

Efficient Deep CNN Model for COVID-19 Classification

Walid El-Shafai^{1,2,*}, Amira A. Mahmoud¹, El-Sayed M. El-Rabaie¹, Taha E. Taha¹, Osama F. Zahran¹, Adel S. El-Fishawy¹, Mohammed Abd-Elnaby³ and Fathi E. Abd El-Samie^{1,4}

¹Department Electronics and Electrical Communications, Faculty of Electronic Engineering, Menoufia University, Menouf, 32952, Egypt

²Security Engineering Lab, Computer Science Department, Prince Sultan University, Riyadh, 11586, Saudi Arabia

³Department of Computer Engineering, College of Computers and Information Technology, Taif University, Taif, 21944, Saudi Arabia

⁴Department of Information Technology, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh, 84428, Saudi Arabia

*Corresponding Author: Walid El-Shafai. Email: eng.waled.elshafai@gmail.com

Received: 11 April 2021; Accepted: 18 June 2021

Abstract: Coronavirus (COVID-19) infection was initially acknowledged as a global pandemic in Wuhan in China. World Health Organization (WHO) stated that the COVID-19 is an epidemic that causes a 3.4% death rate. Chest X-Ray (CXR) and Computerized Tomography (CT) screening of infected persons are essential in diagnosis applications. There are numerous ways to identify positive COVID-19 cases. One of the fundamental ways is radiology imaging through CXR, or CT images. The comparison of CT and CXR scans revealed that CT scans are more effective in the diagnosis process due to their high quality. Hence, automated classification techniques are required to facilitate the diagnosis process. Deep Learning (DL) is an effective tool that can be utilized for detection and classification this type of medical images. The deep Convolutional Neural Networks (CNNs) can learn and extract essential features from different medical image datasets. In this paper, a CNN architecture for automated COVID-19 detection from CXR and CT images is offered. Three activation functions as well as three optimizers are tested and compared for this task. The proposed architecture is built from scratch and the COVID-19 image datasets are directly fed to train it. The performance is tested and investigated on the CT and CXR datasets. Three activation functions: Tanh, Sigmoid, and ReLU are compared using a constant learning rate and different batch sizes. Different optimizers are studied with different batch sizes and a constant learning rate. Finally, a comparison between different combinations of activation functions and optimizers is presented, and the optimal configuration is determined. Hence, the main objective is to improve the detection accuracy of COVID-19 from CXR and CT images using DL by employing CNNs to classify medical COVID-19 images in an early stage. The proposed model achieves a classification accuracy of 91.67% on CXR image dataset, and a classification accuracy of 100% on CT dataset with training times of 58 min and 46 min on CXR and CT datasets, respectively. The best



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

results are obtained using the ReLU activation function combined with the SGDM optimizer at a learning rate of 10^{-5} and a minibatch size of 16.

Keywords: COVID-19; image classification; CNN; DL; activation functions; optimizers

1 Introduction

The epidemic of COVID-19, which appeared in Wuhan city in China, results in pneumonia with fever and cough as the main indications of infection. A study performed on CT images to detect the disease infection proved that the detection rate from CT images is better than that from the RT-PCR. So, a chest CT scan was recommended [1–5].

Classification is an essential process in learning tasks, and it is a fundamental problem in the recognition area, which aims to classify medical images into several different categories. The classification of medical images includes two main steps. Firstly, the most helpful image features are extracted. Secondly, these features are used in building the models for dataset classification. Usually, specialists use their feature extraction experience to categorize medical images into different categories, making the classification sometimes tricky and time-wasting. Recently, DL has arisen due to its high quality and vast application domains in several research areas, especially for classifying medical images since pre-processing or feature extraction is not required before training the model. A CNN is one of the latest progressions in machine learning (ML) area. It can be used for the analysis of medical images.

With the massive growth of neural networks and DL, finding an optimum model architecture for each application is necessary. Much work has been carried out to achieve the desired performance level and to obtain the best accuracy in any classification task. Activation layers such as Sigmoid, Tanh, and ReLU define the non-linearity of the neuron output [6,7]. A CNN comprises several layers ordered as the input layer, convolution layer, activation layer, fully-connected layer, classification layer, and output layer. Moreover, as machine learning algorithms are optimized, a significant improvement in their performance can be achieved. Therefore, finding a suitable activation function and optimizer are basic tasks [6,7].

The objective of this work is to carry out comparisons between different activation functions and different optimizers for the classification of CXR and CT image datasets for COVID-19 detection. The CNNs have proved efficient performance in the classification of medical images. Therefore, this paper presents a CNN model for COVID-19 detection from CXR and CT images with a new training strategy. This strategy depends on the proper selection of the optimizer and the activation function. The rest of this paper is structured as follows. Section 2 summarizes the related work in this field. Section 3 gives short notes about the CNN. Section 4 describes the materials and methods used in the paper. Section 5 illustrates the proposed model architecture. Section 6 shows the experimental results and discussions. Section 7 provides the conclusions.

2 Related Work

The World Health Organization (WHO) has stated that COVID-19 rapidly spread in several countries worldwide. Early detection of COVID-19 cases can significantly control the spread of this virus. Much work has been performed on this topic due to its importance. This paper depends on DL to automatically detect COVID-19 from CXR and CT images. The performance of different classifiers is investigated to determine the optimum one [1–3]. The CXR and CT images can be used to detect COVID-19 cases. The CNN is one of the most popular and effective

tools that identify COVID-19 from medical images [1–3]. Several review studies have been presented to highlight recent contributions to COVID-19 detection [8–16]. Several works used radiology images to identify and classify COVID-19 cases. Zheng et al. [13] proposed a DL model to classify pneumonia. Xu et al. [14] presented a model to classify pneumonia from CXR images based on compressed sensing (CS) with a deep transfer learning model. Sethy et al. [17] used the SVM classifier to classify the features acquired from several CNN models applied on CXR images. They achieved the best performance using the ResNet50 model with SVM.

Wang et al. [18] suggested transfer learning model called COVID-Net to detect COVID-19 from CXR images. Their model achieved 92.4% accuracy for three classes: Normal, Non-COVID pneumonia, and COVID-19. Hemdan et al. [19] applied DL models to detect COVID-19 from CXR images and suggested a model called COVIDXNet. Their model achieved a 0.95 AUC value and a 0.96 sensitivity. Additionally, there is an online service to diagnose COVID-19 from CT images [20]. Wang et al. [21] used a CNN based on the Inception network model to identify COVID-19 cases from CT images. Ioannis et al. [22] proposed a DL model using 224 confirmed COVID-19 images. The authors of [23] proposed a model to classify COVID-19, influenza, and healthy CT image cases. Their model achieved an accuracy of 86.7%. In [24], the authors proposed a learning model to separate the main features in CT images in a pre-processing stage. Their model achieved accuracies of 89.5% and 79.3% with and without the pre-processing stage, respectively. Ozturk et al. [25] suggested a model that classifies CXR COVID-19 images. Their model has been applied to classify three main classes: COVID, No-COVID, and pneumonia, and achieved a classification accuracy of 87.02%. Alsharman et al. [26] used CNNs to classify CT COVID-19 images. They used a pretrained Google-Net CNN architecture and achieved an accuracy of 82.14%.

