14%	0.9621	96.29%	96.13%	0.9976	87.86%
EXP-SEG-9	MobileNetV2	12	SGD	64	0.3	100%	NA	NA	NA	0.4265	90.08%	0.9046	90.73%	90.20%	0.9726	82.14%
EXP-SEG-10	HMB1-COVID19	15	SGD	32	0.17	NA	ReLU	Glorot Normal	L1(10^{-5})	0.3124	91.38%	0.9140	91.77%	91.04%	0.9894	83.33%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

Table 18
Tabular summary of the top-1 experiments with segmentation concerning the last reported results.

Experiment	Model	PO	Batch size	DR[a]	TLR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
EXP-SEG-1	VGG16	SGD	32	0.1	90%	NA	NA	NA	0.0372	99.30%	0.9931	99.31%	99.31%	0.9981	92.12%
EXP-SEG-2	VGG19	SGD	32	0.2	85%	NA	NA	NA	0.0170	99.46%	0.9947	99.47%	99.47%	0.9996	95.44%
EXP-SEG-3	ResNet50	AdaMax	32	0.1	20%	NA	NA	NA	4.0317	28.10%	0.2781	27.84%	27.78%	0.5411	27.83%
EXP-SEG-4	ResNet101	AdaGrad	32	0.0	100%	NA	NA	NA	0.0936	97.10%	0.9711	97.14%	97.09%	0.9957	88.71%
EXP-SEG-5	Xception	AdaMax	32	0.3	100%	NA	NA	NA	0.0311	99.25%	0.9926	99.26%	99.26%	0.9989	92.61%
EXP-SEG-6	DenseNet121	SGD	64	0.1	100%	NA	NA	NA	0.1963	93.99%	0.9388	93.93%	93.83%	0.9922	85.59%
EXP-SEG-7	DenseNet169	Adam	64	0.3	100%	NA	NA	NA	0.2161	93.67%	0.9360	93.80%	93.40%	0.9892	85.27%
EXP-SEG-8	MobileNet	AdaMax	32	0.3	100%	NA	NA	NA	0.2108	94.53%	0.9432	94.42%	94.22%	0.9878	85.91%
EXP-SEG-9	MobileNetV2	SGD	64	0.2	100%	NA	NA	NA	0.4998	86.70%	0.8641	86.58%	86.25%	0.9685	79.16%
EXP-SEG-10	HMB1-COVID19	SGD	32	0.17	NA	ReLU	Glorot normal	L1(10^{-5})	0.3124	91.38%	0.9140	91.77%	91.04%	0.9894	83.33%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

Table 19
EXP-NO-SEG-1: first HMB-HCF learning and optimization experiment without segmentation and with VGG16.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaMax	32	0.0	30%	0.0444	98.27%	0.9823	98.23%	98.23%	0.9996	90.85%
2	AdaGrad	32	0.2	55%	0.0404	98.59%	0.9856	98.56%	98.56%	0.9995	91.33%
3	SGD	32	0.2	55%	0.0375	98.75%	0.9872	98.72%	98.72%	0.9989	91.65%
4	SGD	32	0.1	55%	0.0233	99.30%	0.9930	99.30%	99.30%	0.9996	93.73%
5	SGD	32	0.2	55%	0.0134	99.62%	0.9957	99.60%	99.55%	0.9996	97.13%
6	AdaGrad	32	0.0	55%	0.0168	99.57%	0.9955	99.55%	99.55%	0.9997	95.60%
7	SGD	32	0.0	55%	0.0138	99.78%	0.9976	99.78%	99.73%	0.9996	97.06%
8	SGD	32	0.2	55%	0.0135	99.68%	0.9968	99.68%	99.68%	0.9996	97.15%
9	SGD	32	0.3	55%	0.0151	99.57%	0.9957	99.57%	99.57%	0.9996	96.29%
10	SGD	32	0.3	55%	0.0120	99.62%	0.9962	99.62%	99.62%	0.9997	98.06%
11	SGD	64	0.5	80%	0.0099	99.78%	0.9978	99.78%	99.78%	0.9996	99.88%
12	AdaGrad	32	0.0	55%	0.0158	99.51%	0.9947	99.50%	99.44%	0.9997	95.94%
13	SGD	32	0.5	80%	0.0097	99.78%	0.9984	99.89%	99.78%	0.9996	100.0%
14	SGD	64	0.2	55%	0.0147	99.62%	0.9960	99.60%	99.60%	0.9997	96.48%
15	SGD	32	0.3	55%	0.0099	99.84%	0.9984	99.84%	99.84%	0.9996	99.92%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

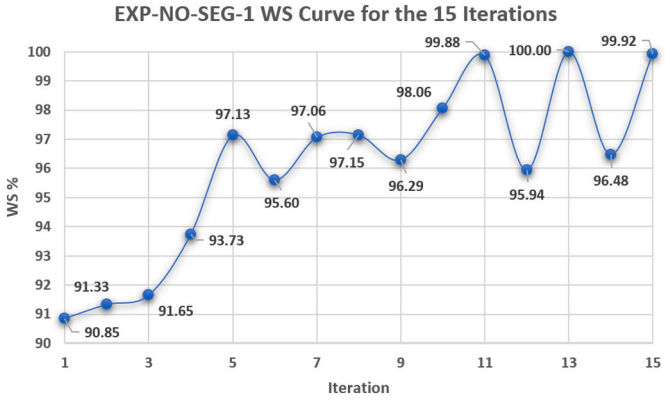


Fig. 21. EXP-NO-SEG-1 WS curve for the 15 iterations.

was the best in 9 iterations. The 100% TF learn ratio was the best in 14 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.3570, 90.24%, 0.9046, 90.73%, 90.20%, 0.9813, and 82.18% respectively while the last reported iteration metrics were 0.4998, 86.70%, 0.8641, 86.58%, 86.25%, 0.9685, and 79.16% respectively. The WS and TF learn ratio had a very high positive correlation (0.97) while the WS and dropout ratio had negligible correlation (0.22). The dropout ratio and TF learn ratio had negligible correlation (0.14). Fig. 19 shows the EXP-SEG-9 WS curve for the 15 iterations.

Table 16 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the proposed HMB1-COVID19 CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 16, the SGD optimizer was the best in 12 iterations. The 32 batch size was the best in 14 iterations. The 17% dropout ratio was the best in 12 iterations. The ReLU hidden activation was the best in 13 iterations. The Glorot Normal parameters initializer was the best in 15 iterations. The L1(10^{-5}) regularizer was the best in 11 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.3124, 91.38%, 0.9140, 91.77%, 91.04%, 0.9894, and

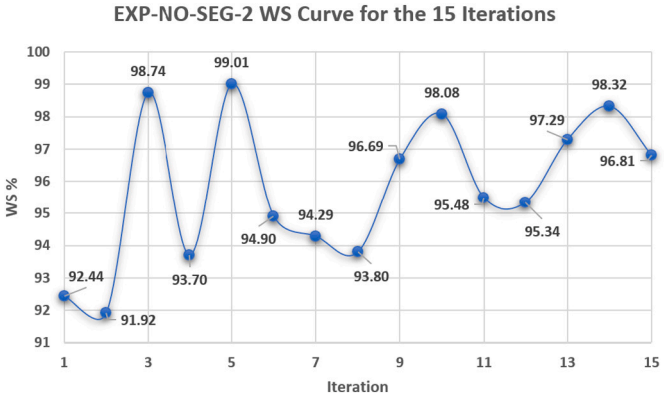


Fig. 22. EXP-NO-SEG-2 WS curve for the 15 iterations.

Table 20
EXP-NO-SEG-2: second HMB-HCF learning and optimization experiment without segmentation and with VGG19.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	SGD	64	0.2	85%	0.0301	98.92%	0.9887	98.90%	98.85%	0.9996	92.44%
2	SGD	64	0.2	85%	0.0343	98.75%	0.9879	98.87%	98.71%	0.9994	91.92%
3	SGD	64	0.2	85%	0.0112	99.73%	0.9973	99.73%	99.73%	0.9997	98.74%
4	SGD	64	0.2	85%	0.0237	99.35%	0.9938	99.41%	99.35%	0.9996	93.70%
5	SGD	64	0.2	85%	0.0107	99.62%	0.9962	99.62%	99.62%	0.9996	99.01%
6	AdaMax	32	0.1	75%	0.0189	99.51%	0.9954	99.57%	99.52%	0.9996	94.90%
7	AdaMax	32	0.1	75%	0.0218	99.62%	0.9962	99.62%	99.62%	0.9996	94.29%
8	SGD	64	0.2	85%	0.0229	99.30%	0.9930	99.30%	99.30%	0.9996	93.80%
9	SGD	64	0.3	75%	0.0144	99.68%	0.9968	99.68%	99.68%	0.9996	96.69%
10	SGD	64	0.2	85%	0.0121	99.73%	0.9973	99.73%	99.73%	0.9997	98.08%
11	SGD	64	0.1	75%	0.0172	99.57%	0.9957	99.57%	99.57%	0.9996	95.48%
12	SGD	64	0.1	85%	0.0172	99.40%	0.9943	99.46%	99.41%	0.9996	95.34%
13	SGD	32	0.0	85%	0.0131	99.57%	0.9957	99.57%	99.57%	1.000	97.29%
14	SGD	32	0.3	85%	0.0116	99.68%	0.9966	99.66%	99.66%	0.9996	98.32%
15	SGD	32	0.0	85%	0.0139	99.52%	0.9952	99.52%	99.52%	1.000	96.81%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 21
EXP-NO-SEG-3: third HMB-HCF learning and optimization experiment without segmentation and with ResNet50.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	Nadam	64	0.1	40%	2.1265	29.62%	0.2363	29.04%	20.03%	0.5429	27.55%
2	Ftrl	64	0.3	35%	1.9089	31.67%	0.2652	44.76%	19.06%	0.5475	30.40%
3	Adam	64	0.2	10%	3.4394	37.57%	0.3668	40.29%	33.74%	0.5661	35.55%
4	Ftrl	64	0.3	35%	1.9754	26.96%	0.2877	33.49%	25.35%	0.5579	27.87%
5	Nadam	64	0.2	10%	1.8882	32.32%	0.2461	33.75%	19.51%	0.5451	29.45%
6	Adam	64	0.2	10%	3.0229	33.46%	0.2903	40.39%	22.87%	0.5999	31.99%
7	Ftrl	64	0.3	55%	1.9645	37.14%	0.1691	65.29%	10.01%	0.5686	33.53%
8	Adam	64	0.2	50%	10.974	26.85%	0.2693	26.93%	26.93%	0.5747	27.26%
9	Ftrl	64	0.3	35%	1.9008	28.59%	0.2752	44.23%	20.24%	0.5525	29.07%
10	Ftrl	32	0.4	10%	1.7385	26.85%	0.2878	35.32%	24.40%	0.5498	27.83%
11	Ftrl	32	0.2	10%	1.8391	26.85%	0.2702	27.35%	26.72%	0.5529	27.12%
12	Ftrl	32	0.4	10%	1.6795	26.85%	0.2491	40.85%	18.14%	0.5455	27.33%
13	RMSProp	64	0.2	30%	7.5304	36.98%	0.3705	37.05%	37.05%	0.5549	35.17%
14	Ftrl	32	0.4	10%	1.6821	26.85%	0.2849	33.86%	24.73%	0.5444	27.64%
15	Ftrl	32	0.4	10%	1.7133	26.85%	0.2839	37.29%	23.09%	0.5558	27.92%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

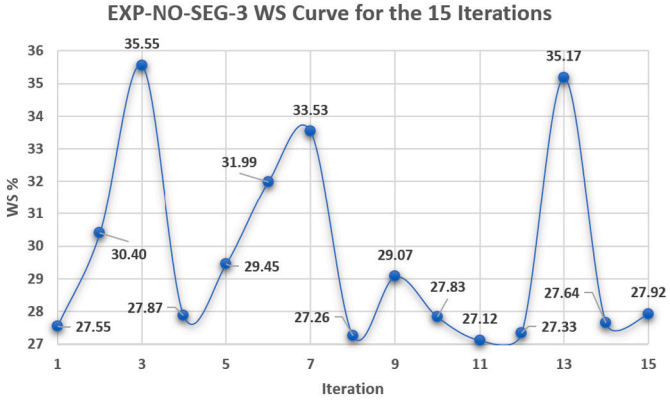


Fig. 23. EXP-NO-SEG-3 WS curve for the 15 iterations.

83.33% respectively while the last reported iteration metrics were 0.3124, 91.38%, 0.9140, 91.77%, 91.04%, 0.9894, and 83.33% respectively. The WS and dropout ratio had a negligible correlation (0.11). Fig. 20 shows the EXP-SEG-10 WS curve for the 15 iterations.

Table 17 summarizes the obtained top-1 results in each experiment with segmentation concerning the highest WS metrics. Table 18 summarizes the obtained top-1 results in each experiment with segmentation concerning the last reported results.

4.3. HMB-HCF learning and optimization experiments without segmentation

HMB1-COVID19 and the pre-trained CNN models are subjected to be optimized where each one at a time. This is applied to get the best combinations from each of them to be used later in the hybrid architecture. To distinguish between the experiments, the annotation "EXP-NO-SEG-xx" is used to denote the HMB-HCF learning and optimization experiments without segmentation where "xx" is the experiment number.

Table 19 reports the top-1 combination in each hyperparameters'

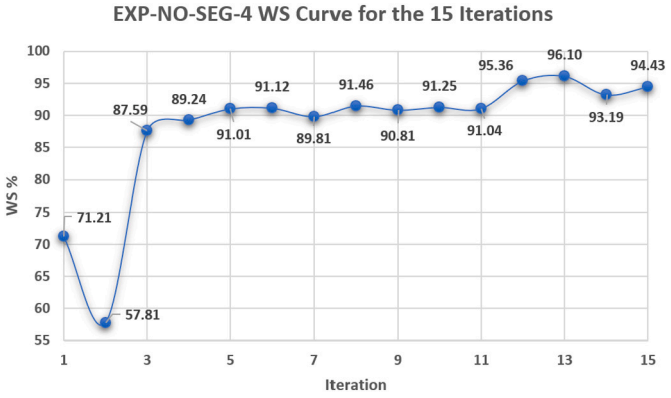


Fig. 24. EXP-NO-SEG-4 WS curve for the 15 iterations.

