Identification of COVID-19 from Lung Ultrasound (LUS) images using deep transfer learning - Experimental study

[1]Chandrika Dadhirao, [2]K. Rasool Reddy, [3]Ram Prasad Reddy Sadi, [4]Raj Kumar Batchu, [5]K. Naga Prakash, [6]Ravi Kumar Vuddagiri

[1,3]Department of CSE, Gandhi Institute of Technology and Management, Deemed to be University Vishakhapatnam, India
[2]Department of ECE, NRI Institute of Technology (Autonomous), Vijayawada, India
3Department of IT, Anil Neerukonda Institute of Technology and Sciences,Visakhapatman, India
[4]Department of CSE, School of Computing Amrita Vishwa Vidyapeetham, Amaravti Campus, India
[5]Department of ECE, Seshadri Rao Gudlavalleru Engineering College (SRGEC), Gudlavalleru, Vijayawada, India
[6]Department of ECE, Koneru Lakshmaiah Educational Foundation, Hyderabad, Telengana, India

Corresponding author: rasool.ellora@gmail.com

ABSTRACT The COVID-19 pandemic has had a devastating impact on global health, economies, and societies. Early detection of COVID-19 is crucial to prevent transmission and reduce mortality, but conventional imaging techniques such as X-rays and computed tomography (CT) scans have limitations in accessibility, cost, and sterilization. Therefore, in this study, we explore the use of lung ultrasound (LUS) for COVID-19 diagnosis and evaluate the performance of various deep learning (DL) models such as AlexNet, ResNet, DenseNet, Inception, VGG, Inception-ResNet, MobileNet, and Xception with transfer learning. In the presented study, initially, we collected 455 COVID-19 and 226 non-COVID images, including bacterial pneumonia and healthy subjects, from a POCOVID-Net database. However, DL networks demand more data to explore and develop a significant model, so we employ data augmentation using geometric transformations. After that, we utilize the suggested deep transfer learning architectures for identifying COVID-19 subjects from LUS images. Finally, we estimate the performance of these models by well-received metrics such as sensitivity, specificity, precision, F1-score, the area under the curve (AUC), and accuracy. From the experimental results, we observed that DenseNet-201 achieved 100% accuracy and outperformed other models. This indicates that deep learning with transfer learning is a promising approach for COVID-19 identification from LUS data when data is scarce. These findings could have important implications for improving the efficiency and accessibility of COVID-19 diagnosis, particularly in resource-limited settings.

INDEX TERMS COVID-19, DL, data augmentation, transfer learning, CT, X-ray, and LUS imagery.

I. INTRODUCTION

Coronaviruses are a diverse group of respiratory viruses that may cause everything from a typical cold to severe respiratory syndromes, such as MERS (Middle East respiratory syndrome) and SARS (severe acute respiratory syndrome). In many cases, these viruses may adapt and infect people, allowing them to spread quickly. Coronavirus disease 2019 (COVID-19) first appeared in humans toward the end of 2019 as a novel strain of coronavirus [1]. The first instance was identified at the end of December 2019 in Wuhan city, China's provincial capital, and spread rapidly to several countries worldwide. As proof of the disease's fast spread, there were approximately 4,600 confirmed cases of COVID-19 in various countries on January 28, 2020, with 106 deaths. These figures climbed to 49,053 cases and 1,381 fatalities by February 15 (less than a month). In March 2020, Italy will be the worst-affected country in Europe. On April 5, 2020, Italian authorities recorded approximately 15.9 thousand deaths, among which 8.9 thousand occurred in the Lombardia region, 2.1 thousand in the Emilia-Romagna region, and 1.2 thousand in the Piedmont area. By April 2020, the number of coronavirus-related deaths in Italy surpassed that reported in China. According to the World Health Organization (WHO) guidelines, Wuhan was quarantined on January 23, 2020 [2], to restrict the spread of the virus by suspending public transportation. The next day, similar restrictions were extended to the nearby towns of Huanggang, Ezhou, Chibi, Jinzhou, and Zhejiang. Later, they enforced the same rules and regulations in several countries, including Europe, India, and the United States of America (USA). Table 1 illustrates the statistics of the top 20 COVID-19-infected countries by April 06, 2023, 13:59 GMT [3].

TABLE 1: COVID-19 STATISTICS OF THE TOP-20 COUNTRIES

S.No	Country	Total cases	Total deaths	Total Recovered	Active Cases	Total Test
1	USA	106,305,779	1,156,300	104,088,992	1,060,487	1,175,014,579
2	India	44,739,054	530,929	44,182,538	25,587	922,164,863
3	France	39,817,657	165,794	39,517,588	134,27	271,490,188
4	Germany	38,366,479	171,279	38,110,400	84,800	122,332,384
5	Brazil	37,319,254	700,556	36,249,161	369,537	63,776,166
6	Japan	33,500,042	74,029	21,722,879	11,703,134	98,535,136
7	South Korea	30,883,824	34,309	30,664,734	184,781	15,804,065
9	United Kingdom	24,448,729	209,396	24,221,314	18,019	522,526,476
10	Russia	22,689,110	397,459	22,060,308	231,343	273,400,000
11	Turkey	17,042,722	101,492	16,639,596	301,634	162,743,369
12	Spain	13,798,747	120,426	13,645,949	32,372	471,036,328
13	Vietnam	11,527,497	43,186	10,615,028	869,283	85,826,548
14	Australia	11,327,773	19,856	11,264,265	43,652	78,835,048
15	Taiwan	10,239,998	19,005	10,187,477	33,516	30,742,304
16	Argentina	10,044,957	130,472	9,914,485	0	35,716,069
17	Netherlands	8,610,372	22,992	8,575,974	11,406	25,984,435
18	Iran	7,592,255	145,391	7,342,490	104,374	55,353,767
19	Mexico	7,550,548	333,570	6,791,127	425,851	19,818,207
20	Indonesia	6,749,564	161,044	6,582,409	6,111	114,158,919

A. Variants

Viruses consistently change over genetic mutations, which can generate a new virus variant. Suppose a virus has one or more mutations; then, it is a variant of the original virus. The Centers for Disease Control and Prevention (CDC) and SARS-CoV-2 Interagency Group (SIG) identified various types of COVID-19 variants, represented in Table 2.

TABLE 2: VARIANTS OF CORONAVIRUS [4]

S.No	WHO Label	Lineage	Date of Identification and Country
1	Alpha	B.1.1.7	November 2020 and Southeastern England (UK)
2	Beta	B.1.351	October 2020 and South Africa
3	Gamma	P.1	January 2021 and Brazil
4	Delta	B.1.617.2	December 2020 and India
5	Epsilon	B.1.427	July 2020 and USA

		B.1.429	
6	Eta	B.1.525	December 2020 and UK, Nigeria
7	Lota	B.1.526	November 2020 and US (New York)
8	Kappa	B.1.617.1	December 2020 and India
9	Zeta	P.2	November 2020 and Brazil
10	Mu	B.1.621, B.1.621.1	January 2021 and Colombia
11	Omicron	B.1.1.529, BA.1, BA.1.1, BA.2, BA.3, BA.4 and BA.5	November 2021 and Botswana

B. Symptoms

Symptoms of COVID-19 vary from person to person, and they range from mild to severe, which are tabulated in Table 3. Usually, these symptoms can begin at any time from 2-14 days after exposure to COVID-19. However, most people suffered from mild to moderate symptoms and healed without hospitalization.