Table 1: Overview of the recent work using deep learning techniques for medical image classification

Study	Image modality	No. of images	Classifier	Accuracy
Ref. [22]	CXR	224 COVID-19 (+)/700 Pneumonia/ 504 Healthy	VGG-19	93.48
Ref. [18]	CXR	53 COVID-19 (+)/5526 COVID-19 (-)/ 8066 Healthy	COVID-Net	92.4
Ref. [17]	CXR	25 COVID-19 (+)/ 25 COVID-19 (-)	ResNet50 + SVM	95.38
Ref. [19]	CXR	25 COVID-19 (+)/25 Normal	COVIDX-Net	90.0
Ref. [21]	CT	195 COVID-19 (+)/258 COVID-19 (-)	M-Inception	82.9
Ref. [25]	CXR	125 COVID-19 (+)/500 Pneumonia/500 No-Findings	DarkCovidNet	87.02
Ref. [13]	CT	313 COVID-19 (+)/229 COVID-19 (-)	UNet + 3D Deep Network	90.8
Ref. [14]	CT	219 COVID-19 (+)/224 Viral pneumonia/ 175 Healthy	ResNet + Location Attention	86.7
Ref. [15]	CT	107 COVID-19 (+)/74 COVID-19 (-)	SVM	99.68%
Ref. [26]	CT	349 COVID-19 (+)/No-finding	GoogleNet	82.14
Ref. [20]	CT	88 Covid-19 (+)/101 bacterial pneumonia/ 86 normal	Deep pneumonia system	94%

The DL growth has a significant effect on the medical field due to the better ability to classify medical images. Several image classification techniques can give radiologists another opinion. The recent research works on medical image classification are summarized in [Tab. 1](#).

In this paper, a DL model is presented to classify COVID-19 CXR and CT images. The proposed model has been trained from scratch without using any feature extraction approaches. It has been trained with 1000 CXR and 1000 CT medical images. One of the essential advantages of well-trained DL models is that they can extract features that are not apparent to the human eye. Hence, accurate classification can be performed.

3 Convolutional Neural Networks (CNNs)

Recently, DL has arisen due to its efficiency in a variety of application domains in several research areas, especially for classifying medical images since pre-processing or feature extraction is not required before the training process. The CNN has gained a significant importance, and it was utilized in most of the state-of-the-art applications. They were extensively used to detect and identify diseases in different medical images. The main difference between a CNN and an ANN is that the CNN has a large number of hidden layers. So, the CNN constitutes a deep architecture. It consists of several stacked layers ordered as input layer, convolution layer, pooling layers, activation layer, fully-connected layer, classification layer, and output layer.

The input layer enhances the image using pre-processing such as normalization and scaling. The convolution layer convolves the image with several suitably adjusted filters. This convolution results in feature maps. Then, the pooling layers are used to minimize the dimensions of the generated feature maps. Pooling is carried out using a window with a proper stride. Either max-pooling or average pooling is used. In max pooling, the maximum value is chosen.

On the other hand, in average pooling, the average value is estimated and used. The activation functions define the non-linearity of the model. Finally, the fully-connected layer is the output layer that clarifies the classification result using the SoftMax classifier to determine the image class.

3.1 Activation Functions

The appropriate activation functions must be carefully chosen, because they significantly affect the neural network performance. The main target of activation functions is that they provide non-linearity to their input. There are three famous activation functions, namely, Sigmoid, Tanh, and ReLU. These functions are used and studied in this work. They are summarized as follows:

- **Logistic Curve (Sigmoid)**

The Sigmoid function is defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

The sigmoid activation function converts its input range from $[-\infty; +\infty]$ to $[0; 1]$. The main disadvantage of the sigmoid is that it is computationally expensive, and it cannot solve the problem of vanishing gradients.

- **Hyperbolic Tangent (Tanh)**

The Tanh is a non-linear function. It converts the range of the input to $[-1, 1]$. It can be defined as:

$$\text{Tanh}(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2)$$

An advantage is that Tanh has steeper derivatives than the sigmoid function. On the other hand, it cannot solve the vanishing gradient problem.


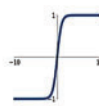

- **Rectified Linear Units (ReLU)**

The ReLU is the most common activation function, and is the mostly-used one. Using the ReLU function in a model makes it easier to train and often achieve better performance. The ReLU function is defined as follows:

$$\text{ReLU} = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \quad (3)$$

The main advantage of the ReLU function is that it contains no exponential terms or divisions, which results in increased the computation speed. However, it easily overfits. The benefits and limitations of different employed activation functions examined through the simulation tests are summarized in [Tab. 2](#).

Table 2: Summary of advantages and disadvantages of the examined activation functions

Activation Function	Advantages	Disadvantages
Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ 	<ul style="list-style-type: none"> • Easy to understand. • Used primarily on shallow networks. • Its output value is in the interval of $[0,1]$. • The activation value does not vanish. 	<ul style="list-style-type: none"> • Avoided when initializing a network with small random weights. • It suffers sharp damp gradients, slow convergence, and non-zero centered output. • Gradient updates not in the same direction.
tanh $\tanh(x)$ 	<ul style="list-style-type: none"> • Zero-centered output. • A gradient of 1 is obtained with 0 input. • Its derivative is steeper. • More efficient due to the wide range. 	<ul style="list-style-type: none"> • Like sigmoid, it suffers from vanishing gradient problem.
ReLU $\max(0, x)$ 	<ul style="list-style-type: none"> • No exponentials or divisions result in increased computation speed. • Sparsity in the hidden units and output values between zero and maximum. 	<ul style="list-style-type: none"> • It easily overfits.

3.2 Optimizers

- **Stochastic Gradient Descent with Momentum (SGDM)**

Optimization of the model greatly contributes to minimizing the loss function. The SGDM is one of the powerful and most-commonly used optimizers. It is an improvement of the SGD optimizer. It depends on the current gradient and the past momentum to estimate the momentum in each dimension. It also accumulates the gradient of the past steps to determine the direction

to go. The SGDM optimizer saves the update at each iteration and decides the following update as a function of the current gradient and the previous momentum update.

$$\Delta w := \alpha \Delta w - \eta \nabla Q_i(w) \quad (4)$$

$$w := w + \Delta w \quad (5)$$

This leads to:

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w \quad (6)$$

where w is the parameter, which decreases $Q(w)$, η is the learning rate, and α is an exponential decay factor between 0 and 1 that controls the relative contribution of the current gradient and the previous one to update the current momentum. Unlike SGD optimizer, the SGDM optimizer tends to keep moving in the same direction to avoid oscillations.