Table 22
EXP-NO-SEG-4: fourth HMB-HCF learning and optimization experiment without segmentation and with ResNet101.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	Adam	32	0.3	100%	0.5661	76.94%	0.7700	80.37%	74.00%	0.9434	71.21%
2	Adam	32	0.3	100%	2.9887	61.88%	0.6196	62.60%	61.34%	0.8245	57.81%
3	AdaGrad	32	0.3	100%	0.1390	96.16%	0.9615	96.15%	96.15%	0.9942	87.59%
4	AdaGrad	32	0.3	100%	0.0770	97.46%	0.9749	97.52%	97.47%	0.9962	89.24%
5	AdaGrad	32	0.3	100%	0.0469	98.59%	0.9865	98.71%	98.60%	0.9990	91.01%
6	AdaGrad	32	0.3	100%	0.0435	98.54%	0.9852	98.52%	98.52%	0.9997	91.12%
7	AdaGrad	32	0.5	100%	0.0678	97.94%	0.9791	97.91%	97.91%	0.9987	89.81%
8	AdaGrad	32	0.5	100%	0.0405	98.75%	0.9876	98.76%	98.76%	0.9988	91.46%
9	AdaGrad	64	0.3	100%	0.0462	98.32%	0.9833	98.33%	98.33%	0.9987	90.81%
10	AdaGrad	32	0.5	100%	0.0397	98.43%	0.9840	98.40%	98.40%	0.9998	91.25%
11	AdaGrad	64	0.5	100%	0.0504	98.86%	0.9885	98.85%	98.85%	0.9974	91.04%
12	SGD	64	0.0	100%	0.0172	99.46%	0.9941	99.44%	99.39%	0.9992	95.36%
13	SGD	64	0.5	100%	0.0157	99.68%	0.9968	99.68%	99.68%	0.9996	96.10%
14	SGD	32	0.0	100%	0.0263	98.75%	0.9865	98.76%	98.55%	0.9997	93.19%
15	SGD	32	0.0	100%	0.0214	99.73%	0.9973	99.73%	99.73%	0.9985	94.43%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 23
EXP-NO-SEG-5: fifth HMB-HCF learning and optimization experiment without segmentation and with Xception.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaGrad	64	0.3	80%	0.7671	79.21%	0.7935	80.11%	78.62%	0.9403	72.95%
2	AdaGrad	64	0.3	80%	0.7672	79.86%	0.7977	80.45%	79.14%	0.9436	73.43%
3	AdaGrad	64	0.3	80%	0.7462	79.97%	0.8007	80.75%	79.43%	0.9431	73.57%
4	AdaGrad	64	0.3	80%	0.6754	81.65%	0.8157	81.90%	81.25%	0.9482	74.93%
5	AdaGrad	64	0.3	80%	0.5898	82.13%	0.8210	82.82%	81.41%	0.9559	75.43%
6	AdaGrad	64	0.3	80%	0.7505	79.86%	0.8005	80.53%	79.59%	0.9428	73.51%
7	AdaGrad	64	0.3	80%	0.7095	81.05%	0.8098	81.38%	80.60%	0.9490	74.45%
8	Nadam	64	0.1	80%	0.7719	81.05%	0.8106	81.51%	80.63%	0.9383	74.36%
9	AdaGrad	64	0.1	80%	0.6485	82.24%	0.8228	82.63%	81.94%	0.9517	75.48%
10	AdaGrad	64	0.1	80%	0.5924	83.97%	0.8381	84.13%	83.49%	0.9586	76.88%
11	AdaGrad	64	0.1	80%	0.6127	84.79%	0.8473	85.04%	84.44%	0.9561	77.54%
12	AdaGrad	64	0.1	80%	0.5690	85.22%	0.8508	85.29%	84.87%	0.9616	77.92%
13	AdaGrad	64	0.1	80%	0.6391	84.68%	0.8468	84.91%	84.46%	0.9526	77.43%
14	AdaGrad	32	0.1	80%	0.5521	85.22%	0.8489	85.27%	84.53%	0.9601	77.86%
15	AdaGrad	64	0.1	80%	0.5740	84.46%	0.8459	84.97%	84.23%	0.9610	77.40%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

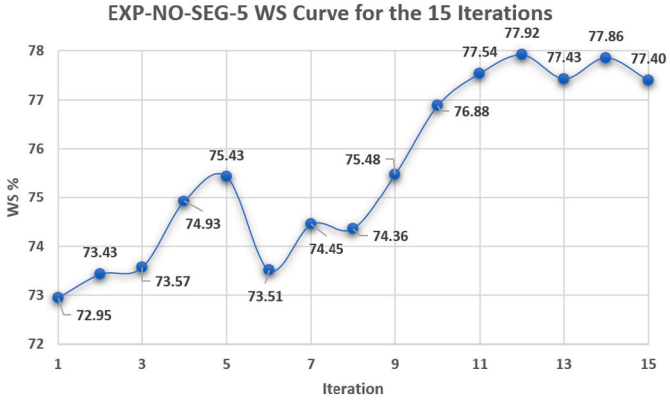


Fig. 25. EXP-NO-SEG-5 WS curve for the 15 iterations.

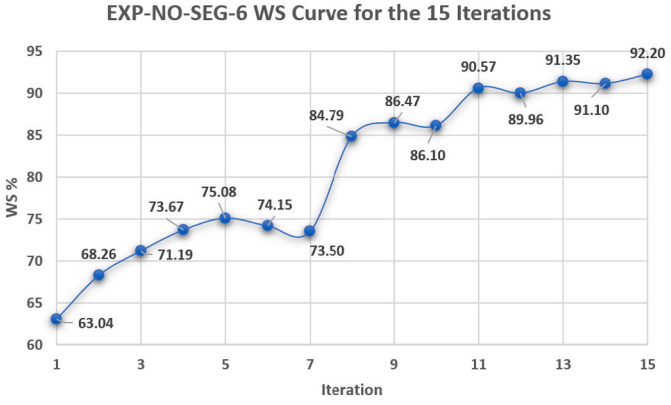


Fig. 26. EXP-NO-SEG-6 WS curve for the 15 iterations.

optimization iteration in the 15 iterations for the VGG16 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 19, the SGD optimizer was the best in 11 iterations. The 32 batch size was the best in 13 iterations. The 20% dropout ratio was the best in 9 iterations. The 55% TF learn ratio was the best in 12 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0097, 99.84%, 0.9984, 99.89%, 99.84%, 0.9997, and 100.0% respectively while the last reported iteration metrics were 0.0099, 99.84%, 0.9984, 99.84%, 99.84%, 0.9996, and 99.92% respectively. The WS and TF learn ratio had a moderate positive correlation (0.67) while the WS and dropout ratio had a moderate positive correlation (0.59). The dropout ratio and TF learn ratio

Table 24
EXP-NO-SEG-6: sixth HMB-HCF learning and optimization experiment without segmentation and with DenseNet121.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	SGD	32	0.2	55%	1.1508	67.57%	0.6736	67.86%	66.88%	0.8963	63.04%
2	SGD	32	0.2	55%	1.1467	73.80%	0.7380	74.01%	73.59%	0.9140	68.26%
3	RMSProp	32	0.3	55%	6.8218	77.96%	0.7792	77.92%	77.92%	0.8814	71.19%
4	AdaMax	64	0.2	60%	1.2286	80.40%	0.8049	80.81%	80.17%	0.9236	73.67%
5	RMSProp	32	0.3	55%	3.9172	82.51%	0.8247	82.47%	82.47%	0.9061	75.08%
6	AdaMax	64	0.3	55%	1.1584	80.94%	0.8094	81.05%	80.84%	0.9312	74.15%
7	AdaMax	64	0.2	60%	1.1925	80.18%	0.8019	80.40%	79.99%	0.9264	73.50%
8	AdaMax	64	0.2	100%	0.2635	93.18%	0.9318	93.23%	93.14%	0.9868	84.79%
9	AdaMax	64	0.2	100%	0.2072	95.13%	0.9512	95.20%	95.04%	0.9891	86.47%
10	AdaMax	64	0.2	100%	0.2093	94.69%	0.9461	94.86%	94.39%	0.9885	86.10%
11	AdaMax	64	0.2	100%	0.0510	98.27%	0.9828	98.28%	98.28%	0.9990	90.57%
12	AdaGrad	64	0.2	100%	0.0543	97.67%	0.9766	97.66%	97.66%	0.9989	89.96%
13	AdaGrad	64	0.2	100%	0.0389	98.43%	0.9857	98.70%	98.44%	0.9994	91.35%
14	AdaGrad	64	0.2	100%	0.0433	98.48%	0.9852	98.55%	98.49%	0.9991	91.10%
15	AdaGrad	64	0.2	100%	0.0334	99.03%	0.9903	99.03%	99.03%	0.9988	92.20%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 25
EXP-NO-SEG-7: seventh HMB-HCF learning and optimization experiment without segmentation and with DenseNet169.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	AdaGrad	32	0.2	100%	0.0379	99.35%	0.9933	99.35%	99.30%	0.9985	92.10%
2	AdaGrad	32	0.2	100%	0.0388	98.97%	0.9889	98.97%	98.81%	0.9981	91.71%
3	AdaGrad	32	0.2	100%	0.0387	98.70%	0.9871	98.74%	98.69%	0.9991	91.54%
4	AdaGrad	32	0.2	100%	0.0319	99.24%	0.9916	99.24%	99.08%	0.9985	92.49%
5	AdaGrad	64	0.2	100%	0.0217	99.13%	0.9914	99.14%	99.14%	0.9996	93.90%
6	AdaGrad	64	0.2	100%	0.0257	99.24%	0.9927	99.30%	99.25%	0.9993	93.29%
7	AdaGrad	64	0.5	100%	0.0196	99.51%	0.9952	99.52%	99.52%	0.9993	94.71%
8	AdaGrad	64	0.2	100%	0.0182	99.24%	0.9925	99.25%	99.25%	0.9993	94.88%
9	SGD	64	0.2	100%	0.0124	99.51%	0.9951	99.51%	99.51%	0.9997	97.65%
10	AdaGrad	64	0.5	100%	0.0202	99.68%	0.9968	99.68%	99.68%	0.9992	94.69%
11	AdaGrad	64	0.2	100%	0.0236	99.19%	0.9919	99.19%	99.19%	0.9996	93.59%
12	AdaGrad	64	0.5	100%	0.0197	99.51%	0.9951	99.51%	99.51%	0.9990	94.69%
13	AdaGrad	64	0.5	100%	0.0163	99.73%	0.9973	99.73%	99.73%	0.9993	95.93%
14	AdaGrad	64	0.5	100%	0.0179	99.62%	0.9962	99.62%	99.62%	0.9993	95.29%
15	AdaGrad	64	0.5	100%	0.0157	99.62%	0.9962	99.62%	99.62%	0.9993	96.05%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

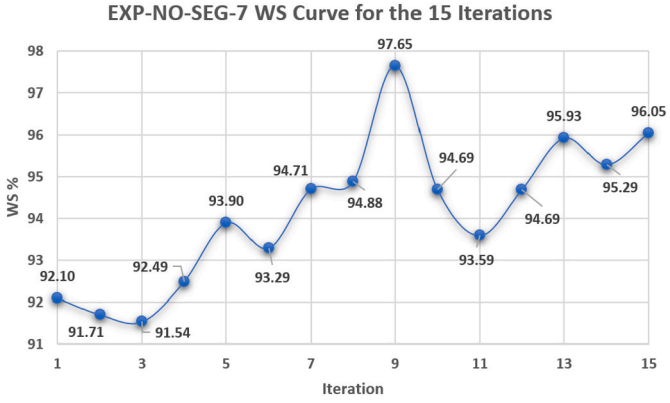


Fig. 27. EXP-NO-SEG-7 WS curve for the 15 iterations.

had a high positive correlation (0.76). Fig. 21 shows the EXP-NO-SEG-1 WS curve for the 15 iterations.

Table 20 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the VGG19 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 20, the SGD optimizer was the best in 13 iterations. The 64 batch size was the best in 10 iterations. The 20% dropout ratio was the best in 7 iterations. The 85% TF learn ratio was the best in 11 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0107, 99.73%, 0.9973, 99.73%, 99.73%, 1.000, and 99.01% respectively while the last reported iteration metrics were 0.0139, 99.52%, 0.9952, 99.52%, 99.52%, 1.000, and 96.81% respectively. The WS and TF learn ratio had negligible correlation (0.12) while the WS and dropout ratio had negligible correlation (0.05). The dropout ratio and TF learn ratio had negligible correlation (0.07). Fig. 22 shows the EXP-NO-SEG-2 WS curve for the 15 iterations.

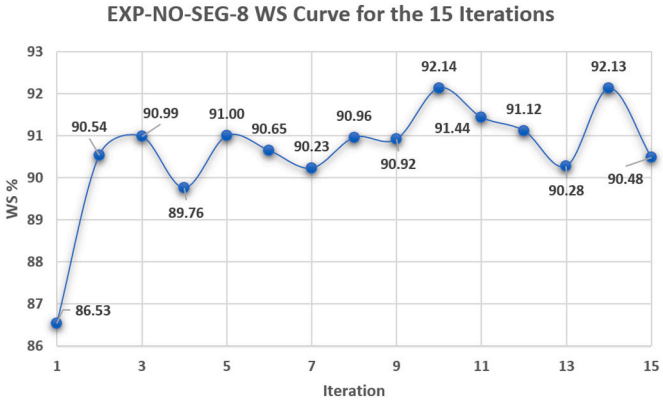


Fig. 28. EXP-NO-SEG-8 WS curve for the 15 iterations.