TABLE 3: SYMPTOMS OF COVID-19 [5]

Mild Symptoms	Moderate Symptoms	Severe Symptoms
Headache	Fever	Breathing trouble
Throat and body pains	Cough	Loss of speech or
Color changing of toes	Fatigue	confusion
Diarrhea	Taste and smell loss	Difficulty in walking
Skin rashes		Bluish skin or lips
		Chest pain

C. Diagnosis

Detection of COVID-19 at the initial stage is essential since symptoms are very similar to other pulmonary diseases. So, we required the necessary diagnostic tools to confirm COVID-19. The US Food and Drug Administration (FDA) and CDC approved the following diagnostic tests [6-7]:

1) NUCLEIC ACID AMPLIFICATION TESTS (NAATS):

Real-time RT-PCR is a widely used molecular test in NATTs to identify the genetic material of COVID-19. Healthcare professionals use a fluid sample from the nose or throat, with results usually obtained within 15 minutes. However, it has limitations like longer turnaround times, high costs, and false negatives.

2) ANTIGEN TEST

The test detects proteins on the coronavirus's outer surface using a nasopharyngeal swab, with low cost and quick results. However, it has high false negatives. C-reactive protein (CRP) and medical imaging methods like X-rays, CT, and ultrasound are used for COVID-19 prognosis, with US imaging being the most cost-effective and portable.

3) PREVENTION MEASURES

To minimize the dissemination of COVID-19 along with other infections like fever, we need to take the following measures [8]:

- Wear a well-fitted mask (ex: N95), especially in public gathering spaces.
- Maintain at least more than 1.5 meters' distance between person-to-person.
- Wash the hands with soap at regular intervals, such as before eating, after the squeeze, after using the toilet, etc.
- Prevent intimate contact with persons who are ill.
- Regularly sanitize the commonly touched surfaces and items.
- Get vaccinated as early as possible if you are eligible, and strengthen the immune system by maintaining a proper diet, exercising, etc.
- Quarantine ourselves when we are unwell for at least one week.

4) IMPACT

The COVID-19 pandemic has severely impacted global health, economy, and food sectors, with 193 million people suffering from extreme hunger in 2021 and 235 million by 2022, with India being particularly affected.

5) HOSPITALITY AND TOURISM

The hospitality and tourism industry includes various businesses such as bars and restaurants, hotels, pubs, service apartments, and amusement parks. It is one of the sectors adversely affected by COVID-19, especially in Asia and Europe, because it contributes significantly to gross domestic product (GDP) growth. Based on the current statistics, this industry slowly bounced back after October 2021. However, due to the COVID-19 crisis, most people hesitate to travel internationally. Fig. 1 represents the opinion of people interested in traveling during COVID-19 [9].

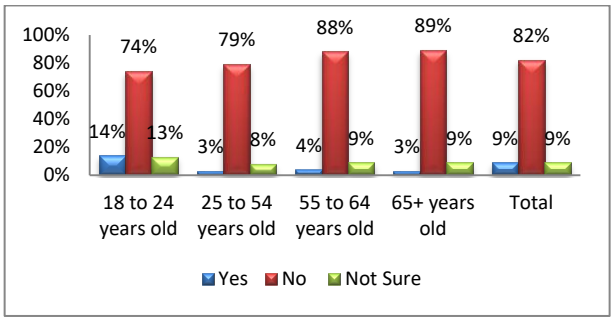


FIGURE 1. **Fig. 1. Graphical representation of people's views toward travelling.**

6) AUTOMOBILE

India is the third largest two-wheeler manufacturer and the fifth largest car manufacturer country globally, contributing approximately 7% of India's GDP. However, in the last 12-18 months, the growth of the automobile or automotive sector has been continuously impacted by axle-load reforms and goods and service tax (GST); in addition, the lockdown commenced by the government in March 2020. Therefore, there is a severe effect on financial companies, sales, suppliers, and auto dealers. Table 4 represents the car sales statistics before and after the coronavirus in 2019 and 2020. This table shows that most company sales decreased during COVID-19, except for Hyundai and Renault.

TABLE 4: PASSENGER CAR SALE REPORT BEFORE AND DURING COVID-19 [10]

Company	Pre-COVID (2019)	During COVID (2020)	Deviation
Maruti Suzuki	51.90	51.30	-0.6
Hyundai	16.40	17.60	+1.2
Mahindra	7.1	6.5	-0.6
Tata	6.3	4.8	-1.5
Toyota	4.5	4.1	-0.4
Honda	5.5	3.7	-1.8
Renault	2.4	3.2	+0.8
Ford	2.8	2.4	-0.4
Volkswagen	1	0.9	-0.1
Skoda	0.5	0.5	0.0

7) REAL ESTATE

The real estate industry, especially housing and commercial subsectors, faced many problems during the COVID-19 pandemic due to immense lockdown restrictions, skilled labor moving to their native places, and construction halting. As a result, companies and builders face many financial crises. Therefore, depending on the company's profits, they removed employees or reduced their salaries to come out of these crises. In the first wave of coronavirus, the sales of old launched housing projects declined by 67% in almost all major cities, except in Noida, whereas new housing projects also reduced by 78% from April-June 2020 [11]. In the same quarter, after one year, according to the survey report of 99 acres, the sales of housing projects drastically decreased by approximately 80%. Table 5 describes the percentage of sales down during April-June, 2020. Similarly, Table 6 illustrates the net revenue of the top 5 companies in India.

TABLE 5: SUMMARY OF THE DEALS FALLING DURING APRIL-JUNE, 2020 [11].

City	Percentage of sales fall
Gurugram	79
Chennai	74
Hyderabad	74
Bangalore	73
Kolkata	75
Mumbai	63
Pune	56
Thane	70

TABLE 6: NET REVENUE (IN $) OF TOP 5 COMPANIES IN INDIA [11].

Company	September 2019	December 2019	March 2020	June 2020
DLF	529.26	451.23	884.72	270.05
Phoenix Mills	434	525	409	147
Goderj Properties	80.62	275.54	780.83	52.27
Oberoi Reality	126.65	179.42	113.48	63.41
Prestige Estate	637.5	1019.7	926.3	685.1

8) EDUCATION

India's education sector, the second largest globally, has been severely impacted by the COVID-19 lockdown, affecting over 300 million students worldwide. The unemployment rate has risen from 8.4% to 23%, affecting employees and other sectors like agriculture, transportation, electronics, energy, and power. To minimize the impact, an automated COVID prediction system based on LUS images using deep learning models has been presented. This work discusses various existing models for predicting COVID-19 from LUS images.

II. Related Work

For the last two years, researchers have developed various methodologies for the early identification of COVID-19 using conventional machine learning and deep learning approaches [12]. Most of these findings utilized computed tomography (CT) and X-rays. However, prediction of COVID-19 from these imaging modalities remains challenging due to radiation involvement, inflexibility to some patients, especially pregnant women, and high cost. Therefore, recently radiologists have preferred the LUS imaging tool due to its flexibility, noninvasive nature, and reliable deployment, particularly in critical situations [13]. Based on this idea, authors recently focused on detecting COVID-19 from LUS data. This section discusses newly developed models and summarizes their findings, tabulated in Table 7.