- **Root Mean Square Propagation (RMSprop)**

Another optimizer is the RMSprop, which also breaks the learning rate using the average exponential decay of squared gradients. It depends on the momentum to minimize the loss function relatively faster. Like momentum, the RMSprop also tries to decrease the oscillations using another method. It automatically adjusts the learning rate by choosing a different one for each parameter. It calculates the running average using the mean square error. It also depends on the past gradient to estimate the learning rate.

$$v(w, t) := \gamma v(w, t - 1) + (1 - \gamma)(\nabla Q_i(w))^2 \quad (7)$$

where γ is the forgetting factor, and the updated parameters are given as:

$$w := w - \frac{\eta}{\sqrt{v(w, t)}} \nabla Q_i(w) \quad (8)$$

- **Adaptive Moment (Adam)**

Adam algorithm merges the properties of momentum and some of the benefits of the RMSprop. Adam optimizer determines the adaptive learning rates for each parameter. Like momentum, Adam optimizer retains an exponential decay average of the past gradient descent v_t to reach a minimum faster, and stores an exponentially decaying average of previously squared gradients m_t like RMSprop [27,28]. The decaying averages of past and past squared gradients m_t and v_t are computed as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (9)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (10)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (11)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (12)$$

The Adam optimizer update rule is given by:

$$\theta_{t+1} \leftarrow \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}} \quad (13)$$

where ε is a small quantity (e.g., 10^{-8}) utilized to avoid division by 0, β_1 (e.g., 0.9) and β_2 (e.g., 0.999) are the forgetting factors and second moments for the gradients, respectively. The benefits and limitations of different optimizers are summarized in [Tab. 3](#).

Table 3: Summary of the advantages and disadvantages of the examined optimizers

Optimizer	Advantages	Disadvantages
SGDM	<ul style="list-style-type: none"> • Very simple to implement. • Fast, robust, and flexible. • Faster convergence and reduced oscillations. • Little required memory specifications. • High speed of convergence. 	<ul style="list-style-type: none"> • One more variable is calculated for every update.
RMSprop	<ul style="list-style-type: none"> • An average of the squared gradient determines the diminishing learning rates. • The magnitude of the previous gradient is employed to normalize the current gradient. • Learning rate is updated automatically. 	<ul style="list-style-type: none"> • The gradients square positive accumulation may reduce the learning rate, significantly.
Adam	<ul style="list-style-type: none"> • Easy to realize. • Computationally inexpensive. • Small memory requirements. • Appropriate for parameter problems and massive data. • Hyper-parameters require little tuning. • As in RMSprop, instead of learning rates established on the first average moment (the mean), the average gradient of the second moment is used to adapt the parameters. 	<ul style="list-style-type: none"> • The conflict in the optimization landscape is reduced. • Low values of the second moment. • The update procedure exceeds an ideal result as a result of a high divergence and learning rate.

4 Material and Methods

This motivation of this work is to offer a proposed simple deep CNN structural design for categorizing and classifying COVID-19 and Non-COVID-19 cases. This section describes all datasets used in this paper. In this study, simulation experiments are conducted on 1000 chest CXR and 1000 CT images of COVID-19 and Non-COVID-19 obtained from the open-source Mendeley datasets [29].

The dataset is divided into a 70% training set and a 30% validation set. The partitioned datasets of the training and testing help in data cross-validation. The cross-validation checks whether the suggested classifier precisely classifies the normal vs. COVID-19 images or not. A sample of employed datasets is shown in [Fig. 1](#).

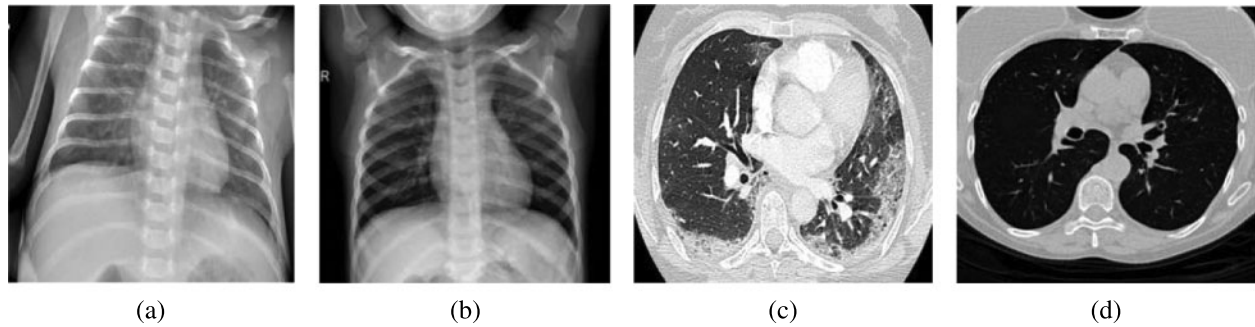


Figure 1: Samples of CXR and CT COVID-19 and non-COVID-19 images (a) CXR COVID-19 images (b) CXR non-COVID images (c) CT COVID-19 images (d) CT non-COVID images

5 Proposed Deep CNN Model

If we examine the performance of a CNN, it is evident that the network performance is enhanced with the increase in network depth. This comes at the cost of large memory requirements. We try in the proposed deep learning model to make a trade-off between network size and network performance. The proposed CNN model is made up of 14 layers, as illustrated in Fig. 2. The input image size is 227×227 pixels, and it is fed into the first convolution layer that has eight filters with size 3×3 and stride 1. The input image is zero-padded to get the output image size the same as that of the input image size. The output is fed into the ReLU function, and finally, it is max-pooled with a window size 2×2 and stride of 2 to down-sample the image. These layers are followed by two similar structures. The first one depends on 16 filters of size 3×3 and stride one, and the second depends on 32 filters of size 3×3 and stride 1 also. The last max-pooling layer is eliminated. We use a SoftMax classifier to convert each class score into a probability distribution, and then use the cross-entropy as the loss function.