Table 26
EXP-NO-SEG-8: eigth HMB-HCF learning and optimization experiment without segmentation and with MobileNet.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	SGD	32	0.3	100%	0.1576	94.91%	0.9499	95.10%	94.89%	0.9945	86.53%
2	SGD	32	0.3	100%	0.0523	98.32%	0.9827	98.38%	98.17%	0.9988	90.54%
3	SGD	32	0.3	100%	0.0444	98.43%	0.9844	98.44%	98.44%	0.9994	90.99%
4	SGD	32	0.3	100%	0.0728	98.00%	0.9803	98.06%	98.01%	0.9980	89.76%
5	SGD	32	0.3	100%	0.0460	98.54%	0.9855	98.55%	98.55%	0.9990	91.00%
6	SGD	32	0.3	100%	0.0482	98.21%	0.9825	98.28%	98.22%	0.9991	90.65%
7	SGD	32	0.3	100%	0.0644	98.38%	0.9838	98.38%	98.38%	0.9978	90.23%
8	Nadam	32	0.3	100%	0.0466	98.48%	0.9860	98.71%	98.50%	0.9991	90.96%
9	SGD	32	0.3	100%	0.0494	98.64%	0.9863	98.63%	98.63%	0.9980	90.92%
10	SGD	32	0.3	100%	0.0322	98.81%	0.9879	98.79%	98.79%	0.9996	92.14%
11	SGD	32	0.5	100%	0.0423	98.86%	0.9887	98.87%	98.87%	0.9987	91.44%
12	SGD	32	0.5	100%	0.0426	98.48%	0.9847	98.47%	98.47%	0.9988	91.12%
13	SGD	32	0.5	100%	0.0596	98.32%	0.9823	98.29%	98.18%	0.9975	90.28%
14	SGD	32	0.3	100%	0.0337	98.97%	0.9898	98.98%	98.98%	0.9985	92.13%
15	SGD	32	0.5	100%	0.0610	98.59%	0.9858	98.58%	98.58%	0.9973	90.48%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

Table 27
EXP-NO-SEG-9: ninth HMB-HCF learning and optimization experiment without segmentation and with MobileNetV2.

Iteration	PO[a]	Batch size	DR[a]	TLR[a]	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	RMSProp	64	0.1	100%	11.1682	73.85%	0.7389	73.89%	73.89%	0.8351	67.45%
2	RMSProp	64	0.1	100%	5.3725	79.75%	0.7974	79.74%	79.74%	0.8809	72.63%
3	AdaMax	32	0.1	100%	0.4518	89.44%	0.8942	89.60%	89.25%	0.9723	81.49%
4	AdaMax	32	0.0	100%	0.2549	92.69%	0.9267	92.75%	92.60%	0.9875	84.41%
5	AdaMax	32	0.0	100%	0.1669	95.23%	0.9527	95.32%	95.22%	0.9921	86.72%
6	AdaMax	32	0.3	100%	0.2514	93.39%	0.9339	93.63%	93.17%	0.9861	84.97%
7	AdaMax	32	0.3	100%	0.1208	96.64%	0.9666	96.70%	96.64%	0.9958	88.11%
8	AdaMax	32	0.0	100%	0.1440	96.10%	0.9607	96.15%	95.99%	0.9932	87.50%
9	SGD	32	0.3	100%	0.1574	95.07%	0.9510	95.28%	94.93%	0.9938	86.64%
10	SGD	32	0.3	100%	0.1174	96.59%	0.9666	96.76%	96.56%	0.9951	88.10%
11	SGD	32	0.3	100%	0.1116	96.97%	0.9694	96.94%	96.94%	0.9958	88.42%
12	SGD	32	0.1	100%	0.1395	96.53%	0.9653	96.53%	96.53%	0.9934	87.88%
13	SGD	32	0.3	100%	0.0921	97.73%	0.9771	97.74%	97.68%	0.9962	89.22%
14	SGD	32	0.3	100%	0.1258	95.83%	0.9588	95.96%	95.80%	0.9961	87.43%
15	SGD	32	0.3	100%	0.1243	97.02%	0.9699	97.04%	96.94%	0.9934	88.35%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio.

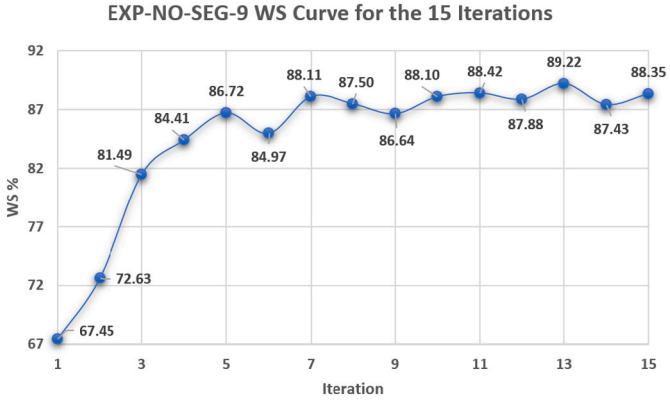


Fig. 29. EXP-NO-SEG-9 WS curve for the 15 iterations.

Table 21 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the ResNet50 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 21, the Ftrl optimizer was the best in 9 iterations. The 64 batch size was the best in 10 iterations. The 20% dropout ratio was the best in 6 iterations. The 10% TF learn ratio was the best in 8 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 1.6795, 37.57%, 0.3705, 65.29%, 37.05%, 0.5999, and 35.55% respectively while the last reported iteration metrics were 1.7133, 26.85%, 0.2839, 37.29%, 23.09%, 0.5558, and 27.92% respectively. The WS and TF learn ratio had negligible correlation (0.11) while the WS and dropout ratio had a low negative correlation (−0.30). The dropout ratio and TF learn ratio had negligible correlation (−0.29). Fig. 23 shows the EXP-NO-SEG-3 WS curve for the 15 iterations.

Table 22 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the ResNet101 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 22, the AdaGrad optimizer was the best in 9 iterations. The 32 batch size was the best in 11 iterations. The 30% dropout ratio was the best in 7 iterations. The 100% TF learn ratio was the best in 15

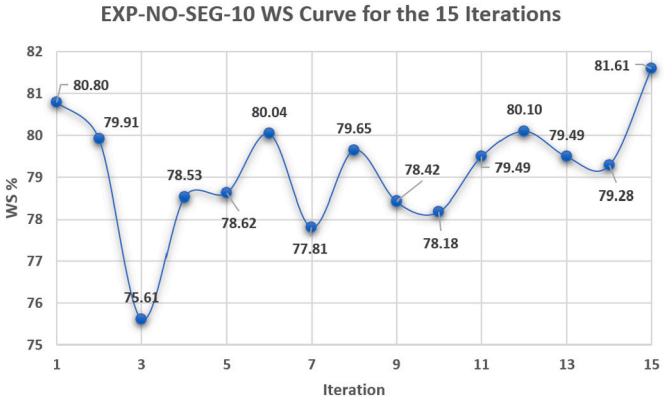


Fig. 30. EXP-NO-SEG-10 WS curve for the 15 iterations.

Table 28
EXP-NO-SEG-10: tenth HMB-HCF learning and optimization experiment without segmentation and HMB1-COVID19.

Iteration	PO[a]	Batch size	DR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
1	Nadam	32	0.11	ELU	Glorot uniform	L1(10^{-5})	0.4888	88.44%	0.8858	88.82%	88.34%	0.9804	80.80%
2	Nadam	32	0.11	ELU	Glorot uniform	L1(10^{-5})	0.5268	87.40%	0.8748	87.68%	87.28%	0.9775	79.91%
3	RMSProp	32	0.47	ReLU	He normal	L1(10^{-5})	0.5306	82.30%	0.8196	82.72%	81.24%	0.9676	75.61%
4	SGD	32	0.37	ReLU	He normal	L1(10^{-4})	0.6864	85.86%	0.8583	86.02%	85.64%	0.9703	78.53%
5	RMSProp	32	0.47	ELU	Glorot uniform	L1(10^{-5})	0.4572	85.75%	0.8587	86.29%	85.47%	0.9763	78.62%
6	SGD	32	0.37	ReLU	He normal	L1(10^{-4})	0.5570	87.57%	0.8756	87.98%	87.16%	0.9805	80.04%
7	SGD	32	0.37	ReLU	He normal	L1(10^{-5})	0.4887	84.90%	0.8469	85.85%	83.60%	0.9743	77.81%
8	SGD	32	0.37	ReLU	He normal	L1(10^{-4})	0.5827	87.08%	0.8713	87.54%	86.73%	0.9796	79.65%
9	RMSProp	32	0.47	ReLU	He normal	L1(10^{-5})	0.4591	85.53%	0.8556	86.37%	84.79%	0.9768	78.42%
10	RMSProp	32	0.47	ELU	Glorot uniform	L1(10^{-5})	0.5154	85.32%	0.8534	85.79%	84.91%	0.9724	78.18%
11	RMSProp	32	0.47	ELU	Glorot uniform	L1(10^{-5})	0.4382	86.85%	0.8680	87.32%	86.30%	0.9799	79.49%
12	RMSProp	32	0.27	ELU	Glorot uniform	L1(10^{-5})	0.4058	87.61%	0.8739	87.95%	86.86%	0.9824	80.10%
13	RMSProp	64	0.27	ReLU	He normal	L1(10^{-5})	0.4973	86.97%	0.8677	87.30%	86.26%	0.9773	79.49%
14	RMSProp	32	0.27	ELU	Glorot uniform	L1(10^{-5})	0.4717	86.59%	0.8663	86.80%	86.46%	0.9783	79.28%
15	RMSProp	64	0.27	ReLU	He normal	L1(10^{-5})	0.4037	89.37%	0.8937	89.69%	89.05%	0.9863	81.61%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

Table 29
Tabular summary of the top-1 experiments without segmentation concerning the highest WS.

Experiment	Iteration	Model	PO[a]	Batch size	DR[a]	TLR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
EXP-NO-SEG-1	VGG16	13	SGD	32	0.5	80%	NA	NA	NA	0.0097	99.78%	0.9984	99.89%	99.78%	0.9996	100.0%
EXP-NO-SEG-2	VGG19	5	SGD	64	0.2	85%	NA	NA	NA	0.0107	99.62%	0.9962	99.62%	99.62%	0.9996	99.01%
EXP-NO-SEG-3	ResNet50	3	Adam	64	0.2	10%	NA	NA	NA	3.4394	37.57%	0.3668	40.29%	33.74%	0.5661	35.55%
EXP-NO-SEG-4	ResNet101	13	SGD	64	0.5	100%	NA	NA	NA	0.0157	99.68%	0.9968	99.68%	99.68%	0.9996	96.10%
EXP-NO-SEG-5	Xception	12	AdaGrad	64	0.1	80%	NA	NA	NA	0.5690	85.22%	0.8508	85.29%	84.87%	0.9616	77.92%
EXP-NO-SEG-6	DenseNet121	15	AdaGrad	64	0.2	100%	NA	NA	NA	0.0334	99.03%	0.9903	99.03%	99.03%	0.9988	92.20%
EXP-NO-SEG-7	DenseNet169	9	SGD	64	0.2	100%	NA	NA	NA	0.0124	99.51%	0.9951	99.51%	99.51%	0.9997	97.65%
EXP-NO-SEG-8	MobileNet	10	SGD	32	0.3	100%	NA	NA	NA	0.0322	98.81%	0.9879	98.79%	98.79%	0.9996	92.14%
EXP-NO-SEG-9	MobileNetV2	13	SGD	32	0.3	100%	NA	NA	NA	0.0921	97.73%	0.9771	97.74%	97.68%	0.9962	89.22%
EXP-NO-SEG-10	HMB1-COVID19	15	RMSProp	64	0.27	NA	ReLU	He Normal	L1(10^{-5})	0.4037	89.37%	0.8937	89.69%	89.05%	0.9863	81.61%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

Table 30
Tabular summary of the top-1 experiments without segmentation concerning the last reported results.

Experiment	Model	PO[a]	Batch size	DR[a]	TLR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
EXP-NO-SEG-1	VGG16	SGD	32	0.3	55%	NA	NA	NA	0.0099	99.84%	0.9984	99.84%	99.84%	0.9996	99.92%
EXP-NO-SEG-2	VGG19	SGD	32	0.0	85%	NA	NA	NA	0.0139	99.52%	0.9952	99.52%	99.52%	1.000	96.81%
EXP-NO-SEG-3	ResNet50	Ftrl	32	0.4	10%	NA	NA	NA	1.7133	26.85%	0.2839	37.29%	23.09%	0.5558	27.92%
EXP-NO-SEG-4	ResNet101	SGD	32	0.0	100%	NA	NA	NA	0.0214	99.73%	0.9973	99.73%	99.73%	0.9985	94.43%
EXP-NO-SEG-5	Xception	AdaGrad	64	0.1	80%	NA	NA	NA	0.5740	84.46%	0.8459	84.97%	84.23%	0.9610	77.40%
EXP-NO-SEG-6	DenseNet121	AdaGrad	64	0.2	100%	NA	NA	NA	0.0334	99.03%	0.9903	99.03%	99.03%	0.9988	92.20%
EXP-NO-SEG-7	DenseNet169	AdaGrad	64	0.5	100%	NA	NA	NA	0.0157	99.62%	0.9962	99.62%	99.62%	0.9993	96.05%
EXP-NO-SEG-8	MobileNet	SGD	32	0.5	100%	NA	NA	NA	0.0610	98.59%	0.9858	98.58%	98.58%	0.9973	90.48%
EXP-NO-SEG-9	MobileNetV2	SGD	32	0.3	100%	NA	NA	NA	0.1243	97.02%	0.9699	97.04%	96.94%	0.9934	88.35%
EXP-NO-SEG-10	HMB1-COVID19	RMSProp	64	0.27	NA	ReLU	He normal	L1(10^{-5})	0.4037	89.37%	0.8937	89.69%	89.05%	0.9863	81.61%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

Table 31
Tabular summary of the top-1 experiments concerning the highest WS.