Michael et al. [14] developed an optimized VGG 19 architecture to predict COVID-19 from CT, X-ray, and LUS images. Jannis et al. [15] suggested an automatic COVID-19 detection network, POCOVID-Net, based on LUS data. Zhang et al. [16] used VGG-19, ResNet 101, and Efficient B5 models to identify pneumonia from LUS images effectively. Julia et al. [17] adopted various pre-trained convolutional neural network (CNN) architectures, such as VGG19, InceptionV3, Xception, and ResNet50, to detect COVID-19 symptoms. Hui et al. [18] derived a multiscale residual CNN architecture to classify COVID-19 from non-COVID images. Awasthi et al. [19] proposed a lightweight CNN framework, namely mini-COVIDNet, for the early identification of COVID-19 using LUS imagery.

Salvia et al. [20] introduced a deep learning-based framework for detecting pneumonia and classifying the severity score of COVID-19. Born et al. [21] suggested a VGG16-CAM-based model for accurately distinguishing COVID-19 LUS images from bacterial pneumonia. Dastider et al. [22] initiated a hybrid model to predict the severity of COVID-19 with the help of CNN, autoencoders, and long short-term memory (LSTM). Muhammad et al. [23] developed a novel architecture by fusing the layers to classify COVID and non-COVID from LUS images. Bruno et al. [24] presented an integrated system using CNN and LSTM to detect COVID-19 using LUS videos.

By working on the prediction of COVID-19 from LUS images, a few computer-aided diagnosis (CAD) approaches have been developed using the concepts of CNN with transfer learning. However, most of the authors worked on only one optimization algorithm. Based on this idea, in this study, we extensively analyzed the performance of various pre-trained CNN models with respect to optimization techniques.

A. Highlights of the Study

1. We introduce, apply, and analyze the importance of AlexNet, ResNet-50, 101, 152, DenseNet-121, 169, 201, InceptionV3, VGG-16, 19, Inception-ResNet-V2, Xception, and MobileNetV2-based pre-trained deep learning models for automatic screening of COVID-19 from LUS data.
2. We employ image augmentation to increase the model's performance by minimizing overfitting.
3. We assess the performance of the presented models using various optimizations such as stochastic gradient momentum (SGDM), Adam, Adagrad, AdaMax, Adadelta, Nadam, and RMSProp. It is the significant difference between the existing and proposed study.
4. We demonstrate that the models used in this study significantly improve the classification accuracy by accurately differentiating COVID and non-COVID from LUS images and comparing them with other well-known methods.
5. The experimental results indicate that LUS is a feasible medical imaging tool for assisting COVID-19 when CT and X-rays are unavailable.

TABLE 7 SUMMARY OF THE EXISTING MODELS

Reference	Methodology	Classification Type (Multiclass/ Binary)	Data Augmentation	Findings

			(Yes/No)	
Michael et al. [14]	VGG-16	Binary	Yes	Accuracy = 98.54% (Healthy vs. COVID & Pneumonia) Accuracy = 100% (COVID vs. Pneumonia)
Jannis et al. [15]	POCOVID-Net	Multiclass	No	Accuracy = 89% (Healthy vs. COVID vs. Pneumonia)
Zhang et al. [16]	VGG-19, ResNet-101, and Efficient-B5	Multiclass	Yes	Accuracy = 95.5% (Healthy vs. COVID vs. Pneumonia) Accuracy = 89.1% (Healthy vs. COVID vs. Pneumonia)
Julia et al. [17]	VGG-19, InceptionV3, Xception, and ResNet-50	Binary and Multiclass	Yes	Accuracy = 93.4% (COVID vs. Pneumonia) Accuracy = 91.5% (COVID vs. Non-COVID)
Hui et al. [18]	Multiscale Residual CNN	Binary	No	Accuracy = 95.11% (COVID vs. Non-COVID)
Awasthi et al. [19]	Mini-COVIDNet	Multi-class	No	Accuracy = 83.2% (Healthy vs. COVID vs. Pneumonia)
Salvia et al. [20]	ResNet50	Multiclass	Yes	Accuracy = 98.43%
Born et al. [21]	VGG16-CAM	Multiclass	No	Accuracy = 90% (Healthy vs. COVID vs. Bacterial Pneumonia)
Muhammad et al. [22]	Multi-layer Fusion	Multiclass	Yes	Accuracy = 92.5% (Healthy vs. COVID vs. Pneumonia)
Bruno et al. [23]	CNN-LSTM	Multi-class	No	Accuracy = 93% (Healthy vs. COVID vs. Bacterial Pneumonia)

III. Materials and Methods

Fig. 2 represents the flow diagram of the suggested model, which includes five phases: dataset collection, image data augmentation, deep learning models, transfer learning (TL) and fine-tuning, and performance evaluation.

A. Dataset

To evaluate the performance of the proposed models, we collected the LUS images from a publicly available database, namely POCOVID-Net [15]. Here, images are acquired by sampling the video sequences gathered from various sources with a frame rate of 3 Hz and 17 frames per video. We obtained 455 COVID-19 and 226 non-COVID subjects, including bacterial pneumonia and healthy subjects. However, this data may not be sufficient for developing a better predictive model. Hence, we employed data augmentation.

B. Image Data Augmentation

The performance of deep learning networks (DNNs) heavily relies on the size and quality of the training database. For example, during the training progress of a DNN, we need more training data, which leads to overfitting problems. Therefore, image data augmentation is vital in improving model prediction accuracy and the model's generalization ability. Image data augmentation means the artificial creation of images using traditional image processing operations such as rotation, scaling, reflection, translation, shearing, etc. The data augmentation operations and the corresponding results are depicted in Table 8. We finally obtained 3185 COVID-19 and 1582 non-COVID LUS images with these operations.

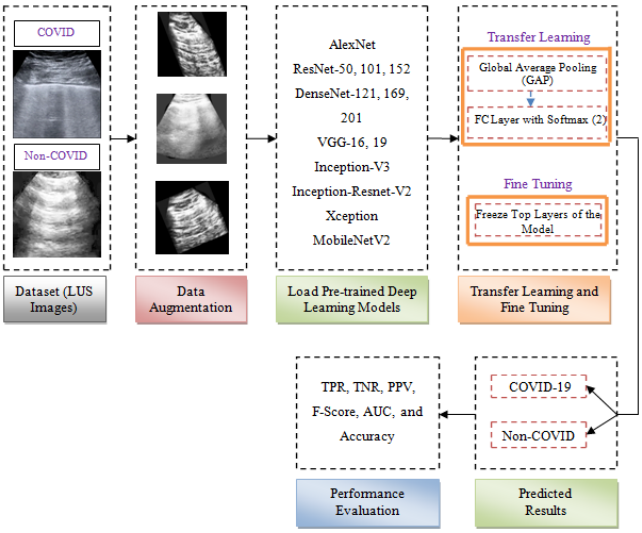


FIGURE 2. Stepwise diagram of the proposed framework.