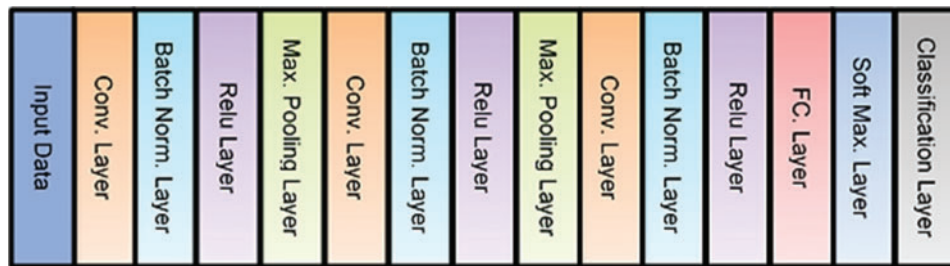


Figure 2: Proposed deep CNN model

6 Experimental Results and Discussions

To validate the suggested CNN model performance, the training procedure is repeated several times with different values of hyperparameters. Different activation functions and optimizers are tested to validate the performance of the proposed model on different CXR and CT images. In this section, the proposed CNN model is firstly trained to categorize and classify the CXR and CT medical images into two categories: COVID-19 and Non-COVID-19.

The first performed experiment is for performance comparison of different activation functions on the two used datasets. The tested neural networks are carried out for six epochs with batch sizes of 8, 16, and 32. In the first tested scenario, the analyzed neural networks are equipped with the SGDM, RMSprop, and Adam optimization techniques. The utilized learning rate is 10^{-5} for the three optimization algorithms. The learning rate is kept constant in the simulation tests, but the network structure and the activation functions are variable. The obtained results using the sigmoid function on the CXR dataset are shown in [Tabs. 4–6](#), and [Tabs. 13–15](#) for the CT database. In addition, the results of the Tanh function applied on the same dataset are shown in [Tabs. 7–9](#) and [Tabs. 16–18](#) for the CT database. Finally, the ReLU function results are shown in [Tabs. 10–12](#) and [Tabs. 19–21](#) for the CT database.

The second performed experiment is for performance comparison of the Adaptive (Adam), Root Mean Square propagation (RMSprop), and Stochastic Gradient Descent with Momentum (SGDM) optimizers at a fixed learning rate of 10^{-5} . All neural networks run on the two datasets, using the previously mentioned activation functions. Additionally, 1, 3, and 6 epochs allow assessment by averting duplicate accuracy values and avoiding overfitting cases.

• Performance of the Proposed Model on CXR Database

Table 4: Model performance using sigmoid activation function with a 8 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	87	00:22:09	75.00	81.67	0.3450	0.5716
	3	261	01:02:56	100.00	80.00	0.0482	0.7276
	6	522	02:04:11	100.00	86.33	0.0702	0.5228
RMSprop	1	87	00:23:09	87.50	76.00	0.2119	0.9053
	3	261	01:02:56	100.00	80.00	0.0482	0.7276
	6	522	02:04:11	100.00	86.33	0.0702	0.5228
Adam	1	87	00:23:36	87.50	64.67	0.2363	0.8838
	3	261	01:10:01	75.00	84.67	0.7864	0.3349
	6	522	02:14:00	87.50	87.33	0.1967	0.3162

Table 5: Model performance using sigmoid activation function with a 16 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy %	Validation accuracy %	Mini-batch loss	Validation loss
SGDM	1	42	00:12:08	56.25	56.00	2.3791	1.8141
	3	129	00:34:09	68.75	89.00	1.0294	0.4070
	6	258	01:05:52	75.00	84.67	0.3124	0.4780
RMSprop	1	42	00:12:47	37.50	50.00	6.4384	3.4840
	3	129	00:33:40	75.00	82.00	1.0728	0.6813
	6	258	01:08:42	100.00	85.67	0.0665	0.7768
Adam	1	42	00:14:37	56.25	63.33	0.6961	0.7373
	3	129	00:38:50	100.00	86.00	0.1160	0.3401
	6	258	01:17:58	93.75	88.33	0.1487	0.3084

Table 6: Model performance using sigmoid activation function with a 32 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	21	00:07:04	78.13	77.00	0.4279	0.4671
	3	63	00:21:13	84.38	83.00	0.2949	0.3978
	6	126	00:44:42	96.88	76.33	0.2024	0.4907
RMSprop	1	21	00:07:47	71.88	55.00	3.7996	1.4522
	3	63	00:22:19	46.88	51.67	3.7887	2.9496
	6	126	00:44:41	75.00	79.33	1.3629	0.6694
Adam	1	21	00:08:23	59.38	80.33	0.9747	0.4499
	3	63	00:23:27	81.25	83.00	0.3955	0.4033
	6	126	00:42:35	78.13	85.67	0.5658	0.3600

Table 7: Model performance using Tanh activation function with a 8 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	87	00:23:49	87.50	88.33	0.9795	0.3907
	3	261	01:13:02	100.00	87.67	0.0149	0.4945
	6	522	02:21:00	100.00	86.67	0.0046	0.4817
RMSprop	1	87	00:22:45	87.50	86.00	0.1619	0.7019
	3	261	01:06:52	75.00	81.33	0.8338	0.9389
	6	522	02:14:30	100.00	84.33	0.0006	0.9540
Adam	1	87	00:28:41	87.50	87.00	1.3976	0.5139
	3	261	01:15:07	100.00	87.00	0.0109	0.6783
	6	522	02:23:42	100.00	87.33	0.0012	0.7052

Table 8: Model performance using Tanh activation function with a 16 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	42	00:12:49	87.50	83.67	0.7059	0.5363
	3	129	00:37:57	93.75	87.33	0.2122	0.4329
	6	258	01:15:12	100.00	87.67	0.0087	0.4564
RMSprop	1	42	00:12:49	81.25	82.33	1.0822	0.5674
	3	129	00:37:50	87.50	84.67	0.3276	0.5866
	6	258	01:14:37	100.00	88.00	0.0100	0.5889
Adam	1	42	00:12:38	81.25	85.33	1.2719	0.4390
	3	129	00:37:24	100.00	86.67	0.0516	0.4524
	6	258	01:14:24	100.00	88.00	0.0007	0.3883

Table 9: Model performance using Tanh activation function with a 32 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	21	00:09:07	87.50	86.00	0.1512	0.4753
	3	63	00:24:56	90.63	88.33	0.1536	0.3545
	6	126	00:50:04	100.00	88.33	0.0454	0.3416
RMSprop	1	21	00:09:50	81.25	79.00	0.7253	0.6730
	3	63	00:30:24	100.00	85.67	0.0231	0.6018
	6	126	00:57:48	96.88	89.67	0.0424	0.7339
Adam	1	21	00:09:12	96.88	80.67	0.2548	0.6865
	3	63	00:25:48	93.75	86.00	0.1817	0.4764
	6	126	00:50:03	100.00	86.33	0.0230	0.4802