Experiment	Iteration	Model	PO[a]	Batch size	DR[a]	TLR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
EXP-NO-SEG-1	VGG16	13	SGD	32	0.5	80%	NA	NA	NA	0.0097	99.78%	0.9984	99.89%	99.78%	0.9996	100.0%
EXP-NO-SEG-2	VGG19	5	SGD	64	0.2	85%	NA	NA	NA	0.0107	99.62%	0.9962	99.62%	99.62%	0.9996	99.01%
EXP-NO-SEG-7	DenseNet169	9	SGD	64	0.2	100%	NA	NA	NA	0.0124	99.51%	0.9951	99.51%	99.51%	0.9997	97.65%
EXP-NO-SEG-4	ResNet101	13	SGD	64	0.5	100%	NA	NA	NA	0.0157	99.68%	0.9968	99.68%	99.68%	0.9996	96.10%
EXP-SEG-2	VGG19	15	SGD	32	0.2	85%	NA	NA	NA	0.0170	99.46%	0.9947	99.47%	99.47%	0.9996	95.44%
EXP-SEG-1	VGG16	13	SGD	32	0.4	90%	NA	NA	NA	0.0176	99.57%	0.9958	99.58%	99.58%	0.9996	95.33%
EXP-SEG-5	Xception	15	AdaMax	32	0.3	100%	NA	NA	NA	0.0311	99.25%	0.9926	99.26%	99.26%	0.9989	92.61%
EXP-NO-SEG-6	DenseNet121	15	AdaGrad	64	0.2	100%	NA	NA	NA	0.0334	99.03%	0.9903	99.03%	99.03%	0.9988	92.20%
EXP-NO-SEG-8	MobileNet	10	SGD	32	0.3	100%	NA	NA	NA	0.0322	98.81%	0.9879	98.79%	98.79%	0.9996	92.14%
EXP-SEG-4	ResNet101	14	SGD	32	0.0	100%	NA	NA	NA	0.0663	97.59%	0.9759	97.66%	97.51%	0.9985	89.56%
EXP-SEG-6	DenseNet121	13	AdaMax	32	0.1	100%	NA	NA	NA	0.0814	97.69%	0.9772	97.72%	97.72%	0.9974	89.37%
EXP-NO-SEG-9	MobileNetV2	13	SGD	32	0.3	100%	NA	NA	NA	0.0921	97.73%	0.9771	97.74%	97.68%	0.9962	89.22%
EXP-SEG-8	MobileNet	10	Nadam	64	0.2	100%	NA	NA	NA	0.1047	96.14%	0.9621	96.29%	96.13%	0.9976	87.86%
EXP-SEG-7	DenseNet169	12	Adam	64	0.3	100%	NA	NA	NA	0.1236	95.87%	0.9576	95.78%	95.73%	0.9955	87.43%
EXP-SEG-10	HMB1-COVID19	15	SGD	32	0.17	NA	ReLU	Glorot Normal	L1(10^{-5})	0.3124	91.38%	0.9140	91.77%	91.04%	0.9894	83.33%
EXP-SEG-9	MobileNetV2	12	SGD	64	0.3	100%	NA	NA	NA	0.4265	90.08%	0.9046	90.73%	90.20%	0.9726	82.14%
EXP-NO-SEG-10	HMB1-COVID19	15	RMSProp	64	0.27	NA	ReLU	He Normal	L1(10^{-5})	0.4037	89.37%	0.8937	89.69%	89.05%	0.9863	81.61%
EXP-NO-SEG-5	Xception	12	AdaGrad	64	0.1	80%	NA	NA	NA	0.5690	85.22%	0.8508	85.29%	84.87%	0.9616	77.92%
EXP-NO-SEG-3	ResNet50	3	Adam	64	0.2	10%	NA	NA	NA	3.4394	37.57%	0.3668	40.29%	33.74%	0.5661	35.55%
EXP-SEG-3	ResNet50	6	AdaMax	64	0.2	10%	NA	NA	NA	2.3876	31.58%	0.3120	32.23%	30.26%	0.5429	30.63%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

Table 32
Tabular summary of the top-1 experiments concerning the last reported results.

Experiment	Model	PO[a]	Batch size	DR[a]	TLR[a]	HAF[a]	PI[a]	Regularizer	Loss	Accuracy	F1	Precision	Recall	AUC	WS
EXP-NO-SEG-1	VGG16	SGD	32	0.3	55%	NA	NA	NA	0.0099	99.84%	0.9984	99.84%	99.84%	0.9996	99.92%
EXP-NO-SEG-2	VGG19	SGD	32	0.0	85%	NA	NA	NA	0.0139	99.52%	0.9952	99.52%	99.52%	1.000	96.81%
EXP-NO-SEG-7	DenseNet169	AdaGrad	64	0.5	100%	NA	NA	NA	0.0157	99.62%	0.9962	99.62%	99.62%	0.9993	96.05%
EXP-SEG-2	VGG19	SGD	32	0.2	85%	NA	NA	NA	0.0170	99.46%	0.9947	99.47%	99.47%	0.9996	95.44%
EXP-NO-SEG-4	ResNet101	SGD	32	0.0	100%	NA	NA	NA	0.0214	99.73%	0.9973	99.73%	99.73%	0.9985	94.43%
EXP-SEG-5	Xception	AdaMax	32	0.3	100%	NA	NA	NA	0.0311	99.25%	0.9926	99.26%	99.26%	0.9989	92.61%
EXP-NO-SEG-6	DenseNet121	AdaGrad	64	0.2	100%	NA	NA	NA	0.0334	99.03%	0.9903	99.03%	99.03%	0.9988	92.20%
EXP-SEG-1	VGG16	SGD	32	0.1	90%	NA	NA	NA	0.0372	99.30%	0.9931	99.31%	99.31%	0.9981	92.12%
EXP-NO-SEG-8	MobileNet	SGD	32	0.5	100%	NA	NA	NA	0.0610	98.59%	0.9858	98.58%	98.58%	0.9973	90.48%
EXP-SEG-4	ResNet101	AdaGrad	32	0.0	100%	NA	NA	NA	0.0936	97.10%	0.9711	97.14%	97.09%	0.9957	88.71%
EXP-NO-SEG-9	MobileNetV2	SGD	32	0.3	100%	NA	NA	NA	0.1243	97.02%	0.9699	97.04%	96.94%	0.9934	88.35%
EXP-SEG-8	MobileNet	AdaMax	32	0.3	100%	NA	NA	NA	0.2108	94.53%	0.9432	94.42%	94.22%	0.9878	85.91%
EXP-SEG-6	DenseNet121	SGD	64	0.1	100%	NA	NA	NA	0.1963	93.99%	0.9388	93.93%	93.83%	0.9922	85.59%
EXP-SEG-7	DenseNet169	Adam	64	0.3	100%	NA	NA	NA	0.2161	93.67%	0.9360	93.80%	93.40%	0.9892	85.27%
EXP-SEG-10	HMB1-COVID19	SGD	32	0.17	NA	ReLU	Glorot Normal	L1(10^{-5})	0.3124	91.38%	0.9140	91.77%	91.04%	0.9894	83.33%
EXP-NO-SEG-10	HMB1-COVID19	RMSProp	64	0.27	NA	ReLU	He Normal	L1(10^{-5})	0.4037	89.37%	0.8937	89.69%	89.05%	0.9863	81.61%
EXP-SEG-9	MobileNetV2	SGD	64	0.2	100%	NA	NA	NA	0.4998	86.70%	0.8641	86.58%	86.25%	0.9685	79.16%
EXP-NO-SEG-5	Xception	AdaGrad	64	0.1	80%	NA	NA	NA	0.5740	84.46%	0.8459	84.97%	84.23%	0.9610	77.40%
EXP-NO-SEG-3	ResNet50	Ftrl	32	0.4	10%	NA	NA	NA	1.7133	26.85%	0.2839	37.29%	23.09%	0.5558	27.92%
EXP-SEG-3	ResNet50	AdaMax	32	0.1	20%	NA	NA	NA	4.0317	28.10%	0.2781	27.84%	27.78%	0.5411	27.83%

[a] PO: Parameters Optimizer, DR: Dropout Ratio, TLR: TF Learn Ratio, HAF: Hidden Activation Function, PI: Parameters Initializer.

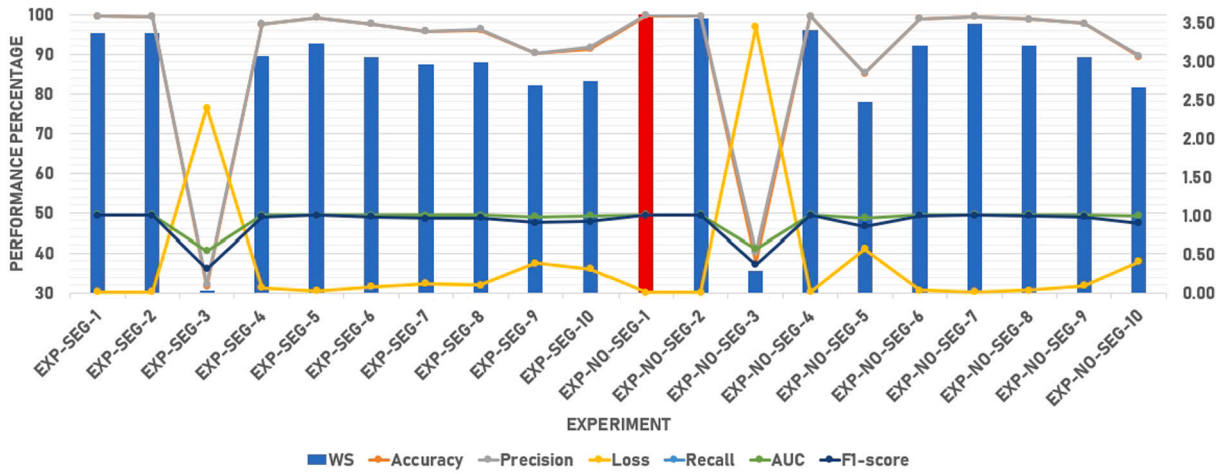


Fig. 31. Overall reported top-1 experiments graphical summary concerning the highest WS metrics.

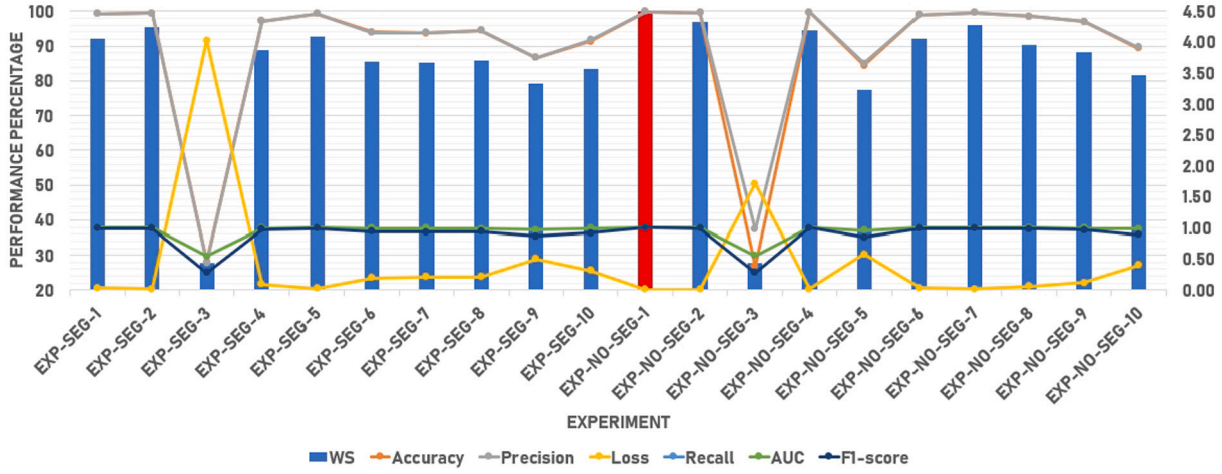


Fig. 32. Overall reported top-1 experiments graphical summary concerning the last reported results.

iterations. The best-achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0157, 99.73%, 0.9973, 99.73%, 99.73%, 0.9998, and 96.10% respectively while the last reported iteration metrics were 0.0214, 99.73%, 0.9973, 99.73%, 99.73%, 0.9985, and 94.43% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had negligible correlation (−0.07). The dropout ratio and TF learn ratio correlation could not be determined. Fig. 24 shows the EXP-NO-SEG-4 WS curve for the 15 iterations.

Table 23 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the Xception pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 23, the AdaGrad optimizer was the best in 14 iterations. The 64 batch size was the best in 14 iterations. The 10% dropout ratio was the best in 8 iterations. The 80% TF learn ratio was the best in 15 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.5521, 85.22%, 0.8508, 85.29%, 84.87%, 0.9616, and 77.92% respectively while the last reported iteration metrics were 0.5740, 84.46%, 0.8459, 84.97%, 84.23%, 0.9610, and 77.40% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had a high negative correlation (−0.80). The dropout ratio and TF learn ratio correlation could not be determined. Fig. 25 shows the EXP-NO-SEG-5 WS curve for the 15 iterations.

Table 24 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the DenseNet121 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 24, the AdaMax optimizer was the best in 7 iterations. The 64 batch size was the best in 11 iterations. The 20% dropout ratio was the best in 12 iterations. The 100% TF learn ratio was the best in 8 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0334, 99.03%, 0.9903, 99.03%, 99.03%, 0.9994, and 92.20% respectively while the last reported iteration metrics were 0.0334, 99.03%, 0.9903, 99.03%, 99.03%, 0.9988, and 92.20% respectively. The WS and TF learn ratio had a very high positive correlation (0.94) while the WS and dropout ratio had a low negative correlation (−0.38). The dropout ratio and TF learn ratio had a moderate negative correlation (−0.57). Fig. 26 shows the EXP-NO-SEG-6 WS curve for the 15 iterations.

Table 25 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the DenseNet169 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64

Table 33
Generalization validation results on the available online lung X-ray images datasets.

No.	Dataset	Used images count	Correct predictions count	Prediction accuracy
1	"Pneumonia (virus) vs COVID-19" [50]	70	69	98.57%
2	"Covid-19 Xray images using CNN" [51]	284	280	98.59%
3	"COVID-19 X-ray Images5" [52]	1443	1437	99.58%
4	"COVID-19 Patients Lungs X Ray Images 10,000" [53]	98	97	98.98%
5	"COVID-19 Chest X Rays" [54]	148	145	97.97%
6	"Chest X-ray (Covid-19 & Pneumonia)" [57]	2159	2132	98.75%
7	"COVID-19 Xray Dataset (Train & Test Sets)" [59]	94	93	98.94%
8	"COVID19-xray" [60]	1161	1133	97.59%
9	"Chest Xray for covid-19 detection" [61]	347	343	98.85%
10	"COVID-19 & Normal Posteroanterior(PA) X-rays" [64]	279	277	99.28%
11	"Covid-GAN and Covid-Net mini Chest X-ray" [66]	3054	2528	82.78%
12	"Chest X-ray Images" [68]	1583	1563	98.74%
13	"Chest Xray Images PNEUMONIA and Covid-19" [69]	1845	1814	98.32%

epochs.