The database is organized into two folders (training and testing) containing COVID and non-COVID subfolders. Table 9 represents the number of images for each class in a given folder, and Fig. 3 illustrates the sample images used in this article. Afterward, we deployed various pre-trained models on these augmented images by employing transfer learning and the necessary fine-tuning process.

TABLE 8 IMAGE DATA AUGMENTATION OPERATIONS UTILIZED IN THIS WORK

Augmentation operator	Value
Rotation	Randomly from -35^0 to 35^0
Translation	Translate along X- (horizontal) and Y- (vertical) direction with a range of [-10 10]
Reflection	Reflect randomly along X- and Y- direction
Scale	Uniform scaling with a range of [0.5 5]
Shear	Shearing along vertical and horizontal with a range of [0 45^0]

TABLE 9 NUMBER OF SAMPLES FOR EACH CLASS IN THE COVID-19 LUS DATASET

Database	Training set		Testing set	
	COVID	Non-COVID	COVID	Non-COVID

COVID-19 LUS	364	181	91	45
COVID-19 LUS + Augmentation	2548	1265	637	317

C. Deep Learning Models

1) BACKGROUND

Deep learning (DL) architectures can learn complex tasks by hierarchically constructing feature maps. Among all the available DL models, CNN-based methods are more popular and have the following layers: convolutional, pooling, activation, batch normalization, fully connected (FC), dropout, and softmax.

2) CONVOLUTIONAL LAYER

Among all the layers of CNN, the convolutional layer is a crucial aspect, particularly in tumor identification scenarios. The convolutional layer builds feature maps by accumulating the layers over each other in a hierarchical manner. Note that each convolution layer takes feature maps as input from its previous layer, except the first convolutional layer, since it is directly associated with the input image. Usually, the convolutional layer produces many feature maps as output.

The convolutional layers generate feature maps, F by convolving the input image with the corresponding nonlinear feature generators; namely, the filter kernel in a sliding window manner using Eq. (1), and the tiny size of the rectangular box illustrates them.

$$F(m,n) = \sum_{u}\sum_{v} C(u,v)K(m-u,n-v) \tag{1}$$

where C denotes the LUS image; K represents the convolutional filter kernel; u and v are the dimensions of image C, while m and n give the dimensions of the generated feature map. The kernels in a convolutional layer behave like edge detectors, learning distinct features by the characteristics of training image data.

4) BATCH NORMALIZATION

Normalization is mainly used to normalize the features obtained from a convolutional layer. In this work, we used batch normalization, also called the batch norm. The significant advantage of the usage of a batch norm is as follows:

1. Improving the training speed of the network.
2. Enhance the network's performance by smoothing the objective function [24].
3. Minimize the internal covariance shift [25].
4. Reduces the over-fitting since it has a slight regularization effect.

The entire procedure involved in the batch norm is illustrated in Algorithm 1.

Algorithm 1: Batch normalization

Input: Values of F over a mini-batch: $b = \{F_1, F_2, \ldots, F_K\}$;

Parameters to be learned: γ, ξ.

Output: $b_n(F_j)$

$$\mu_b = \frac{1}{M}\sum_{j=1}^{M} F_j, \tag{2}$$

$$\sigma_b^2 = \frac{1}{M}\sum_{j=1}^{M}\left(F_j - \mu_b\right)^2, \tag{3}$$

$$F_j = \frac{F_j - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}}, \tag{4}$$

$$b_n(F_j) = \gamma F_j + \xi, \tag{5}$$

where γ represents scale; ξ illustrates shift; K is the number of feature inputs; μ and σ^2 are the mean and variance across the batch, b; ϵ is a constant, which is used to enhance the stability when σ_b^2 is too small.

5) ACTIVATION LAYER

Let us assume a neural network (NN) without an activation function. Each neuron will conduct only a linear transformation to the inputs in that scenario based on the bias and weights. Despite this, the linear transformation makes the network simpler. However, the corresponding network will not be able to learn complex tasks. Hence, to limit these issues,

introduce a nonlinear transformation into the output of each neuron and achieve it by action functions. Due to these activation functions, the network will be able to learn many complex tasks by making use of important information and minimizing inappropriate data points. Among several activation functions, ReLU and its variants, such as ELU, PReLU (probabilistic ReLU), and leaky ReLU, are widely used in CNNs since they significantly overcome vanishing gradient problems.

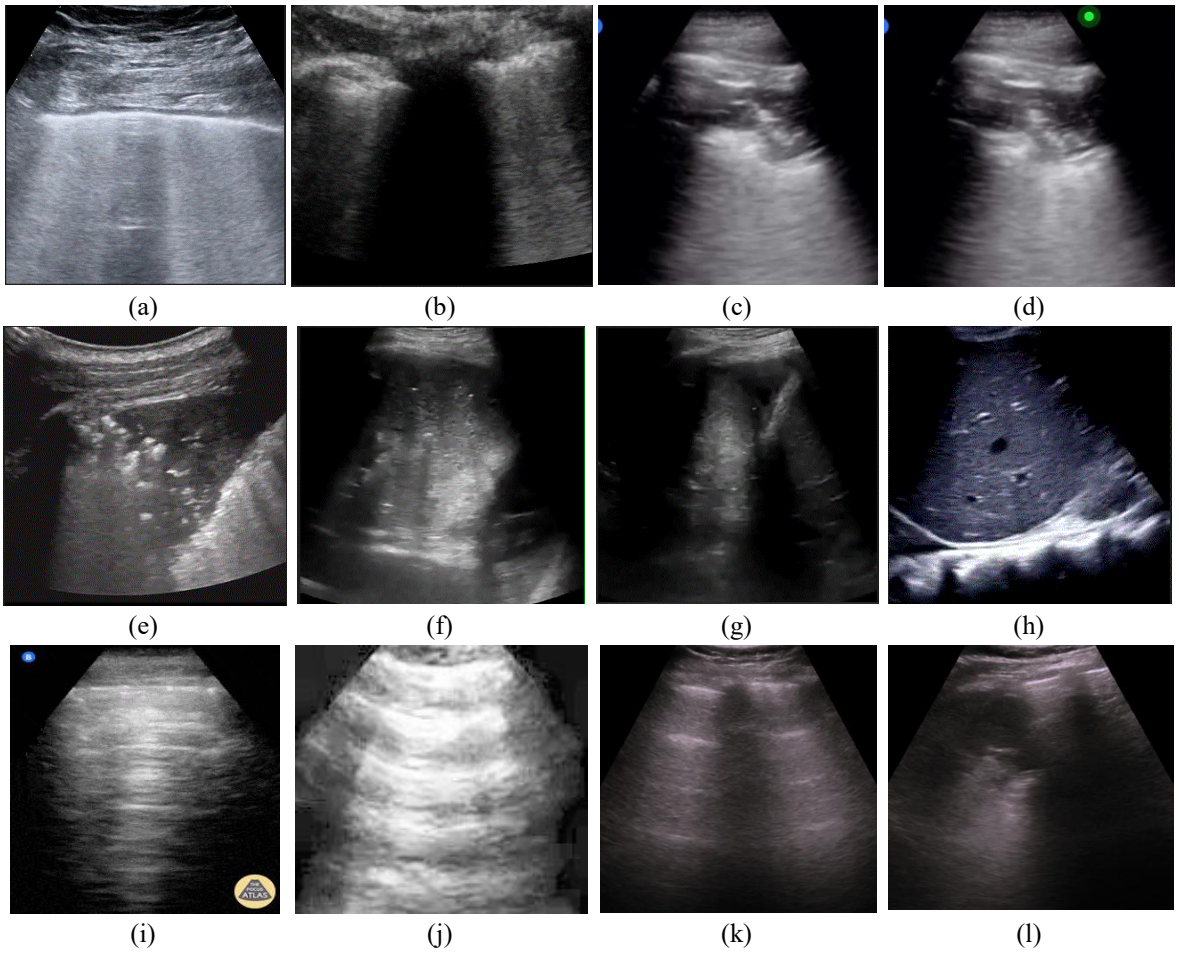


FIGURE 3. **Sample LUS images used in this article: (a)-(d) COVID-19; (e)-(h) Bacterial Pneumonia; (i)-(l) Healthy**