Table 10: Model performance using ReLU activation function with a 8 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	87	00:15:26	87.50	87.67	0.7032	0.4226
	3	261	00:31:45	84.74	89.65	0.0522	0.6226
	6	522	01:22:03	100.00	90.00	0.0029	0.5387
RMSprop	1	87	00:14:09	87.50	71.33	0.9210	0.9480
	3	261	00:30:74	97.30	88.32	0.0110	0.7480
	6	522	01:20:25	100.00	90.33	0.0057	0.6414
Adam	1	87	00:14:24	87.50	86.67	0.1563	0.3667
	3	261	00:30:54	97.20	88.10	0.1203	0.2154
	6	522	01:21:09	100.00	90.00	0.0100	0.4292

Table 11: Model performance using ReLU activation function with a 16 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	42	00:10:30	100.00	82.33	0.1424	0.5366
	3	129	00:29:53	100.00	89.67	0.0122	0.2681
	6	258	00:58:29	100.00	91.67	0.0256	0.2378
RMSprop	1	42	00:10:06	75.00	78.00	1.1117	0.9517
	3	129	00:27:39	100.00	88.33	0.0226	0.6092
	6	258	00:55:29	100.00	86.33	0.0015	0.7413
Adam	1	42	00:10:13	75.00	88.33	1.4540	0.4697
	3	129	00:28:18	100.00	88.00	0.0793	0.3937
	6	258	00:52:51	100.00	89.00	0.0044	0.3820

Table 12: Model performance using ReLU activation function with a 32 mini-batch size and a 10^{-5} learning rate on the CXR database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	21	00:06:02	90.63	88.33	0.1381	0.3018
	3	63	00:17:26	90.63	90.33	0.2324	0.2672
	6	126	00:32:18	100.00	90.33	0.0423	0.2445
RMSprop	1	21	00:07:04	87.50	85.67	0.3897	0.4325
	3	63	00:19:06	53.13	55.00	1.8954	3.3760
	6	126	00:39:08	100.00	90.67	0.0141	0.3842
Adam	1	21	00:06:50	78.13	91.33	0.3447	0.3520
	3	63	00:19:43	100.00	90.33	0.0345	0.3131
	6	126	00:37:54	100.00	91.00	0.0153	0.3376

• **Performance of The Proposed Model on CT Database**

Table 13: Model performance using sigmoid activation function with a 8 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	87	00:22:22	87.50	96.67	0.2241	0.0983
	3	261	01:05:59	62.50	98.00	0.9690	0.0500
	6	522	02:40:34	100.00	97.00	5.3644e-07	0.1171
RMSprop	1	87	00:26:32	75.00	95.33	0.5611	0.1605
	3	261	01:17:40	100.00	94.33	0.0004	0.3350
	6	522	02:16:44	100.00	97.67	1.4901e-08	0.1633
Adam	1	87	00:26:32	75.00	95.33	0.5611	0.1605
	3	261	01:07:01	87.50	95.33	0.2723	0.2727
	6	522	02:08:41	100.00	96.00	0.0007	0.3056

Table 14: Model performance using sigmoid activation function with a 16 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	42	00:15:16	100.00	96.00	0.0112	0.1047
	3	129	00:36:47	100.00	95.33	0.0746	0.1110
	6	258	01:10:18	100.00	98.00	0.0006	0.0595
RMSprop	1	42	00:16:24	100.00	97.33	0.0115	0.0802
	3	129	00:46:10	100.00	98.33	3.0548e-06	0.0493
	6	258	01:25:20	100.00	99.00	4.3513e-06	0.0573
Adam	1	42	00:24:57	93.75	96.67	0.4868	0.1575
	3	129	01:20:05	100.00	96.33	0.0491	0.1556
	6	258	02:34:27	100.00	98.33	0.0175	0.0341

Table 15: Model performance using sigmoid activation function with a 32 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	21	00:07:32	96.88	97.33	0.2239	0.0815
	3	63	00:28:37	96.88	96.67	0.0366	0.0791
	6	126	00:50:48	90.63	96.67	0.4635	0.1192
RMSprop	1	21	00:07:32	96.88	97.33	0.2239	0.0815
	3	63	00:21:03	96.88	98.00	0.0805	0.0852
	6	126	00:44:11	90.63	94.33	0.3367	0.4265
Adam	1	21	00:08:32	87.50	83.67	0.3059	0.4871
	3	63	00:23:55	100.00	96.67	0.0071	0.0802
	6	126	00:44:07	96.88	98.00	0.0622	0.0577

Table 16: Model performance using Tanh activation function with a 8 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	87	00:24:06	100.00	97.00	$1.7092e-05$	0.0715
	3	261	01:10:07	100.00	99.67	0.0009	0.0133
	6	522	02:18:40	100.00	99.00	0.0003	0.0538
RMSprop	1	87	00:23:38	100.00	99.00	0.0250	0.0623
	3	261	01:09:04	100.00	99.00	$4.9174e-07$	0.0642
	6	522	02:16:58	100.00	98.67	$1.5374e-05$	0.0827
Adam	1	87	00:23:17	100.00	94.33	0.0003	0.4791
	3	261	01:08:27	100.00	98.33	$2.6803e-05$	0.1680
	6	522	02:18:12	100.00	98.33	$1.4901e-08$	0.1293

Table 17: Model performance using Tanh activation function with a 16 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	42	00:13:22	100.00	96.00	0.0002	0.1288
	3	129	00:38:33	100.00	97.33	0.0010	0.0788
	6	258	01:15:44	100.00	97.33	0.0005	0.0779
RMSprop	1	42	00:12:52	62.50	98.67	1.9588	0.0356
	3	129	00:37:29	100.00	99.33	0.0001	0.0333
	6	258	01:14:00	100.00	98.67	0.0081	0.0820
Adam	1	42	00:12:27	93.75	97.67	0.0818	0.1737
	3	129	00:36:59	100.00	98.33	0.0001	0.1090
	6	258	01:13:26	100.00	98.33	0.0006	0.1501

Table 18: Model performance using Tanh activation function with a 32 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	21	00:07:59	93.75	96.33	0.3697	0.1680
	3	63	00:22:40	100.00	98.00	0.0003	0.0599
	6	126	00:44:49	100.00	98.67	0.0016	0.0506
RMSprop	1	21	00:07:59	100.00	97.33	0.0093	0.0945
	3	63	00:22:42	100.00	98.67	0.0002	0.0335
	6	126	00:44:46	100.00	98.33	5.9324e-05	0.0761
Adam	1	21	00:07:58	96.88	98.67	0.4982	0.1688
	3	63	00:22:43	100.00	98.33	0.0015	0.0742
	6	126	00:44:42	100.00	99.33	5.9233e-07	0.0248