From Table 25, the AdaGrad optimizer was the best in 14 iterations. The 64 batch size was the best in 11 iterations. The 20% dropout ratio was the best in 9 iterations. The 100% TF learn ratio was the best in 15 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0124, 99.73%, 0.9973, 99.73%, 99.73%, 0.9997, and 97.65% respectively while the last reported iteration metrics were 0.0157, 99.62%, 0.9962, 99.62%, 99.62%, 0.9993, and 96.05% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had a moderate positive correlation (0.51). The dropout ratio and TF learn ratio correlation could not be determined. Fig. 27 shows the EXP-NO-SEG-7 WS curve for the 15 iterations.

Table 26 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the MobileNet pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 26, the SGD optimizer was the best in 14 iterations. The 32 batch size was the best in 15 iterations. The 30% dropout ratio was the best in 11 iterations. The 100% TF learn ratio was the best in 15 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0322, 98.97%, 0.9898, 98.98%, 98.98%, 0.9996, and 92.14% respectively while the last reported iteration metrics were 0.0610, 98.59%, 0.9858, 98.58%, 98.58%, 0.9973, and 90.48% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had a low positive correlation (0.10). The dropout ratio and TF learn ratio correlation could not be determined. Fig. 28 shows the EXP-NO-SEG-8 WS curve for the 15 iterations.

Table 27 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the MobileNetV2 pre-trained CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 27, the SGD optimizer was the best in 7 iterations. The 32 batch size was the best in 13 iterations. The 30% dropout ratio was the best in 8 iterations. The 100% TF learn ratio was the best in 15 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.0921, 97.73%, 0.9771, 97.74%, 97.68%, 0.9962, and 89.22% respectively while the last reported iteration metrics were 0.1243, 97.02%, 0.9699, 97.04%, 96.94%, 0.9934, and 88.35% respectively. The WS and TF learn ratio correlation could not be determined while the WS and dropout ratio had a low positive correlation (0.38). The dropout ratio and TF learn ratio correlation could not be determined. Fig. 29 shows the EXP-NO-SEG-9 WS curve for the 15 iterations.

Table 28 reports the top-1 combination in each hyperparameters' optimization iteration in the 15 iterations for the proposed HMB1-COVID19 CNN model with the corresponding testing performance metrics. Each iteration has 10 solutions where each solution is trained for 64 epochs.

From Table 28, the RMSProp optimizer was the best in 9 iterations. The 32 batch size was the best in 13 iterations. The 47% dropout ratio was the best in 5 iterations. The ReLU hidden activation was the best in 8

Table 34
Comparison between the current study and other related studies.

Study	Year	Dataset size	Dataset type	Approach	Best metric
Bukhari et al. [27]	2020	278	X-ray	CNN + TF	98.18% accuracy
Gozes et al. [31]	2020	270	CT	2D and 3D analysis	0.996 AUC
Apostolopoulos et al. [32]	2020	1428	X-ray	CNN + TF	96.78% accuracy
Chowdhury et al. [35]	2020	3487	X-ray	CNN + TF	98.30% accuracy
Abbas et al. [38]	2020	1764	X-ray	DeTraC	93.10% accuracy
Wang et al. [39]	2020	453	CT	CNN + TF	73.10% accuracy
Abraham et al. [41]	2020	960 and 78	X-ray	CNN + CFS	91.16% and 97.44% accuracies
Islam et al. [42]	2020	4575	X-ray	CNN + LSTM	99.40% accuracy
Polsinelli et al. [44]	2020	460	CT	CNN + TF	85.03% accuracy
Aslan et al. [45]	2020	142	X-ray	MLP + CNN	96.30% accuracy
Bahgat et al. [46]	2021	12,933	CT	OTLD-COVID-19	98.47% accuracy
Jain et al. [47]	2021	6432	X-ray	CNN + TL	97.97% accuracy
Wang et al. [48]	2021	1065	CT	CNN + TL	89.50% accuracy
Current study	2021	13,711	X-ray	HMB-HCF	100% and 99.92% WS

iterations. The He Normal parameters initializer was the best in 8 iterations. The L1(10^{-5}) regularizer was the best in 12 iterations. The best achieved metrics for loss, accuracy, F1-score, precision, recall, AUC, and WS were 0.4037, 89.37%, 0.8937, 89.69%, 89.05%, 0.9863, and 81.61% respectively while the last reported iteration metrics were 0.4037, 89.37%, 0.8937, 89.69%, 89.05%, 0.9863, and 81.61% respectively. The WS and dropout ratio had a moderate negative correlation (−0.63). Fig. 30 shows the EXP-NO-SEG-10 WS curve for the 15 iterations.

Table 29 summarizes the obtained top-1 results in each experiment without segmentation concerning the highest WS metrics. Table 30 summarizes the obtained top-1 results in each experiment without segmentation concerning the last reported results.

4.4. Experiments summarization, generalization validation and comparisons. Experiments summarization

Table 31 summarizes the obtained top-1 results in each experiment concerning the highest WS metrics. It shows that "EXP-NO-SEG-1" experiment reports the highest WS score. Table 32 summarizes the obtained top-1 results in each experiment concerning the last reported results. Both tables are sorted from the highest WS to the lowest. It shows that "EXP-NO-SEG-1" experiment reports the highest WS score.

Both tables show the maximum reported accuracy is 99.84%, the maximum reported F1-score is 0.9984, the maximum reported precision is 99.89%, the maximum reported recall is 99.84%, the maximum reported AUC is 1 (or 100%) and the minimum reported loss is 0.0097 throughout all experiments. Fig. 31 summarizes the obtained top-1 results in each experiment concerning the highest WS metrics. Fig. 32 summarizes the obtained top-1 results in each experiment concerning the last reported results.

4.4.1. Generalization validation

To validate and verify the generalization of the suggested hybrid COVID-19 model, 13 available datasets, from Table 2, are evaluated and the corresponding results are reported in Table 33. It reports the number of correctly predicted images and the corresponding prediction accuracy. The prediction accuracy is calculated by dividing the correct predictions' count by the used images' count.

It is worth mentioning that the available datasets categories that are outside the current study used categories (i.e., "Normal", "Pneumonia-Viral", "Pneumonia-Bacterial", and "COVID-19") such as "Pneumonia" and "Lung Opacity" are ignored.

4.4.1.1. Related studies comparison. The current study's best-reported performance metrics are compared with some of the COVID-19 related studies. The comparison is presented in Table 34. The table is sorted from the oldest to the latest by the year.

5. Conclusions and future work

After the rapid spread of the COVID-19 (Coronavirus), computer science engineers shared in the development of the automatic diagnosis of that disease aiming at reaching an early and accurate diagnosis to face any future spread of that virus. The current study proposed a hybrid COVID-19 framework named HMB-HCF. It contained nine phases that were discussed in detail. A lung segmentation algorithm using X-Ray images named HMB-LSAXI was suggested in the third phase. An abstract CNN model named HMB1-COVID19 was designed and discussed in the sixth phase. It was cascaded in three blocks where each block contained convolutional and pooling layers. Batch normalization, regularization, dropout, and data augmentation were used in the current study. A combined deep learning and genetic algorithm overall algorithm for learning and optimization named HMB-DLGA was suggested in the ninth phase. A hybrid hierarchy model using HMB1-COVID19 and transfer learning pre-trained models was proposed also in the ninth phase. Extensive experiments were applied in the current study and the state-of-the-art performance metrics were reported. The used dataset was collected, filtered, and unified from 8 different sources. The proposed HMB-LSAXI algorithm reported a segmentation success ratio of 53.30%. The weighted sum (WS) approach using loss, accuracy, F1-score, precision, recall, and AUC metrics with different ratios were used to evaluate the models' performance in the different experiments. The experiments with (and without) segmentation used VGG16, VGG19, ResNet50, ResNet101, Xception, DenseNet121, DenseNet169, MobileNet, and MobileNetV2 pre-trained models besides the proposed HMB1-COVID19 model. Nine models achieved WS metrics above 90%. The best model was VGG16 as it reported a WS value of 99.92% while the worst model was ResNet50 as it reported only 27.73% concerning the last reported results. The hybrid hierarchy model was verified and validated on unseen datasets and it reported state-of-the-art prediction accuracies. Also, a comparison between the current study and other related studies was applied. This concludes the applicability of the suggested approach. In future work, more experiments can be applied using different hyperparameters optimizers such as Harris-Hawks Optimization or Manta Ray Foragining Optimizer. The segmentation algorithm can be improved to segment the unbalanced and low-resolution images. CT datasets along with X-Ray ones can be used in the hybrid approach for more accurate and precise results.

Source code

The source code is available at GitHub: https://github.com/HossamBalaha/Hybrid-COVID-19-Segmentation-and-Recognition-Framework-HMB-HCF.

Funding

No funding was received for this work (i.e., study).

Intellectual property

We confirm that, we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, concerning intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

Research ethics

We further confirm, if existing, that any aspect of the work covered in this manuscript that has involved human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript. Written consent to publish potentially identifying information, such as details of the case and photographs, was obtained from the patient(s) or their legal guardian(s).

Data availability

The datasets, if existing, that are used, generated, or analyzed during the current study (A) if the datasets are owned by the authors, they are available from the corresponding author on reasonable request, (B) if the datasets are not owned by the authors, the supplementary information including the links and sizes are included in this published article.

Compliance with ethical standards

The current study does not contain any studies with human participants and/or animals performed by any of the authors.

Consent to participate

There is no informed consent for the current study.

Consent for publication

Not applicable.

Contact with the editorial office

The "Corresponding Author" who is declared on the title page. This author submitted this manuscript using his account in the editorial submission system. (A) We understand that the "Corresponding Author" is the sole contact for the Editorial process (including the editorial submission system and direct communications with the office). He is responsible for communicating with the other authors about progress, submissions of revisions, and final approval of proofs. (B) We confirm that the email address shown below is accessible by the "Corresponding Author" and is the address to which "Corresponding Author" editorial submission system account is linked and has been configured to accept email from the editorial office (email: hossam.m.balaha@mans.edu.eg).

CRediT authorship contribution statement

We, the undersigned authors, declare that this manuscript is original, has not been published before, and is not currently being considered for publication elsewhere. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. We understand that the "Corresponding Author" is the sole contact for the editorial process. He is responsible for communicating with the other authors about progress, submissions of revisions, and final approval of proofs.

Declaration of competing interest

No conflict of interest exists. We wish to confirm that, there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Appendix 1. Convolutional neural networks

A convolutional neural network (CNN), in the current study, is the core of the learning process and model construction. A CNN is constructed by cascading layers. It is used to analyze and learn from visual elements. The major advantage of using CNNs is that the features are extracted automatically by the selected CNN architecture [115]. Hence, there is no demand for the feature extraction manual and traditional ML techniques [116,117].

Basically, an image x from the X dataset is the input to a CNN (i.e., X-Ray lung image in our case). The CNN learns and extracts features automatically from X, which is named the training subset after the splitting phase. X is an array of images (Eq. (16)) where an image x has a size shown in Eq. (17).

$$X = [x_1, \ldots, x_r, \ldots, x_{m-1}, x_m] \tag{16}$$

$$\text{Size}(x_r) = (w, h, ch) \tag{17}$$

where x_r is the image at an index r and m is the number of images. Commonly, w and h are equal. Different types of layers can be used with CNNs such as the convolutional layer, the pooling layer, the fully-connected (FC) layer, the activation layer, and the dropout layer [118].

Convolutional layer

A convolutional (Conv. for short) layer applies filters for feature extraction. Each layer can represent a specific image feature extractor such as horizontal edges [119]. A single convolutional layer can contain n_c filters where each of them has a size shown in Eq. (18).

$$\text{Size}\left(\text{conv}_{\text{filter}}\right) = (w_c, h_c, ch_c) \tag{18}$$

where w_c is the convolutional filter width, h_c is the convolutional filter height, and ch_c is the number of convolutional filter's channels (ch_c is equal to or less than ch). Commonly, w_c and h_c are equal. Any convolutional layer has two main factors, padding, and stride. Padding is the process of adding zeros on the boundaries of the input matrix symmetrically. This will help to maintain the output dimension similar to the input dimension. Stride is the number of added steps while we are moving in each direction. The stride is one by default [120]. The convolutional layer output size can be computed from Eq. (19).

$$\text{Size}(\text{conv}) = \left(\lfloor \frac{w_{in} - w_c + 2 \times p_c}{s_{cw}} \rfloor + 1 \, , \lfloor \frac{h_{in} - h_c + 2 \times p_c}{s_{ch}} \rfloor + 1 \, , n_c \right) \tag{19}$$

where w_{in} is the input width, h_{in} is the input height, s_{cw} is the convolutional stride width, s_{ch} is the convolutional stride height, and p_c is the convolutional padding size. Commonly, s_{cw} and s_{ch} are equal. For example, if the input is (32, 32, 3) and the convolutional kernel is sized (5, 5, 3) with no padding, stride size (1, 1), and 10 filters, then the convolution output size will be (28, 28, 10). Fig. 33 illustrates the convolution layer process graphically.

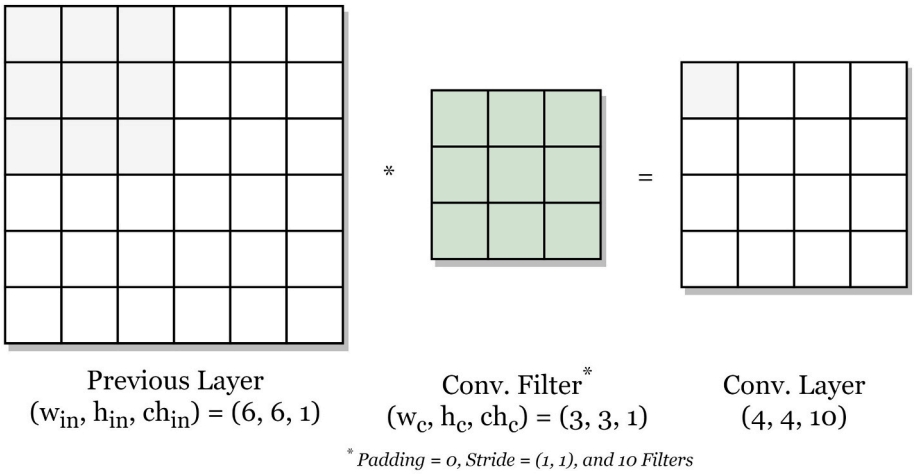


Fig. 33. Graphical illustration of the convolution layer process.