6) POOLING LAYER

Pooling is another vital aspect of CNN and typically occurs after the convolutional layer. The main objective of pooling is as follows:

- It shrinks the spatial size of feature maps. As a result, it minimizes the number of parameters to learn and the network's computational time.
- As we know, the convolutional layer is effectively generated from its previous layers. However, these feature maps are sensitive to variations in the position of the features of the input space. Due to this, we will obtain different feature maps. To avoid this, we perform pooling by summarizing the features present in the patches of the feature map.

Average and max-pooling are the most frequently used pooling approaches [26] and are represented in Fig. 4. After performing pooling, the size of the feature map can be reduced from $F \times F \times f_c$ to $W \times W \times f_c$ using Eq. (6). Here, f_c defines the number of filters; F and W represent the size of a feature map before and after performing a pooling operation.

$$W = \left(\frac{(F_m - Z)}{D} + 1\right) \times \left(\frac{(F_n - Z)}{D} + 1\right) \tag{6}$$

where F_m and F_n illustrate the height and width of the feature map, F ; Z indicates the padding size; D describes the length of the stride.

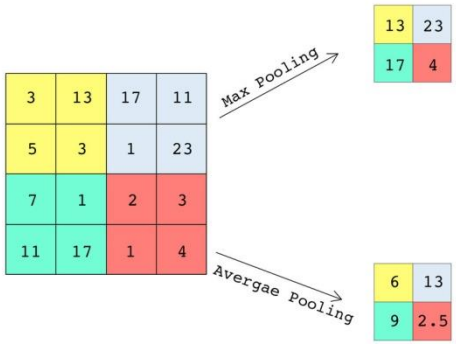


FIGURE 4. **Average and max-pooling with a filter kernel of 2×2 and stride of [2 2]**

7) FULLY CONNECTED OR DENSE LAYER

The fully connected (FC) or dense layer is a crucial element of CNN and has proven to be very successful in identifying and classifying images, particularly in computer vision applications. The FC layers are input from previous layers, either the convolutional or final pooling layer and then transformed into a single feature vector, s by flattening. Note that the number of FC layers varies with the model in which you are trained. Therefore, in the proposed model, the output value of the FC layer is set to 2 since we have two classes.

8) DROPOUT LAYER

DNNs include numerous hidden layers of nonlinear nature, building a significant model that can effectively learn complex relationships between their inputs and outputs. However, when dealing with limited data, these complex relationships can result in sampling noise in the training database and overfitting problems. Several approaches have been implemented to minimize the issue, as mentioned above [27]; the dropout layer is typically used in DNN [28]. The main idea is to randomly nullify or drop some hidden layers during the training progress, which will result in reducing the co-adaption.

9) SOFTMAX LAYER

Generally, the softmax activation function takes place at the end of the neural network to transform the features into class probabilities. The softmax yields a value for each class based on the probabilities' computation using Eq. (7)

$$P(y=i/s)=\frac{e^{s^T w_i}}{\sum_{i=1}^{N} e^{s^T w_i}}, \tag{7}$$

where, s is the feature vector, T indicates the transpose operator, w illustrates the weight vector, P is the predicted probability of the $i-th$ class and finally, N represents the number of classes. In this work, N will be chosen as 2 since we perform binary classification. Based on these layers, researchers have developed numerous deep-learning networks [29]. We adopted the following DL models (section 3.3.2) from the literature to detect COVID-19 from LUS imagery, and they are summarized in Table 10.

D. Pretrained DL architectures

1) ALEXNET

AlexNet [30] is a popular and widely used CNN architecture in deep learning, especially in the machine and computer vision applications. It won the Image Net Large Scale vision recognition challenge 2012 competition (ILSVRC 2012) [31] with a 15.3% error rate. This network was trained on 12 lakhs of high-resolution natural images with a size of 227×227 into 1000 distinct categories with 60 million parameters, 6.5 lakhs of neurons, and 0.63 billion connections. AlexNet comprises five convolutional layers with filter sizes 11×11, 5×5, and 3×3, overlapping max-poling layers of filter size 3×3 with stride two at each of three convolutional layers, and three fully-connected layers with 4096 and 1000 neurons. Note that each convolution layer is equipped with a ReLU actuation layer.

2) RESNET

Generally, deeper networks are more capable of learning highly complex tasks. However, many researchers have identified that after some depth, the performance of the deeper networks degrades when the deeper networks begin to converge. With this, the deeper networks lead to higher training errors. To limit this problem, a deep residual network, namely ResNet, was proposed [32]. ResNet stands for residual networks and is a backbone for many computer and machine vision applications. In this model, we used skip connections to resolve the exploding gradient problem or minimize the training error while increasing the number of layers. Hence, it won first place in the ILSVRC 2015 competition with an error rate of 3.5%, and it also achieved first place in the COCO 2015 challenging round on the COCO object detection database [33] with an improvement of 28% accuracy. The main advantages of ResNet are

- Easy to optimize compared to the other “plain” networks.
- Quickly gain accuracy from highly deeper networks and yield better performance than other networks.

The available ResNet variants are 18, 34, 50, 101, and 152. However, in this work, we utilized ResNet-50, 101, and 152 to detect COVID-19 from LUS data.

3) DENSENET

Many researchers have identified that vanishing-gradient issues might arise as the deeper network's depth increases. Therefore, as mentioned earlier, ResNet was proposed in 2015 to address this problem [32]. However, the use of skipping connections in the residual learning approach reduces the learning capacity of the model [34]. Hence, to mitigate this problem, the authors [35] proposed the DenseNet architecture. DenseNet is a recently developed DNN architecture suggested by the corn well and Tsinghua universities, along with Facebook AI research in 2017. Due to its dense connections, DenseNet gives surprising results on widely used object detection databases, namely CIFAR-10, CIFAR-100, and ImageNet 2015.
The main features of Dense Net are as follows:

- Each layer is connected to all other preceding layers in a feed-forward manner.
- Enhance the feature propagation in both forward and backward computation.
- Support (or) Encourage feature reuse.
- Enhance the performance of detection and classification with less computational complexity due to feature-map concatenation.
- Require fewer parameters to realize the architecture.

DenseNet consists of transition layers, dense blocks, convolutional layers, max, average, global-averaging pooling layers, fully connected layers, and a softmax layer. Each transition layer comprises a 1×1 convolutional layer followed by average pooling with a 2×2 filter size and stride 2. DenseNet has various versions; we utilized DenseNet-121, 169, and 201.