Table 19: Model performance using ReLU activation function with a 8 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	87	00:13:11	97.25	96.74	1.3251e-04	0.0562
	3	261	00:40:52	100.00	98.67	1.4111e-06	0.0927
	6	522	01:21:40	100.00	99.33	1.1921e-07	0.0746
RMSprop	1	87	00:15:46	93.54	88.74	1.1451e-04	0.2162
	3	261	00:43:18	100.00	91.37	1.5311e-06	0.0307
	6	522	01:27:25	100.00	99.67	1.6391e-07	0.0727
Adam	1	87	00:13:56	94.37	94.62	1.2451e-04	0.1062
	3	261	00:42:17	99.18	98.88	1.3611e-06	0.0007
	6	522	01:22:40	100.00	99.67	1.6391e-07	0.0053

Table 20: Model performance using ReLU activation function with a 16 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	42	00:08:18	100.00	99.67	0.0009	0.0160
	3	129	00:23:52	100.00	97.00	4.3246e-05	0.0888
	6	258	00:46:59	100.00	100.00	4.5806e-05	0.0059
RMSprop	1	42	00:07:51	100.00	99.33	0.0003	0.0145
	3	129	00:23:25	100.00	99.33	0.0002	0.0024
	6	258	00:47:31	100.00	99.33	4.2246e-06	0.0130
Adam	1	42	00:08:52	93.75	96.00	0.3215	0.3321
	3	129	00:28:58	100.00	95.67	0.0451	0.4021
	6	258	00:51:59	100.00	98.67	2.0862e-07	0.1061

Table 21: Model performance using ReLU activation function with a 32 mini-batch size and 10^{-5} learning rate on the CT database for 1, 3, and 6 epochs

Optimizer	Epoch No.	Iteration No.	Time elapsed (hh:mm:ss)	Mini-batch accuracy (%)	Validation accuracy (%)	Mini-batch loss	Validation loss
SGDM	1	21	00:05:03	93.75	96.67	0.1431	0.1082
	3	63	00:14:34	100.00	97.33	0.0039	0.0556
	6	126	00:29:19	100.00	99.67	0.0011	0.0455
RMSprop	1	21	00:05:54	100.00	99.33	0.0016	0.0134
	3	63	00:15:48	100.00	99.33	1.3560e-06	0.0011
	6	126	00:30:38	100.00	99.33	6.9663e-07	0.0012
Adam	1	21	00:05:13	90.63	96.33	0.4924	0.1225
	3	63	00:15:01	100.00	99.00	5.6997e-07	0.0457
	6	126	00:31:13	100.00	98.33	0.0006	0.0782

• Finding the Optimal Configuration

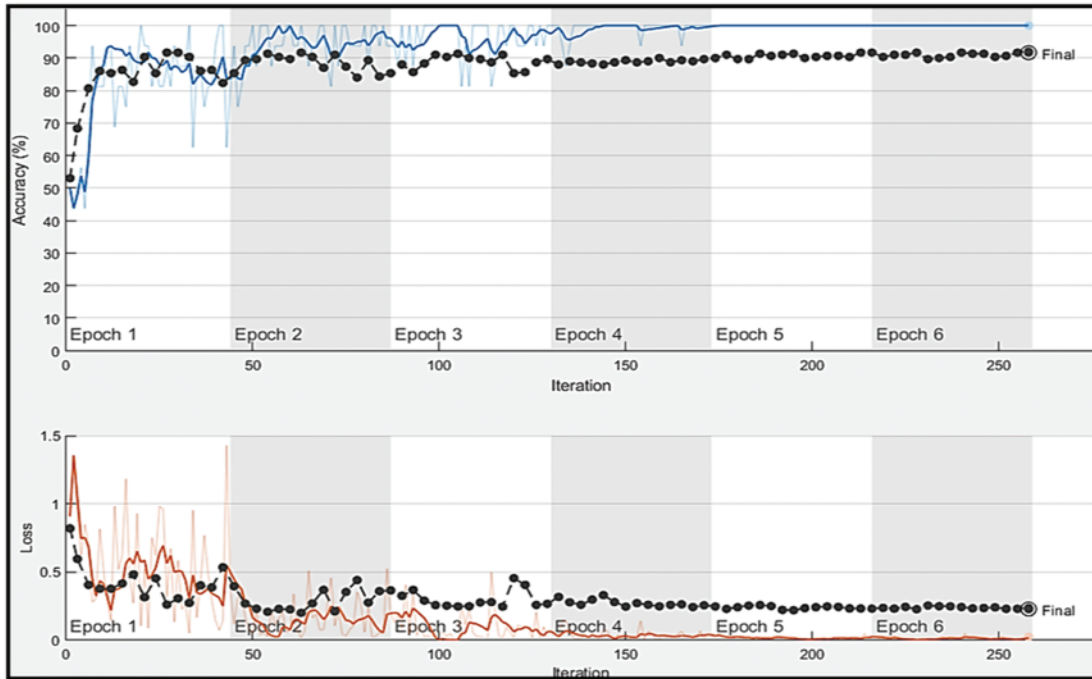
In this section, the effect of combining different optimizers with activation functions is studied and analyzed for improved COVID-19 detection. So, the third experiment scenario is the performance comparison of combining different optimizers and activation functions. The CXR dataset shows that the combination of SGDM with the ReLU activation function gives the best accuracy for 16 mini-batch sizes and a learning rate of 10^{-5} . The training process and the confusion matrix are shown in Fig. 3. Therefore, the employed neural network with a combination of the SGDM optimizer and the ReLU function work better than other combination scenarios. Thus, the SGDM/ReLU configuration can find a smaller local minimum with few epochs. Performing the same test on the CT database, it is also proved that combining the SGDM optimizer with the ReLU activation function gives the best accuracy for 16 mini-batch sizes and a 10^{-5} learning rate. An accuracy of 91.67% is achieved on the CXR dataset (with 93.3% Precision, 93.1% Sensitivity, and 90.3% Specificity). It is increased to 100% on the CT dataset (with 100% Precision, 100% Sensitivity, and 100% Specificity).