Pooling layer

Pooling layers can follow the convolution layers for dimensionality reduction. They will scale down and keep the most important features from the previous convolutional layer according to the pooling layer type. Pooling layer types include max-, average-, min-, and sum- pooling. The commonly used pooling layer is the max-pooling layer [121]. The kernel size of a pooling layer is shown in Eq. (20).

$$\text{Size}(\text{pool}_{\text{kernel}}) = (w_p, h_p) \tag{20}$$

where w_p is the pool filter width and h_p is the pool filter height. Commonly, w_p and h_p are equal. Similar to convolutional layers, a pooling layer has padding and stride. The pooling layer output size can be computed from Eq. (21).

$$\text{Size}(\text{pool}) = \left(\lfloor \frac{w_{in} - w_p + 2 \times p_p}{s_{pw}} \rfloor + 1 \, , \lfloor \frac{h_{in} - h_p + 2 \times p_p}{s_{ph}} \rfloor + 1 \, , n_{in} \right) \tag{21}$$

where n_{in} is the input number of filters, s_{pw} is the pooling stride width, s_{ph} is the pooling stride height, and p_p is the pool padding size. Commonly, s_{pw} and s_{ph} are equal. For example, if the input is (28,28,10) and the pooling kernel is sized (2, 2) with no padding and stride size (2, 2), then the pooling output size will be (14,14,10). Fig. 34 illustrates the pooling layer process graphically.

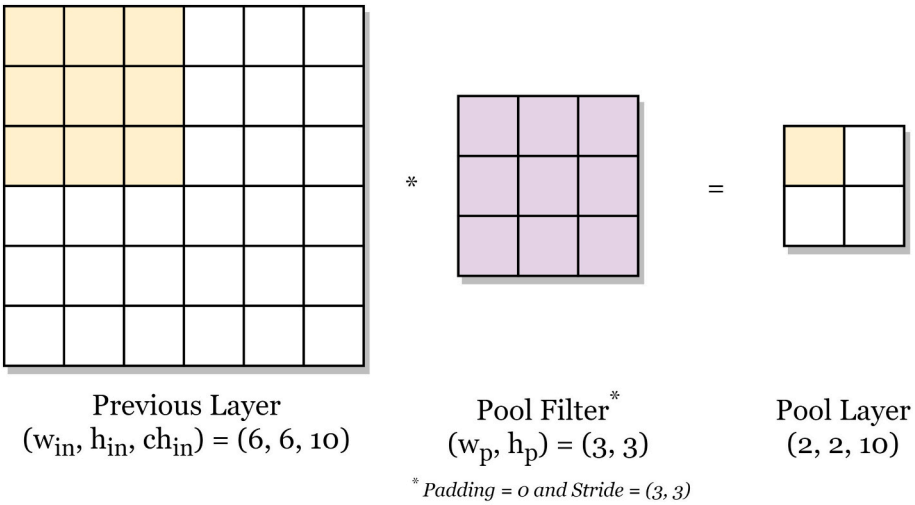


Fig. 34. Graphical illustration of the pooling layer process.

Fully-connected (FC) layer

FC layers exist after cascading the convolutional and pooling layers. An FC layer connects all of the previous layer neurons to every single neuron it has (i.e., flattens the previous multi-dimensional layer's neurons). No more spatial information can be extracted in the FC layers. Hence, there can be no more convolutional layers after an FC layer [122]. Multiple FC layers with different numbers of neurons can be cascaded. The output layer is also an FC layer but its number of neurons is equal to the number of categories (i.e., classes) in the applied task.

Activation function layer

The activation function layer determines whether each neuron should be activated or not. The major advantage of it is the ability to provide non-linearity into the neural network [123]. They are added in the hidden layers such as convolutional and FC layers. There are different activation functions such as Sigmoid and ReLU [124].

ReLU is a widely used piece-wise linear function that will output the input value directly if it is positive and zero otherwise. It has three main advantages (1) easy to compute and (2) does not saturate, and (3) does not fall in the vanishing gradient problem [125]. But it suffers from the dying ReLU problem [126] which is solved by the Leaky ReLU activation function [127]. Eqs. (22) and (23) show the computational formulas for the ReLU and Leaky ReLU respectively.

$$\text{ReLU}(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{Otherwise} \end{cases} \tag{22}$$

$$\text{LeakyReLU}(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha \times z & \text{Otherwise} \end{cases} \tag{23}$$

where z is an input value and α is a small selected value (i.e., 0.3).

The SoftMax function is a generalization of the logistic regression binary form. It is widely used for multi-class classification tasks [128]. It is used to convert an array of values into probabilities, where each value is proportional to a relative scale of each value in the array as shown in Eq. (24). The lower the value of it, the more favorable [129].

$$\text{SoftMax}(Z)_r = \frac{e^{z_r}}{\sum_{t=1}^{N} e^{z_t}} \tag{24}$$

where Z is the input array of values, z_r is the value at index r, and z_t is the value at another index t. Table 35 summarizes the equations of the used activation functions.

Table 35
Summarization of the used activation functions.

Function	Equation
Exponential linear unit (ELU)	$f(z) = \begin{cases} \alpha \times (e^z - 1), & \text{if } z < 0. \\ z, & \text{Otherwise.} \end{cases}$
Exponential	$f(z) = e^z$
Rectified linear unit (ReLU)	$f(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{Otherwise.} \end{cases}$
Scaled exponential linear unit (SELU)	$f(z) = \begin{cases} scale \times \alpha \times (e^z - 1), & \text{if } z < 0 \\ scale \times z, & \text{Otherwise.} \end{cases}$
Sigmoid	$f(z) = \frac{1}{1 + e^{-z}}$
Hyperbolic tangent (Tanh)	$f(z) = \frac{2}{1 + e^{(-2 \times z)}} - 1$

Parameters optimization

Parameters (i.e., weights) optimization algorithms and techniques help the models generalize, achieve better performance metrics, and reach the global minimum faster [130]. Gradient descent algorithm is the default and traditional approach to optimize the learnable parameters of neural networks and many other ML algorithms [131]. It takes small steps in the negative gradient direction of the loss (i.e., error) function as shown in Eq. (25).

$$\theta_{u+1} = \theta_u - \eta \times E(\theta_u) \tag{25}$$

where θ is the parameters, u is the iteration number, η is the learning rate, and E is the error function. The standard gradient descent algorithm is applied to the entire dataset. The learning rate, η, is a hyperparameter that controls the adjusting step. The lower the learning rate value, the slower the move along the downward slope, and the more time it required.

There are parameters optimization techniques that overcome the gradient descent drawbacks such as Adaptive Moment Optimization Algorithm (Adam), AdaGrad, AdaDelta, AdaMax, RMSProp, Nesterov-accelerated Adaptive Moment Estimation (Nadam), and Follow-the-regularized-leader (Ftrl).

Adagrad adapts the learning rate to the parameters in two ways (1) applies smaller updates for the parameters that are associated with frequent features, and (2) applies larger updates for the parameters that are associated with infrequent features. Hence, it is suitable to handle sparse data

[132].

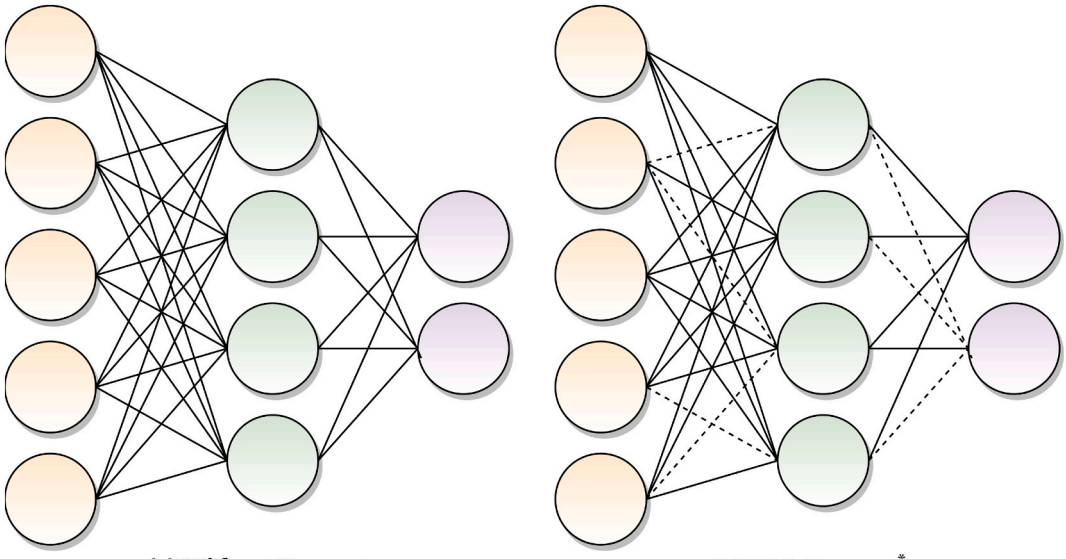
Adadelta is a derivative of the Adagrad. It seeks to reduce the Adagrad monotonically and aggressive decreasing learning rate. Adadelta applies a fixed size window to accumulate the past squared gradients instead of accumulating all of them [133].

RMSProp and Adadelta were developed independently to resolve the Adagrad's radically diminishing learning rates [134]. Adam [34] computes adaptive learning rates for each parameter. It combines the Momentum [135] and RMSProp heuristics as they apply contrasting approaches. The Momentum accelerates the search in the minima direction the RMSProp impedes the search in the direction of the oscillations.

AdaMax exhibits stable behavior [136]. Nadam combines Adam and Nesterov Accelerated Gradient (NAG) optimizers [137]. Ftrl gives better performance compared with the sparsity trade-off [138].

Dropout layer

Dropout is an important parameters regularization technique. It is the process of dropping some of the neurons of the preceding layer to prevent overfitting on the training data and achieve better generalization [139]. Selecting a very high dropout ratio will decrease the number of parameters and may lead to a decrease in the performance metrics [29]. Fig. 35 shows a graphical illustration of the dropout process.



* *Dashed Lines: The removed connections because of the dropout process.*
Solid Lines: The remaining connections.

Fig. 35. Graphical illustration of the dropout process.

Batch normalization (BN) layer

BN is used to standardize the inputs, stabilize the parameters learning process, and reduce the internal covariate shift. Standardization means rescaling the parameters to have a mean value of zero and a standard deviation value of one [140]. Eq. (26) shows the standardization method.

$$\text{Data}_{\text{Standardized}} = \frac{\text{Data} - \text{Mean}}{\text{Standard Deviation}} \tag{26}$$

Regularization

Regularization is a technique that modifies the learning algorithm slightly. This can improve the model performance and generalizes better as well on the unseen data. L1 and L2 are the widely used regularization types [141].

Appendix 2. Genetic algorithms (GAs)

The genetic algorithms (GAs) major advantages can be (1) it supports multi-objective optimization tasks, (2) it handles noisy environments, (3) it merges different combinations to boost the best one, (4) it is used to solve many noisy and stochastic problems effectively, and (5) simpler to implement compared to other techniques that perform the same task [142–144].

GA steps include the creation of the initial population, fitness function calculation, sorting, selection, crossover, and mutation. After a set of iterations, the final and best chromosome with the highest fitness function value is reported upon completion. GA steps can be summarized as follows [145,146]:

Step 1: The GA starts by initiating a set of initial solutions which is called the "population". Each solution is called a chromosome and is characterized by a set of values called "genes".

Step 2: Start working on each solution by iterating on them or applying parallel processing. For the retrieved solution, a fitness function is used to determine how fit it is. It takes a fitness score to compete with others. The chance that a solution will be selected or not for the next generation is based on its fitness score (i.e., solutions with high fitness scores will have higher probabilities to be selected for the next generation).

Step 3: The selection step takes place after computing all of the fitness scores. The fittest solutions will remain and pass their genes to the next generation. This can occur by sorting them according to their fitness scores in descending order and selecting the first two solutions or half of them. The number of selected solutions depends on the task itself.

Step 4: Crossover is considered the most significant step in a GA. First, a crossover point is selected randomly or pre-defined. Second, offspring are created by swapping the genes of parent solutions among themselves. Finally, the new offspring are added to the new population of the next generation.

Step 5: Some of the new solutions' genes are subjected to a mutation with a low random probability. This denotes that some of the genes in the solution can be changed. The mutation process prevents premature convergence and maintains diversity within the population. It is preferred to use a low-rate mutation to avoid any search space randomness.

Step 6: Repeat Step 2 until convergence (i.e., offspring are not significantly different from the previous generation) or repeat for a set of iterations if the task does not focus on the convergence.

Step 7: Upon completion, the required solutions are reported.

Fig. 36 illustrates the GA steps graphically.

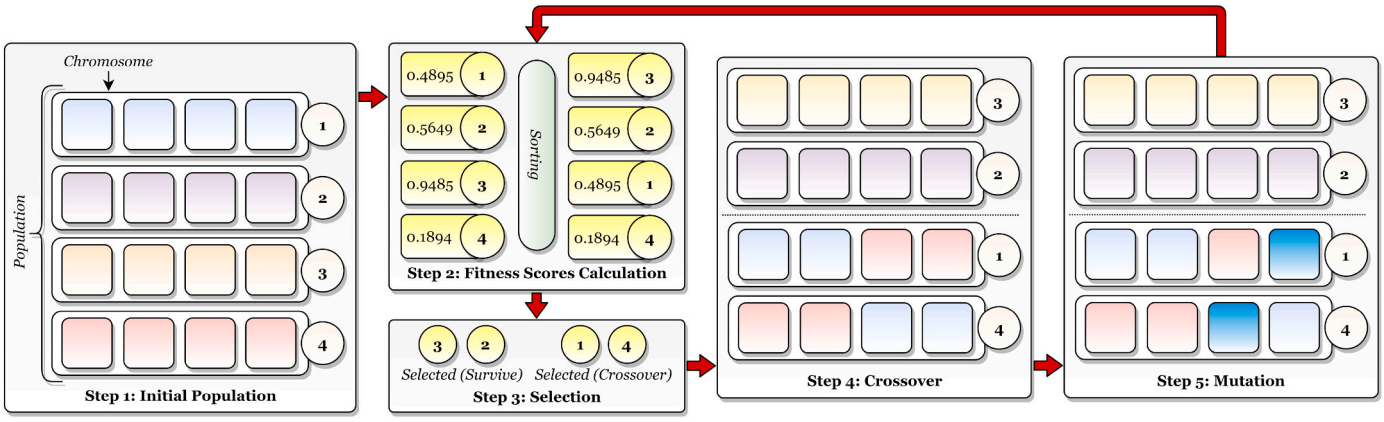


Fig. 36. Graphical illustration of the genetic algorithm (GA) steps.