4) VGGNET

The VGG [36] is a very deep CNN architecture developed by K. Simonyand and Zisserman in 2014 from Oxford University, which stands for "Visual Geometry Group." In this model, rather than large filter kernels (11×11 and 5×5 in [30]), they utilize tiny filter kernels such as 3*3 throughout the network. Furthermore, they also introduce a 1×1 convolution filter followed by a ReLU activation function. Thus, we can reduce the number of trainable and non-trainable parameters, which improves the training speed. It is the significant advantage of VGG over AlexNet [30] and attained 92.7% of the top-5 test classification accuracy. There are different types of VGGNet configurations: VGG-11, 13, 16, and 19. This article considers the VGG-16 and VGG-19 models to detect COVID-19 from the LUS image database. The VGG-16 has 16 layers: thirteen convolution layers (ten 3×3 and three 1×1) and three dense layers, whereas VGG-19 includes sixteen convolutional and three dense layers.

5) MOBILENETV2

MobileNetV2 [37] is a newly developed mobile network architecture for next-generation mobile computing applications such as classification, object recognition, and semantic segmentation. Usually, MobileNetV2 is an enhanced version of mobile network architecture such as MobileNetV1 [38]. In MobileNetV2, they introduced an inverted residual structure to reduce computational cost and the size of the mobile model network. In addition, it also eliminates non-linearity in narrow layers. Due to this, the network is adequate for mobile (or) devices with less computational power. MobileNetV2 mainly consists of two different types of blocks. The first is a residual block with stride one and a residual block with stride two for downsizing. Each block has three layers:

- 1×1 convolutional layer followed by ReLu6.
- Depth-wise, 3×3 convolutional layer along with ReLu6.
- 1×1 convolution layer with linearity.

6) XCEPTION

Xception [39] is a 71-layer deep CNN architecture with depthwise separable convolution. It was introduced by a Google researcher, Francois Chollet, in 2017 by inspiring the Inception network, which stands for "Extreme Inception." The fundamental difference between Xception and Inception is as follows:

- **Order of convolution operations:** In the Inception model, we initially perform an 11 convolution (a pointwise convolution) and then implement depthwise convolution (spatial convolution) independently over each input channel. However, in Xception, we first perform spatial convolution across channels and then apply pointwise convolution.
- **Presence of ReLU activation:** In Xception, we do not introduce any nonlinearity after each convolution operation, whereas, in Inception, we employ ReLU activation after both convolutions.

Based on the above remarks, we observed that the Xception network performs slightly better than the Inception model on the ImageNet 2015 dataset.

7) INCEPTIONV3

The Inception networks are deep CNN architectures, and they were developed by Google researchers in 2015. There are three versions of Inception networks, namely InceptionV1 (GoogleNet), V2, and V3. Among them, InceptionV3 [40] performs significantly better than other versions since it yields a low-error rate and low-computational cost. InceptionV3 is a 42-layer deep learning network, and it was the first runner-up in the ImageNet 2015 challenging competition. By analyzing all versions of Inception models, versions V2 and V3 have similar features but slight modifications in InceptionV3, and they are:

- Factorization into smaller convolutions.
- Factorization into asymmetric convolutions.
- Usage of auxiliary classifiers.
- Efficient grid size reduction.

8) INCEPTION-RESNET-V2

The Inception-ResNet [41] is a 164-layer deep CNN model, and it was introduced in 2016 by Google with a combination of Inception and residual frameworks. Due to this, we can speed up the Inception network's training progress and significantly reduce the training error compared to other models such as Inception V1, V2, and V3. They suggested two residual network-based Inception models, such as Inception-ResNet-V1 and Inception-ResNet-V2. In this work, we adopted Inception-ResNet-V2 to classify COVID-19 and non-COVID from LUS data.

The main features of this architecture are as follows:

- Computationally less expensive
- Utilize filter expansion
- It does not employ batch normalization after summations.
- Replace the pooling layers used in the Inception block with residual connections.

The presented DL models are evaluated on the ImageNet dataset, which includes a thousand classes. However, in our study, the data is limited due to the expensive cost of acquisition, the scarcity of diseases, and ethical and legal issues. Therefore, furthermore, we employ transfer learning (TL) on the suggested DL architectures due to feature reuse capability, which improves the detection rate of COVID-19 from LUS imagery. In the following section, we briefly summarize the significance of TL and its process in detail.

TABLE 10 SUMMARY OF THE IMPLEMENTED CNN MODELS

Network	Layers	Freezing the layers (Yes/No)	GAP (Yes/No)	Trainable Parameters	Non-trainable parameters	Total Parameters
AlexNet	8	No	No	71,940,166	19,140	71,959,306
ResNet-50	50	No	Yes	23,538,690	53,120	23,591,810
ResNet-101	101	No	Yes	42,556,930	105,344	42,662,274
ResNet-152	152	No	Yes	58,223,618	151,424	58,375,042
DenseNet-121	121	No	Yes	6,955,906	83,648	7,039,554
DenseNet-169	169	No	Yes	12,487,810	158,400	12,646,210
DenseNet-201	201	No	Yes	18,096,770	229,056	18,325,826
InceptionV3	42	No	Yes	21,772,450	34,432	21,806,882
Inception-ResNet-V2	164	No	Yes	54,280,803	60,544	54,341,347
VGG 16	16	Yes	Yes	119,554,050	14,714,688	134,268,738
VGG 19	19	Yes	Yes	119,554,050	20,024,384	139,578,434
MoblieNetV2	53	Yes	Yes	125,442	2,257,984	2,383,426
Xception	71	No	Yes	20,811,050	54,528	20,865,578

E. Transfer Learning and Fine-Tuning

1) TRANSFER LEARNING

Transfer learning (TL) is a widely used machine learning approach in various classification problems, including natural language processing (NLP), computer vision, medical image processing, agriculture, etc. Generally, TL implies transferring knowledge from one task to another to enhance learning. Expressly, in deep learning, TL signifies transferring a pre-trained CNN model's learning parameters (weights and bias) to a new task of our model, which means that instead of training our model from scratch, we transfer the learned features. As a result, the learning process can be faster and more accurate, especially when dealing with a smaller database. Fig. 5 describes the working of transfer learning.

The proposed TL process includes the following sequence of steps:

- Choose one of the architectures from the existing CNN models and import the data from that architecture.
- We truncate the output layer (softmax or classification layer) in the selected model, replace it with our model output layer, and then utilize the rest of the network as a feature extractor for our model. For example, the classification layer comes with thousands of categories in the existed DL networks, but our model works on two classes (COVID vs. non-COVID). Therefore, the new classification layer of the model will be two classes instead of a thousand.
- Freezing layers: During training, we do not train the layers in the feature extraction, which leads to high classification accuracy.

The main benefits of TL in our framework are as follows:

- Conquers the data scarcity issues. In addition, we can save hardware resources and time.
- Minimize the data size.
- Lower distance between the source and target.

2) FINE-TUNING

Fine-tuning is one technique to transfer learning and behaves like an optimization. The main objective of fine-tuning is to improve the network's performance by minimizing the loss with the help of changing the number of layers and filters, learning rate, etc. In this work, to obtain the desired performance, we modify the learning rate of various optimization techniques (section 4) from 0.001-0.01, represented in Table 11.