• Result Discussion

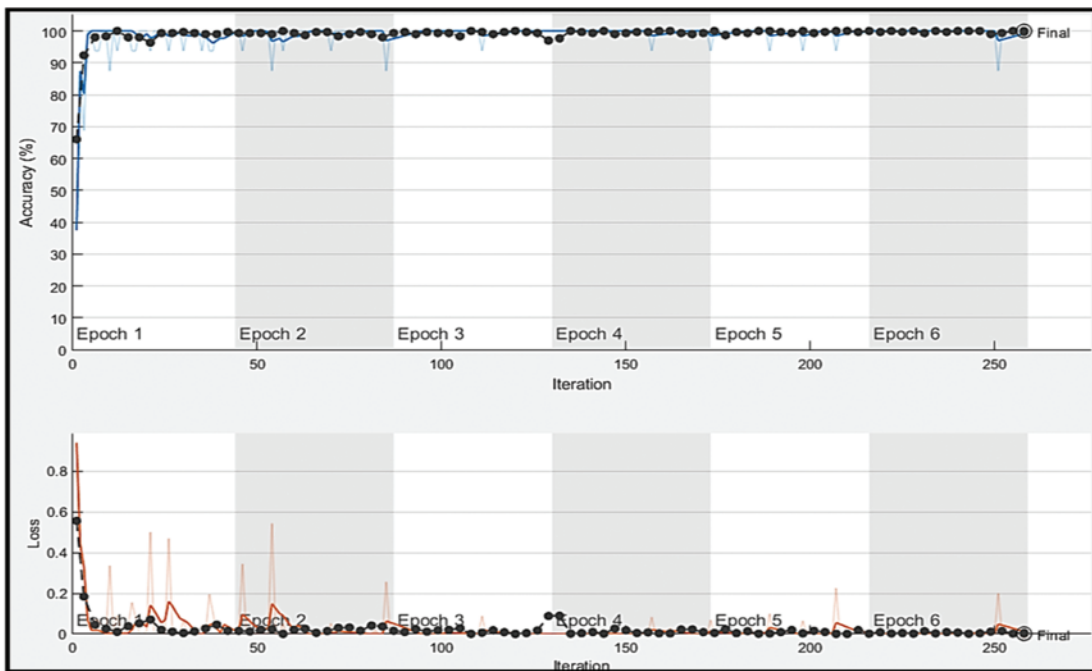
This paper concentrates on the benefits of using different activation functions and optimizers to build a model that can classify the COVID-19 from CXR and CT medical images. The test findings reveal that the suggested deep CNN model is very effective and helpful in discovering and classifying COVID-19 cases. It is recommended to use a CT scan, because the best classification results can be obtained on CT images. The CXR dataset can be increased in size for more improved classification accuracy. It is shown that the main advantage of the sigmoid function is that it is easy to implement on shallow networks. Its output value is in the range of 0 to 1, when the input is in the range of $-\infty$ to $+\infty$. Hence, the activation value does not vanish.

Conversely, the sigmoid function is not suitable, when the neural network is initialized for small weights. The Tanh function outperforms the sigmoid function as it gives a superior performance. It has a steeper derivative leading to fast learning. Similar to the sigmoid function, the Tanh function suffers from the vanishing gradient problem. Sigmoid and Tanh functions activate the majority of the neurons in the same way.

The ReLU function is preferred over the sigmoid function or Tanh function with generalized increased computation speed, since it does not depend on exponentials or divisions. However, the ReLU function has a restriction that it overfits compared to the sigmoid function.



(a)



(b)

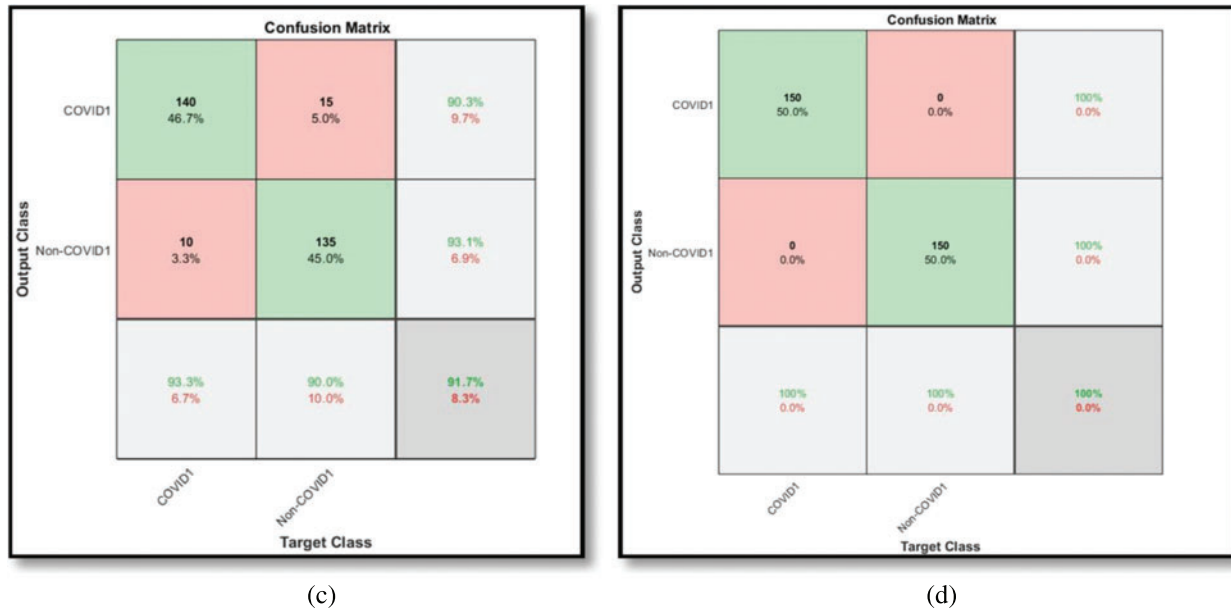


Figure 3: Performance of the proposed SGDM/ReLU model with 16 minibatch sizes at a learning rate of $1e-5$. (a) The training and validation processes of the proposed model on the CXR dataset (b) The training and validation processes of the proposed model on the CT dataset (c) The confusion matrix of the proposed model on the CXR dataset (d) The confusion matrix of the proposed model on the CT dataset

The SGDM optimizer can find a less minimum without overshooting on fewer epochs. Unlike the SGD optimizer, the SGDM optimizer tends to move in one direction to avoid oscillations. The Adam optimizer is another optimizer that determines the adaptive learning rate of first and second moments for each parameter. It also decreases the learning rates. Adam optimizer can be viewed as a combination of momentum and RMSprop. It also carries out the exponential moving gradients mean to update the learning rate instead of a simple average as in RMSprop. It maintains an exponentially decaying average of previous gradients, and is computationally effective with little memory specifications.

7 Conclusions and Future Work

This paper revealed the benefits of using different activation functions and optimizers to build a model capable of identifying COVID-19 cases based on CXR and CT images. Three optimization algorithms, namely SGDM, RMSprop, and Adam, have been studied. These optimizers are often described as adaptive optimizers, because the learning step is modified corresponding to the contour topology. Out of the above three algorithms, it is found that the SGDM is the best algorithm. Simulation results revealed that all algorithms can converge to various optimal local minima offered by the same loss. Adam optimizer combines the best attributes of the momentum and RMSprop algorithms. It is relatively easy to configure and it can handle sparse gradients. The simulation outcomes demonstrated that the proposed deep CNN approach is valuable and cost-effective in discovering COVID-19 cases. The simulation findings can be enhanced for a future plan, when acquiring massive CXR images and CT images.