Appendix 3. Table of abbreviations

Table 36 presents the "Table of abbreviations" used in the current study and it is ordered in ascending order.

Table 36
Table of abbreviations.

Abbreviation	Description
ACGAN	Auxiliary classifier generative adversarial network
Adam	Adaptive moment optimization algorithm
AI	Artificial intelligence
ANN	Artificial neural network
AUC	Area under curve
BiGAN	Bidirectional generative adversarial network
BN	Batch normalization
CCGAN	Context-conditional generative adversarial network
CFS	Correlation-based feature selection
CS	Computer science
CSV	Comma separated values
CT	Computerized tomography
CV	Computer vision
CNN	Convolutional neural network
COVID-19	Coronavirus 19
DA	Data augmentation
DBN	Deep belief network
DL	Deep learning
ELU	Exponential linear unit
FC	Fully-connected
Ftrl	Follow-the-regularized-leader
GA	Genetic algorithm
GAN	Generative adversarial network

(*continued on next page*)

Table 36 (*continued*)

Abbreviation	Description
LSTM	Long short-term memory
MERS	Middle east respiratory syndrome
ML	Machine learning
NA	Non-applicable
Nadam	Nesterov-accelerated adaptive moment estimation
NAG	Nesterov accelerated gradient
NP-hard	Non-deterministic polynomial-time hard
PR	Pattern recognition
ReLU	Rectified linear unit
RGB	Red green blue
RNN	Recurrent neural network
ROI	Region of interest
SARS	Severe acute respiratory syndrome
SELU	Scaled exponential linear unit
Tanh	Hyperbolic tangent
TF	Transfer learning
WHO	World health organization
WS	Weighted sum

Appendix 4. Table of symbols

Table 37 presents the "table of symbols" used in the current study.

Table 37
Table of symbols.

Symbol	Description	Symbol	Description
N	The number of classes (i.e., categories)	p_p	The pooling padding size
c	A specific class (i.e., category) where $c \in C$	s_{pw}	The pooling stride width
C	The used classes (i.e., categories) where $count(C) = N$	s_{ph}	The pooling stride height
ch	The number of the image channels	n_{in}	The filter input filters count
R, G, B	Red, Green, and Blue respectively	z	An input value
G_r	The resultant grayscale pixel value	α	A small value for the Leaky ReLU activation function
π	The PI value (i.e., 3.14 or $\frac{22}{7}$)	Z	Array of inputs
σ	The standard deviation	r	The iteration number
x_k	The x-coordinates of the kernel matrix K	θ	The parameters (i.e., weights)
y_k	The y-coordinates of the kernel matrix K	η	The learning rate
w	The image width	E	The error function
h	The image height	TP	True Positive
i_k	The kernel row index	TN	True Negative
j_k	The kernel column index	FP	False Positive
K	The kernel matrix	FN	False Negative
X_c	The images in a c class (i.e., category)	ε	A very small added value to avoid the division by zero
X_{ci}	The images in a ci class (i.e., category)	w	The weight used in the WS method such as w_1 and w_2
X	The images dataset	N_p	The size of the population
x	A single image	N_s	The number of genetic algorithm iterations
x_r	A single image at index r	N_c	The number of required combinations to be returned
m	The number of images	S_r	The split ratio
n_c	The number of convolutional filters	Y	The dataset labels
w_c	The convoltional filter width	O_s	The list of the used parameters optimizers
h_c	The convoltional filter height	W_s	The list of the used parameters initializers
ch_c	The convoltional filter number of channels	D_s	The list of the used dropout ratios
w_{in}	The filter input width	B_s	The list of the used batch sizes
h_{in}	The filter input height	L_s	The list of the used transfer learning learn ratios
p_c	The convolutional padding size	H_s	The list of the used hidden activation functions
s_{cw}	The convolutional stride width	M_s	The list of the used performance metrics
s_{ch}	The convolutional stride height	R_s	The list of the used regularizers
w_p	The pooling filter width	L	The number of returned categories (i.e., classes)
h_p	The pooling filter height	ch_{in}	The number of input channels

References

[1] Jiang F, et al. Review of the clinical characteristics of coronavirus disease 2019 (covid-19). J Gen Intern Med 2020:1–5.

[2] WHO. WHO director-general's opening remarks at the media briefing on COVID-19 - 16 March 2020. https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19-16-march-2020. [Accessed 4 January 2021].

[3] Worldometers. COVID-19 coronavirus pandemic. https://www.worldometers.info/coronavirus/. [Accessed 4 January 2021].

[4] Worldometers. Age, sex, existing conditions of COVID-19 cases and deaths. https://www.worldometers.info/coronavirus/coronavirus-age-sex-demographics/. [Accessed 4 January 2021].

[5] Worldometers. Coronavirus symptoms (COVID-19). https://www.worldometers.info/coronavirus/coronavirus-symptoms/#typical. [Accessed 4 January 2021].

[6] Wang D, et al. Clinical characteristics of 138 hospitalized patients with 2019 novel coronavirus–infected pneumonia in Wuhan, China. Jama 2020;323(11):1061–9.

[7] Al-Tawfiq JA, Zumla A, Memish ZA. Coronaviruses: severe acute respiratory syndrome coronavirus and middle east respiratory syndrome coronavirus in travelers. Curr Opin Infect Dis 2014;27(5):411–7.
[8] Chung M, et al. Ct imaging features of 2019 novel coronavirus (2019-ncov). Radiology 2020;295(1):202–7.
[9] Liu P, Xz Tan. 2019 novel coronavirus (2019-ncov) pneumonia. Radiology 2020; 295(1):19.
[10] Ali RMM, Ghonimy MBI. Radiological findings spectrum of asymptomatic coronavirus (covid-19) patients. Egypt J Radiol Nucl Med 2020;51(1):1–6.
[11] Shan F, et al. (2020) Lung infection quantification of covid-19 in ct images with deep learning. arXiv preprint arXiv:200304655.
[12] Bijl D, Hyde-Thomson H (2001) Speech to text conversion. US Patent 6,173,259.
[13] Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw 2015;61:85–117.
[14] Deng D, Liu H, Li X, Cai D. Pixellink: detecting scene text via instance segmentation in proceedings of the AAAI conference on artificial intelligenceVol. 32; 2018.
[15] Korns MF, May T. Strong typing, swarm enhancement, and deep learning feature selection in the pursuit of symbolic regression-classification in *genetic programming theory and practice XVI*. Springer; 2019. p. 59–84.
[16] Howard J, Ruder S. Universal Language Model Fine-tuning for Text Classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics; 2018. p. 328–39. https://doi.org/10.18653/v1/P18-1031.
[17] Wang Y, Xu W. Leveraging deep learning with lda-based text analytics to detect automobile insurance fraud. Decis Support Syst 2018;105:87–95.
[18] Chatterjee A, et al. Understanding emotions in text using deep learning and big data. Comput Hum Behav 2019;93:309–17.
[19] Guo Y, et al. Deep learning for visual understanding: a review. Neurocomputing 2016;187:27–48.
[20] Balaha HM, Ali HA, Badawy M. Automatic recognition of handwritten arabic characters: a comprehensive review. Neural Comput Applic 2021;33(7):3011–34.
[21] Balaha HM, Saafan MM. Automatic exam correction framework (aecf) for the mcqs, essays, and equations matching. IEEE Access 2021;9:32368–89.
[22] Sharma N, Jain V, Mishra A. An analysis of convolutional neural networks for image classification. Prog Comput Sci 2018;132:377–84.
[23] Aghdam HH, Heravi EJ. Guide to convolutional neural networksvol. 10. New York, NY: Springer; 2017. p. 978-973.
[24] Jamshidi M, et al. Artificial intelligence and covid-19: deep learning approaches for diagnosis and treatment. IEEE Access 2020;8:109581–95.
[25] Zeroual A, Harrou F, Dairi A, Sun Y. Deep learning methods for forecasting covid-19 time-series data: a comparative study. Chaos, Solitons Fractals 2020;140: 110121.
[26] Das R, Arshad M, Manjhi P, Thepade SD. Covid-19 identification with chest x-ray images merging handcrafted and automated features for enhanced feature generalization in. 2020 5th International Conference on Computing, Communication and Security (ICCCS) (IEEE) 2020:1–6.
[27] Bukhari Syed Usama Khalid, Bukhari Syed Safwan Khalid, Syed Asmara, Shah Syed Sajid Hussain. The diagnostic evaluation of Convolutional Neural Network (CNN) for the assessment of chest X-ray of patients infected with COVID-19. medRxiv 2020. https://doi.org/10.1101/2020.03.26.20044610. https://europepmc.org/article/PPR/PPR134117.
[28] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proc IEEE Conf Comput Vis Pattern Recognit 2016:770–8.
[29] Baldi P, Sadowski PJ. Understanding dropout. Adv Neural Inf Proces Syst 2013; 26:2814–22.
[30] Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. Journal of Big Data 2019;6(1):60.
[31] Gozes O, et al. (2020) Rapid ai development cycle for the coronavirus (covid-19) pandemic: initial results for automated detection & patient monitoring using deep learning ct image analysis. arXiv preprint arXiv:200305037.
[32] Apostolopoulos ID, Mpesiana TA. Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. In: Physical and engineering sciences in medicine; 2020. p. 1.
[33] Hara K, Saito D, Shouno H. Analysis of function of rectified linear unit used in deep learning in 2015 international joint conference on neural networks (IJCNN). IEEE; 2015. p. 1–8.
[34] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv. 2017, 1412.6980.
[35] Chowdhury Muhammad Enamul Hoque, Rahman Tawsifur, Khandakar Amith, Mazhar Rashid, Kadir Muhammad Abdul, Mahbub Zaid Bin, Islam Khandakar Reajul, Khan Muhammad Salman, Iqbal Atif, Al-Emadi Nasser, Reaz Mamun Bin Ibne. Can AI help in screening viral and COVID-19 pneumonia? CoRR 2020;abs/2003.13145. https://arxiv.org/abs/2003.13145. arXiv: 2003.13145, https://dblp.org/rec/journals/corr/abs-2003-13145.bib.
[36] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks in proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 4700–8.
[37] Iandola Forrest N, Moskewicz Matthew W, Ashraf Khalid, Han Song, Dally William J, Keutzer Kurt. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. CoRR 2016:1602.07360. http://arxiv.org/abs/1602.07360.
[38] Abbas Asmaa, Abdelsamea Mohammed M, Medhat Gaber Mohamed. Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. arXiv 2020:2003.13815.
[39] Wang S, et al. A deep learning algorithm using ct images to screen for corona virus disease (covid-19). MedRxiv; 2020.
[40] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. Proc IEEE Conf Comput Vis Pattern Recognit 2016:2818–26.
[41] Abraham B, Nair MS. Computer-aided detection of covid-19 from x-ray images using multi-cnn and bayesnet classifier. Biocyber Biomed Eng 2020;40(4): 1436–45.
[42] Islam MZ, Islam MM, Asraf A. A combined deep cnn-lstm network for the detection of novel coronavirus (covid-19) using x-ray images. Informatics in Medicine Unlocked 2020;20:100412.
[43] Sherstinsky A. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena 2020;404: 132306.
[44] Polsinelli M, Cinque L, Placidi G. A light CNN for detecting COVID-19 from CT scans of the chest. Pattern Recogn. Lett. 2020;140(0167-8655):95–100. https://doi.org/10.1016/j.patrec.2020.10.001.
[45] Ahsan MM, E Alam T, Trafalis T, Huebner P. Deep mlp-cnn model using mixed-data to distinguish between covid-19 and non-covid-19 patients. Symmetry 2020; 12(9):1526.
[46] Bahgat WM, Balaha HM, AbdulAzeem Y, Badawy MM. An optimized transfer learningbased approach for automatic diagnosis of covid-19 from chest x-ray images. PeerJ Comp Sci 2021;7:e555.
[47] Jain R, Gupta M, Taneja S, Hemanth DJ. Deep learning based detection and analysis of covid-19 on chest x-ray images. Appl Intell 2021;51(3):1690–700.
[48] Wang S, et al. A deep learning algorithm using ct images to screen for corona virus disease (covid-19). Eur Radiol 2021:1–9.
[49] Thelin EP, et al. Evaluation of novel computerized tomography scoring systems in human traumatic brain injury: an observational, multicenter study. PLoS Med 2017;14(8):e1002368.
[50] Mahasin M. Pneumonia (virus) vs COVID-19. https://www.kaggle.com/muhammadmasdar/pneumonia-virus-vs-covid19. [Accessed 1 January 2021].
[51] srikar. Covid-19 Xray images using CNN. https://www.kaggle.com/akkinasrikar/covid19-xray-images-using-cnn. [Accessed 1 January 2021].
[52] uddipta das. COVID-19 X-ray Images5. https://www.kaggle.com/uddiptadas/covid19-xray-images5. [Accessed 1 January 2021].
[53] Sajid N. COVID-19 patients lungs X ray images 10000. https://www.kaggle.com/nabeelsajid917/covid-19-x-ray-10000-images. [Accessed 1 January 2021].
[54] Sreeraman R. COVID-19 chest X rays. https://www.kaggle.com/rupeshs/covid19-chest-x-rays. [Accessed 1 January 2021].
[55] Riaz S. COVID-19 dataset. https://www.kaggle.com/syedrz/covid19-dataset. [Accessed 1 January 2021].
[56] Sait U. Curated chest X-ray image dataset for COVID-19. https://www.kaggle.com/unaissait/curated-chest-xray-image-dataset-for-covid19. [Accessed 1 January 2021].
[57] Patel P. Chest X-ray (Covid-19 & pneumonia). https://www.kaggle.com/prashant268/chest-xray-covid19-pneumonia. [Accessed 1 June 2021].
[58] Asraf A. COVID19 with pneumonia and normal chest Xray (PA) dataset. https://www.kaggle.com/amanullahasraf/covid19-pneumonia-normal-chest-xray-pa-dataset. [Accessed 1 June 2021].
[59] Khoong WH. COVID-19 Xray dataset (train & test sets). https://www.kaggle.com/khoongweihao/covid19-xray-dataset-train-test-sets. [Accessed 1 June 2021].
[60] Elmasry A. COVID19-xray. https://www.kaggle.com/anaselmasry/covid19xray. [Accessed 1 June 2021].
[61] Fenta F. Chest Xray for covid-19 detection. https://www.kaggle.com/fusicfenta/chest-xray-for-covid19-detection. [Accessed 1 June 2021].
[62] RoRonoA-TKO. Covid_19_2020. https://www.kaggle.com/tikoboss/covid-19-2020. [Accessed 1 June 2021].
[63] Deshpande D. COVID-19 detection X-ray dataset. https://www.kaggle.com/darshan1504/covid19-detection-xray-dataset. [Accessed 4 June 2021].
[64] Singh T. COVID-19 & normal posteroanterior(PA) X-rays. https://www.kaggle.com/tarandeep97/covid19-normal-posteroanteriorpa-xrays. [Accessed 4 June 2021].
[65] Viradiya P. COVID-19 radiography dataset. https://www.kaggle.com/preetviradiya/covid19-radiography-dataset. [Accessed 4 June 2021].
[66] Chaudhary Yash. Covid-GAN and Covid-Net mini chest X-ray. https://www.kaggle.com/yash612/covidnet-mini-and-gan-enerated-chest-xray. [Accessed 4 June 2021].
[67] Asraf A. COVID19_pneumonia_normal_chest_Xray_PA_dataset. https://www.kaggle.com/asraf047/covid19-pneumonia-normal-chest-xray-pa-dataset. [Accessed 4 June 2021].
[68] Dincer T. Chest X-ray images. https://www.kaggle.com/tolgadincer/labeled-chest-xray-images. [Accessed 4 June 2021].
[69] Refat CMM. Chest Xray images pneumonia and Covid-19. https://www.kaggle.com/masumrefat/chest-xray-images-pneumonia-and-covid19. [Accessed 4 June 2021].
[70] Saravanan C. Color image to grayscale image conversion in 2010 second international conference on computer engineering and applicationsVol. 2. IEEE; 2010. p. 196–9.
[71] Wan Y, Xie Q. A novel framework for optimal rgb to grayscale image conversion in 2016 8th international conference on intelligent human-machine systems and cybernetics (IHMSC)Vol. 2. IEEE; 2016. p. 345–8.
[72] Čadík M. Perceptual evaluation of color-to-grayscale image conversions in *computer graphics forum*Vol. 27. Wiley Online Library; 2008. p. 1745–54.