TABLE 11: PARAMETER SETTINGS OF VARIOUS OPTIMIZATION TECHNIQUES

Optimizer	Parameters
SGD	α = 0.01
Adam	α = 0.001, β_1 = 0.9, β_2 =0.999, and ε = 1e-07
AdaMax	α = 0.002, β_1 = 0.9, β_2 =0.999, and ε = 1e-07
Adagrad	α = 0.001 and ε = 1e-07
Adadelta	α = 0.001, ε = 1e-07 and β = 0.95
RMSProp	α = 0.001, ε = 1e-07 and β = 0.9
Nadam	α = 0.001, β_1 = 0.9, β_2 =0.999, and ε = 1e-07

Note: α represents the learning rate; β_1, and β_2 are the decay factors; ε is the constant for numerical stability and usually takes a smaller value.

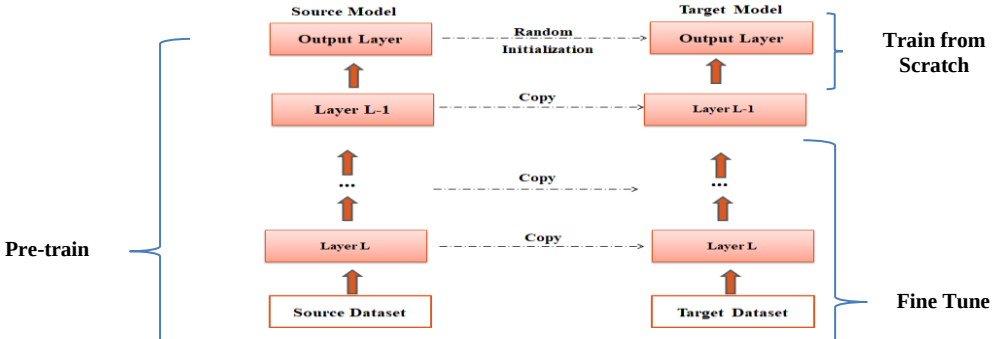


FIGURE 5. Working flow of transfer learning [42]

F. Evolution Criteria

The performance of the suggested approach is validated through the following widely used measures:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN} \tag{9}$$

$$\text{True Negative Rate (TNR)} = \frac{TN}{TN + FP} \tag{10}$$

$$\text{Positive Predictive Value (PPV)} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{F-Score} = 2\left(\frac{PPV \times TPR}{PPV + TPR}\right) \tag{12}$$

$$\text{Area Under Curve (AUC)} = \frac{TPR + TNR}{2} \tag{13}$$

where TP = true positive; FN = false negative; FP = false positive and TN = true negative.

G. Optimization Algorithms

Optimization algorithms play a pivotal role in enhancing the performance of neural networks by modifying the weights and learning rate of the model during the training progress, which results in minimizing the error or loss function. To achieve this, the authors developed various optimization algorithms [43]. Here, we discuss a few widely used approaches.

1) STOCHASTIC GRADIENT DESCENT

The traditional gradient descent (GD) approach is computationally expensive in each iteration when dealing with massive data and cannot be utilized for online learning. Therefore, stochastic gradient descent (SGD) was developed [44], one of the most popular variants of gradient descent. The main idea behind SGD is that instead of considering all samples for each iteration, we randomly choose one sample per iteration to update the gradient. Due to this, SGD significantly improves convergence efficiency and reduces the computational cost compared to GD. The mathematical expression for updating the gradient is as follows:

$$w' = w + \alpha\left(y^j - f_w\left(x^j\right)\right)x^j \tag{14}$$

where w is the mapping function parameter, x^j is the input feature vector of the $j-th$ sample, y^j represents the corresponding labeling of the $j-th$ sample, and $f(x)$ is the mapping function.

2) ADAPTIVE GRADIENT DESCENT

Adaptive gradient descent (Adagrad) [45] is also a gradient-based optimization algorithm, but there is a minute difference between them. In gradient descent, we used a fixed learning rate for all iterations, while in Adagrad; we dynamically changed the learning rate based on the gradients of previous iterations. The significant benefit of Adagrad is eliminating the manual tuning of the learning rate. The mathematical intuition behind Adagrad is as follows:

$$w_{i+1} = w_i - \alpha\left(\frac{g_i}{H_i}\right) \tag{15}$$

$$g_i = \frac{\partial L\left(w_i\right)}{\partial w} \tag{16}$$

$$H_i = \sqrt{\sum_{k=1}^{i}\left(g_k\right)^2 + \varepsilon} \tag{17}$$

where g_i denotes the gradient of w at iteration i, $L(w)$ is the loss function, H_i is the collection of previous gradients (or gradient history) of w at iteration i and w_i is the value of w at iteration i.

3) ROOT MEAN SQUARE PROPAGATION

Ideally, root mean square propagation (RMSProp) [46] is an extended version of Rprop [47] and solves the varying gradients problem. The issues with gradients, some of them were small, but others were massive, which was challenging. Thus, defining a single learning rate may not be the best action. Therefore, in Rprop, they utilized two gradients. If they have the same sign, increase the step size; if they have opposite signs, decrease the step size. However, Rprop performs poorly on larger datasets when we deal with mini-batch weight updates. Thus, the authors introduced RMSProp, which also considers the idea of the Adagrad optimization algorithm. The RMSProp optimizer primarily concentrates on speeding up the optimization process by lowering the number of function evaluations to meet the local minimum. The mathematical intuition behind RMSProp is as follows:

$$E\left[g^2\right]_i = \beta E\left[g^2\right]_{i-1} + (1-\beta)g_i^2 \tag{18}$$

$$w_i = w_{i-1} - \frac{\alpha}{\sqrt{E\left[g^2\right]_i}} g_i \tag{19}$$

3) ADADELTA

The main drawbacks of RMSProp and Adagrad optimizers are as follows:

- The initial learning rate must and should be manually set.
- Decaying learning rate issue.

Due to the earlier problems, the network may not learn new information/knowledge after a few iterations in both optimizers. Hence, Adadelta was introduced based on the adaptive learning concept [48]. In Adadelta, instead of considering the sum of all previous gradients, we employed an exponential moving average over a period in a given window by the following expression:

$$H_i = \sqrt{\beta H_{i-1} + (1-\beta)(g_i)^2} \tag{20}$$

4) ADAPTIVE MOMENT ESTIMATION

Adaptive movement estimation (Adam) [49] is an extension of SGD that utilizes different learning rates for each iteration. The Adam optimizer is developed by combining the features of both RMSProp and Adadelta, integrating moment and adaptive learning rate concepts. Unlike in Adadelta and RMSProp, the Adam optimizer also updates the learning rate based on the exponential moving average of past gradients (m_i) as follows:

$$w_{i+1} = m_i - \alpha\left(\frac{\sqrt{1-\beta_2}}{1-\beta_1}\right)\left(\frac{m_i}{H_i + \varepsilon}\right) \tag{21}$$

$$m_i = \beta_1 m_{i-1} + (1-\beta_1) g_i \tag{22}$$

$$H_i = \sqrt{\beta_2 H_{i-1} + (1-\beta_2) g_i^2} \tag{23}$$

5) MAXIMUM ADAPTIVE MOMENT ESTIMATION

The maximum adaptive moment estimation (AdaMax) [49] is a generalization or extension version of the Adam optimization algorithm. In the Adam optimizer, updated weights are inversely proportional to the scaled l_2 norm of present and past gradients, but in AdaMax, this was extended from the l_2 norm to the l_∞ norm. The mathematical expression for updating weights in AdaMax:

$$w_i = w_{i-1} - \left(\frac{\alpha}{1-\beta_1^i}\right)\left(\frac{m_i}{H_i}\right) \tag{24}$$

$$H_i = \beta_2^\infty H_{i-1} + \left(1-\beta_2^\infty\right)|g_i|^\infty = \max\left(\beta_2 H_{i-1}, |g_i|\right) \tag{25}$$