Acknowledgement: The authors would like to acknowledge the support received from Taif University Researchers Supporting Project Number (TURSP-2020/147), Taif University, Taif, Saudi Arabia.

Funding Statement: This work was funded and supported by the Taif University Researchers Supporting Project Number (TURSP-2020/147), Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

- [1] K. Sharma, R. Sharma and V. Sharma, "Corona virus epidemiology: A review article," *International Journal of Community Medicine and Public Health*, vol. 7, no. 12, pp. 5219–5224, 2020.
- [2] C. Huang and Y. Wang, "Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China," *The Lancet*, vol. 395, no. 10223, pp. 497–506, 2020.
- [3] T. Singhal, "A review of coronavirus disease-2019 (COVID-19)," *Indian J. Pediatrics*, vol. 87, no. 4, pp. 281–286, 2020.
- [4] C. Lai, T. Shih, W. Ko, H. Tang and P. Hsueh, "Severe acute respiratory syndrome coronavirus 2 (SARS-Cov-2) and coronavirus disease-2019 (COVID-19): The epidemic and the challenges," *Int. J. Antimicrob. Agents*, vol. 55, no. 3, pp. 1–15, 2020.
- [5] F. Wu, S. Zhao and B. Yu, "A new coronavirus associated with human respiratory disease in China," *China Nature*, vol. 579, no. 7798, pp. 265–269, 2020.
- [6] C. Bircanoğlu and N. Arıca, "A comparison of activation functions in artificial neural networks," in *Proc. 26th Signal Processing and Communications Applications Conf. (SIU)*, Izmir, Turkey, pp. 1–4, 2018.
- [7] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation functions: Comparison of trends in practice for deep learning," *Journal of Neural Engineering*, vol. 16, no. 3, pp. 1–14, 2019.
- [8] D. Dong, Z. Tang, S. Wang, H. Hui, L. Gong *et al.*, "The role of imaging in the detection and management of COVID-19: A review," *IEEE Review in Biomedical Engineering*, vol. 5, no. 2, pp. 1–19, 2020.
- [9] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang *et al.*, "Artificial intelligence distinguishes COVID-19 from community acquired pneumonia on chest CT," *Radiology*, vol. 7, no. 4, pp. 1–14, 2020.
- [10] F. Shi, J. Wang, J. Shi, Z. Wu, Q. Wang *et al.*, "Review of artificial intelligence techniques in imaging data acquisition, segmentation and diagnosis for COVID-19," *IEEE Reviews in Biomedical Engineering*, vol. 4, no. 7, pp. 1–19, 2020.
- [11] A. Narin, C. Kaya and Z. Pamuk, "Automatic detection of coronavirus disease (COVID-19) using x-ray images and deep convolutional neural networks," *Pattern Analysis and Applications*, vol. 2, no. 1, pp. 1–21, 2021.
- [12] W. Kong and P. Agarwal, "Chest imaging appearance of COVID-19 infection," *Radiology Cardiothoracic Imaging*, vol. 2, no. 1, pp. 819–824, 2020.
- [13] C. Zheng, X. Deng, Q. Fu, Q. Zhou, J. Feng *et al.*, "Deep learning-based detection for COVID-19 from chest CT using weak label," medRxiv, 2020.
- [14] X. Xu, X. Jiang, C. Ma, P. Du, X. Li *et al.*, "Deep learning system to screen coronavirus disease 2019 pneumonia," *Engineering*, vol. 6, no. 10, pp. 1122–1129, 2020.
- [15] M. Barstugan, U. Ozkaya and S. Ozturk, "Coronavirus (COVID-19) classification using CT images by machine learning methods," *International Journal of Imaging Systems and Technology*, vol. 31, no. 1, pp. 5–15, 2021.
- [16] X. Chen, L. Yao and Y. Zhang, "Residual attention U-net for automated multi-class segmentation of COVID-19 chest CT images," *IET Image Processing*, vol. 10, no. 2, pp. 1–14, 2021.

- [17] P. Sethy and S. Behera, "Detection of coronavirus disease (COVID-19) based on deep features," *International Journal of Mathematical, Engineering and Management Sciences (IJMEMS)*, vol. 8, no. 2, pp. 1–17, 2020.
- [18] L. Wang and A. Wong, "COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest radiography images," *Sci. Rep.*, vol. 10, no. 1, pp. 1–12, 2020.
- [19] E. Hemdan, M. Shouman and M. Karar, "A framework of deep learning classifiers to diagnose COVID-19 in X-ray images," *Complex & Intelligent Systems*, vol. 7, no. 1, pp. 235–247, 2021.
- [20] Y. Song, S. Zheng, L. Li, X. Zhang, Z. Huang *et al.*, "Deep learning enables accurate diagnosis of novel coronavirus (COVID-19) with CT images," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 5, no. 2, pp. 1–14, 2021.
- [21] S. Wang, B. Kang, J. Ma, X. Zeng, M. Xiao *et al.*, "A deep learning algorithm using CT images to screen for corona virus disease (COVID-19)," *European Radiology*, vol. 5, no. 2, pp. 1–9, 2021.
- [22] D. Ioannis and B. Tzani, "Automatic detection from X-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, vol. 43, no. 2, pp. 635–640, 2021.
- [23] M. Holshue and C. DeBolt, "First case of 2019 novel coronavirus in the United States," *New England Journal of Medicine*, vol. 3, no. 1, pp. 1–14, pp. 2020.
- [24] S. Wang, B. Kang, J. Ma, X. Zeng, M. Xiao *et al.*, "A deep learning algorithm using CT images to screen for corona virus disease (COVID-19)," *European Radiology*, vol. 4, no. 2, pp. 1–9, 2021.
- [25] T. Ozturk, M. Talo, E. Yildirim, U. Baloglu, O. Yildirim *et al.*, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," *Computers in Biology and Medicine*, vol. 121, no. 4, pp. 235–245, 2020.
- [26] N. Alsharman and I. Jawarneh, "Googlenet CNN neural network towards chest CT-coronavirus medical image classification," *Journal of Computer Science*, vol. 16, no. 5, pp. 620–625, 2020.
- [27] P. Kingma and B. Jimmy, "Adam: A method for stochastic optimization," *Scientific Reports*, vol. 11, no. 1, pp. 1–8, 2019.
- [28] J. Reddi, S. Kale and S. Kumar, "On the convergence of adam and beyond," arXiv preprint arXiv: 1904.09237, 2019.
- [29] W. El-Shafai and F. Abd El-Samie, "Extensive COVID-19 X-ray and CT chest images dataset," *Mendeley Data*, v3, [Online]. Available: <http://dx.doi.org/10.17632/8h65ywd2jr.3>, 2020.