[73] Gedraite ES, Hadad M. Investigation on the effect of a gaussian blur in image filtering and segmentation. Proceedings ELMAR-2011 (IEEE) 2011:393–6.
[74] Tronarp F, Garcia-Fernandez AF, Särkkä S. Iterative filtering and smoothing in nonlinear and non-gaussian systems using conditional moments. IEEE Signal Processing Letters 2018;25(3):408–12.
[75] Chaki N, Shaikh SH, Saeed K. A comprehensive survey on image binarization techniques in *exploring image binarization techniques*. Springer; 2014. p. 5–15.
[76] Soille P. Morphological image analysis: Principles and applications. Springer Science & Business Media; 2013.
[77] Gil JY, Kimmel R. Efficient dilation, erosion, opening, and closing algorithms. IEEE Trans Pattern Anal Mach Intell 2002;24(12):1606–17.
[78] Khosravy M, Gupta N, Marina N, Sethi IK, Asharif MR. Morphological filters: an inspiration from natural geometrical erosion and dilation in *nature-inspired computing and optimization*. Springer; 2017. p. 349–79.
[79] Law G. Quantitative comparison of flood fill and modified flood fill algorithms. Int J Comp Theory Eng 2013;5(3):503–8.
[80] Nosal EM. Flood-fill algorithms used for passive acoustic detection and tracking in *2008 new trends for environmental monitoring using passive systems*. IEEE; 2008. p. 1–5.
[81] Jawas N, Suciati N. Image inpainting using erosion and dilation operation. Int J Adv Sci Technol 2013;51:127–34.
[82] Chan TF, Vese LA. Active contours without edges. IEEE Trans Image Process 2001;10(2):266–77.
[83] Wang L, et al. Active contours driven by edge entropy fitting energy for image segmentation. Signal Process 2018;149:27–35.
[84] Bieniek A, Moga A. An efficient watershed algorithm based on connected components. Pattern Recogn 2000;33(6):907–16.
[85] Gao H, Xue P, Lin W (2004) A new marker-based watershed algorithm in 2004 IEEE international symposium on circuits and systems (IEEE Cat No. 04CH37512). (IEEE), Vol. 2, pp. II–81.
[86] Li D, Zhang G, Wu Z, Yi L. An edge embedded marker-based watershed algorithm for high spatial resolution remote sensing image segmentation. IEEE Trans Image Process 2010;19(10):2781–7.
[87] Perez L, Wang J. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. CoRR 2017:1712.04621. arXiv, http://arxiv.org/abs/1712.04621.
[88] Wang Xiang, Wang Kai, Lian Shiguo. A survey on face data augmentation. CoRR 2019:1904.11685. arXiv, http://arxiv.org/abs/1904.11685.
[89] Antoniou A, Storkey A, Edwards H. Data augmentation generative adversarial networks. arXiv 2017:1711.04340.
[90] Odena A, Olah C, Shlens J. Conditional image synthesis with auxiliary classifier gans in *international conference on machine learning*. PMLR; 2017. p. 2642–51.
[91] Donahue J, Krähenbühl P, Darrell T (2016) Adversarial feature learning. arXiv preprint arXiv:160509782.
[92] Denton Emily L, Gross Sam, Fergus Rob. Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks. CoRR 2016:1611.06430. http://arxiv.org/abs/1611.06430.
[93] Torrey L, Shavlik J. Transfer learning in handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global; 2010. p. 242–64.
[94] Pan SJ, Yang Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 2009;22(10):1345–59.
[95] Qassim H, Verma A, Feinzimer D. Compressed residual-vgg16 cnn model for big data places image recognition in 2018 IEEE 8th annual computing and communication workshop and conference (CCWC). IEEE; 2018. p. 169–75.
[96] Carvalho T, De Rezende ER, Alves MT, Balieiro FK, Sovat RB. Exposing computer generated images by eye's region classification via transfer learning of vgg19 cnn in 2017 16th IEEE international conference on machine learning and applications (ICMLA). IEEE; 2017. p. 866–70.
[97] Akiba Takuya, Suzuki Shuji, Fukuda Keisuke. Extremely large minibatch SGD: training ResNet-50 on ImageNet in 15 minutes. CoRR 2017:1711.04325. http://arxiv.org/abs/1711.04325.
[98] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. Mobilenetv2: inverted residuals and linear bottlenecks in proceedings of the IEEE conference on computer vision and pattern recognition. 2018. p. 4510–20.
[99] Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition. Proc IEEE Conf Comput Vis Pattern Recognit 2018:8697–710.
[100] Iandola Forrest N, Moskewicz Matthew W, Karayev Sergey, Girshick Ross B, Darrell Trevor, Keutzer Kurt. DenseNet: implementing efficient ConvNet descriptor pyramids, CoRR, arXiv, 1404.1869. https://dblp.org/rec/journals/corr/IandolaMKGDK14.bib; 2014.
[101] Xia X, Xu C, Nan B (2017) Inception-v3 for flower classification in *2017 2nd international conference on image, vision and computing (ICIVC)* (IEEE), pp. 783–787.
[102] Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning in proceedings of the AAAI conference on artificial intelligenceVol. 31; 2017.
[103] Deng J, et al. Imagenet: a large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition (IEEE) 2009:248–55.
[104] Weiss K, Khoshgoftaar TM, Wang D. A survey of transfer learning. J Big Data 2016;3(1):9.
[105] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM 2017;60(6):84–90.
[106] Balaha HM, Ali HA, Saraya M, Badawy M. A new arabic handwritten character recognition deep learning system (ahcr-dls). Neural Comput Applic 2021;33(11):6325–67.
[107] Seliya N, Khoshgoftaar TM, Van Hulse J. A study on the relationships of classifier performance metrics in. In: 2009 21st IEEE International Conference on Tools with Artificial Intelligence; 2009. p. 59–66.
[108] Marler RT, Arora JS. The weighted sum method for multi-objective optimization: new insights. Struct Multidiscip Optim 2010;41(6):853–62.
[109] Bernardin K, Stiefelhagen R. Evaluating multiple object tracking performance: the clear mot metrics. EURASIP J Image Video Process 2008;2008:1–10.
[110] Hämäläinen W. Class np, np-complete, and np-hard problems. 2006.
[111] Chouhan SS, Kaul A, Singh UP. Soft computing approaches for image segmentation: a survey. Multimed Tools Appl 2018;77(21):28483–537.
[112] Wang SC. Genetic algorithm in *interdisciplinary computing in Java programming*. Springer; 2003. p. 101–16.
[113] Balaha HM, et al. Recognizing arabic handwritten characters using deep learning and genetic algorithms. Multimed Tools Appl 2021:1–37.
[114] 8023 I. COVID-19 radiography database. https://github.com/ieee8023/covid-chestxray-dataset. [Accessed 1 January 2021].
[115] Liu L, Shen C, van den Hengel A. The treasure beneath convolutional layers: crossconvolutional-layer pooling for image classification in proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 4749–57.
[116] Bkassiny M, Li Y, Jayaweera SK. A survey on machine-learning techniques in cognitive radios. IEEE Commun Surv Tutorials 2012;15(3):1136–59.
[117] Woźniak M, Siłka J, Wieczorek M. Deep neural network correlation learning mechanism for ct brain tumor detection. Neural Comput Applic 2021:1–16.
[118] O'Shea Keiron, Nash Ryan. An introduction to convolutional neural networks, CoRR, arXiv, 1511.08458. https://dblp.org/rec/journals/corr/OSheaN15.bib; 2015.
[119] Kalchbrenner Nal, Grefenstette Edward, Blunsom Phil. A convolutional neural network for modelling sentences. CoRR, arXiv, 1404.2188. https://dblp.org/rec/journals/corr/KalchbrennerGB14.bib; 2014.
[120] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks in *European conference on computer vision*. Springer; 2014. p. 818–33.
[121] Hidaka A, Kurita T. Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks. Proceedings of the ISCIE international symposium on stochastic systems theory and its applications (The ISCIE symposium on stochastic systems theory and its applications) 2017;Vol. 2017:160–7.
[122] Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. Insights Imaging 2018;9(4):611–29.
[123] Sharma S. Activation functions in neural networks. Towards Data Science 2017;6.
[124] Agarap Abien Fred. Deep learning using Rectified Linear Units (ReLU). CoRR, arXiv, 1803.08375. https://dblp.org/rec/journals/corr/abs-1803-08375.bib; 2018.
[125] Hu Yuhuang, Huber Adrian EG, Anumula Jithendar, Liu Shih-Chii. Overcoming the vanishing gradient problem in plain recurrent networks. CoRR, arXiv, 1801.06105. https://dblp.org/rec/journals/corr/abs-1801-06105.bib; 2018.
[126] Lu Lu. Dying ReLU and initialization: theory and numerical examples. Communications in Comput. Phys. 2020;28(5):1991–7120. https://doi.org/10.4208/cicp.OA-2020-0165.
[127] Dubey AK, Jain V. Comparative study of convolution neural network's relu and leakyrelu activation functions in *applications of computing, automation and wireless systems in electrical engineering*. Springer; 2019. p. 873–80.
[128] Peng H, Li J, Song Y, Liu Y. Incrementally learning the hierarchical softmax function for neural language models. Proceedings of the AAAI Conference on Artificial Intelligence 2017;Vol. 31.
[129] Duan K, Keerthi SS, Chu W, Shevade SK, Poo AN. Multi-category classification by soft-max combination of binary classifiers in *international workshop on multiple classifier systems*. Springer; 2003. p. 125–34.
[130] Young SR, Rose DC, Karnowski TP, Lim SH, Patton RM. Optimizing deep learning hyper-parameters through an evolutionary algorithm. Proceedings of the workshop on machine learning in high-performance computing environments 2015:1–5.
[131] Ruder Sebastian. An overview of gradient descent optimization algorithms. CoRR, arXiv, 1609.04747. https://dblp.org/rec/journals/corr/Ruder16.bib; 2016.
[132] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 2011;12(7).
[133] Zeiler Matthew D. ADADELTA: an adaptive learning rate method. CoRR, arXiv, 1212.5701. https://dblp.org/rec/journals/corr/abs-1212-5701.bib; 2012.
[134] Bengio Y, CA M (2015) Rmsprop and equilibrated adaptive learning rates for nonconvex optimization. *corr abs/1502.04390*.
[135] Xiang T, Wang J, Liao X. An improved particle swarm optimizer with momentum. In: *2007 IEEE congress on evolutionary computation*. (IEEE); 2007. p. 3341–5.
[136] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Binarized neural networks. Adv Neural Inf Proces Syst 2016;29:4107–15.
[137] Dozat T. Incorporating nesterov momentum into adam. 2016.
[138] McMahan HB, et al. Ad click prediction. a view from the trenches in Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining 2013:1222–30.
[139] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.
[140] Ioffe Sergey, Szegedy Christian. Batch normalization: accelerating deep network training by reducing internal covariate shift. CoRR, arXiv, 1502.03167. https://dblp.org/rec/journals/corr/IoffeS15.bib; 2015.

[141] Van Laarhoven T (2017) L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05350.
[142] Mirjalili S. Genetic algorithm in *evolutionary algorithms and neural networks*. Springer; 2019. p. 43–55.
[143] Woźniak M, Połap D. Bio-inspired methods modeled for respiratory disease detection from medical images. Swarm Evol Comput 2018;41:69–96.
[144] Ke Q, et al. A neuro-heuristic approach for recognition of lung diseases from x-ray images. Expert systems with applications 2019;126:218–32.
[145] Genlin J. Survey on genetic algorithm [j]. Computer Applications and Software 2004;2(1):69–73.
[146] Li T, Shao G, Zuo W, Huang S. Genetic algorithm for building optimization: state-of-the-art survey in proceedings of the 9th international conference on machine learning and computing. 2017. p. 205–10.