6) NESTEROV-ACCELERATED ADAPTIVE MOMENT ESTIMATION

In Adam, we use different learning rates during the training process; however, this may reduce the learning rate to a very small value. Hence, researchers proposed a Nesterov-accelerated moment Adam optimizer (Nadam) by incorporating the Nesterov-accelerated moment (NAG) into the Adam optimizer [50], which results in a lower-training time than Adam. In Nadam, the learning rate is updated by the following expressions:

$$w_{i+1} = w_i - \frac{\alpha}{\sqrt{H_i} + \varepsilon}\left(\beta_1 m_i + \frac{(1-\beta_1) g_i}{1-\beta_1^i}\right) \tag{26}$$

$$m_i = \frac{m_i}{1-\beta_1^i} \tag{27}$$

$$m_i = \beta_1 m_{i-1} + (1-\beta_1) g_i \tag{28}$$

$$H_i = \frac{H_i}{1-\beta_2^i} \tag{29}$$

$$H_i = \beta_2 H_{i-1} + (1-\beta_2) g_i^2 \tag{30}$$

IV. Results and Discussion

Medical imaging modalities, namely chest X-ray, CT, and LUS, play a pivotal role in validating the primary prognosis of COVID-19 from the RT-PCR test. These modalities are also crucial in monitoring disease advancement and patient care. Thus, extracting relevant features from imaging approaches is a critical phase of training deep learning models because a model's accuracy significantly relies on the features we extracted. Fig. 6 shows feature maps generated by the first three convolution layers of the AlexNet architecture. Among X-ray, CT, and LUS, we mainly focus on LUS imagery due to its low-cost and low harm, particularly for pregnant women.

In this research, we mainly focus on the comprehensive analysis of the classification of COVID and non-COVID from LUS imagery using various existing pre-trained CNN models. To evaluate the performance of these models, we split the dataset into 80% training and 20% testing. Table 12 represents the configurations utilized in the proposed models. The training and testing procedures of the suggested models were performed in Python 3 using a high-level application programming interface of TensorFlow, such as the Keras deep learning framework, and run on the Collaboratory (Colab) GPU accelerator developed by Google researchers with 25.45 GB RAM. All these operations were carried out on a Windows-based computer with Intel(R) Core(TM) i3-5005U CPU configuration @ 2.00 GHz with 12 GHz RAM.

TABLE 12 CONFIGURATIONS OF THE PROPOSED ARCHITECTURES

Optimizers	Learning Rate	Batch Size	Epochs
SGD, Adam, Adagrad, Adadelta, AdaMax, Nadam, RMSProp	0.001-0.01	32	30

Tables 13-19 illustrate the performance of thirteen pre-trained deep learning architectures under seven optimization techniques. In this work, to analyze the importance of our architectures in predicting COVID-19, we are mainly focused on the F-score, AUC, and accuracy since these are some crucial indicators for the analysis of medical image applications [51]. Here, red-colored underlined values indicate the topmost (first-best) value. Similarly, purple-colored underlined values show the second-best value and blue-colored underlined values signify the third-best value of our CNN models. The Adam optimizer performed well compared to the other techniques, especially on DenseNet-121, with 100% accuracy, AUC, and F-score. In this work, we examine the experimental results in two aspects:

- Based on optimization algorithms.
- Based on the deep learning architecture.

A. Based on optimization algorithms

From tables 13-19, we made the following observations based on the performance of optimization techniques:

- The Adam optimizer performed well compared to the other techniques, especially on DenseNet-121, with 100% accuracy, AUC, and F-Score since it slows down when converging to the local minima and minimizes the high variance.
- AdaMax optimization attained more than 99.16% accuracy on almost all the networks except DenseNet-201 and MobileNetV2, with a learning rate of 0.002 because it is less sensitive to noise in the gradients.
- Among all optimizers, Adadelta yields poor outcomes on the proposed pre-trained models except DenseNet-169 because the learning rate will become very low in the late training period. Similarly, Adagrad has yet to attain significant performance compared to all other optimizers except Adadelta since the learning rate will decrease due to many iterations.
- RMSProp obtained relatively better performance than Adadelta but is still low compared to other approaches since it may replicate the update process around the local minimum in the late training period.
- Overall the SGD optimizer achieved reasonably good accuracy compared to Adadelta and RMSProp however; it was lower than Adam, AdaMax, and Nadam because it may overshoot even after reaching global minima and holds a high variance.

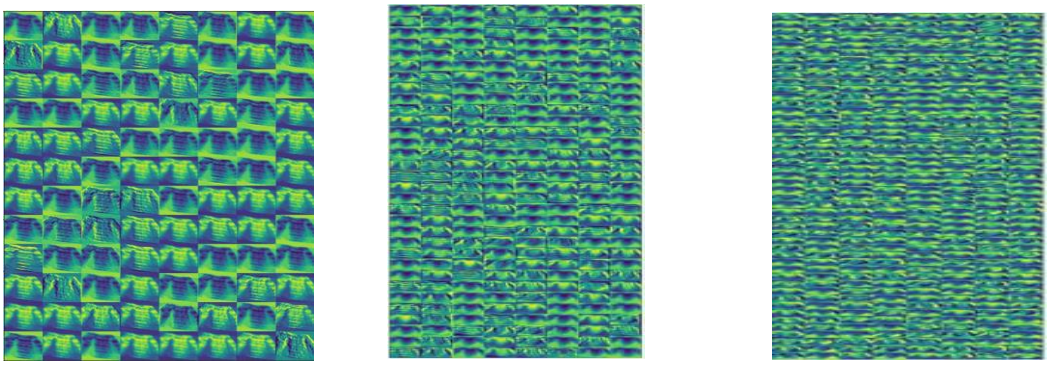
Based on the above analysis, we conclude that even though the Adam optimizer attained high accuracy (100%), AdaMax performs better since it consistently performs well on almost all architectures.

B. Based on the deep learning model

Tables 13-19, we identify the following remarks based on the performance of pre-trained CNN techniques:

- Among all the networks, the ResNet model consistently performed well on all optimizers (more than 98% accuracy) because for the following reasons:
  - It effectively minimizes the impact of the vanishing gradient problem during the training process.
  - Trained with a large number of layers without improving the percentage of training error.
- The DenseNet architecture achieved relatively good performance, especially DenseNet-121 on the Adam optimizer, which yields the highest accuracy with a value of 100% compared to other pre-trained CNN models due to the following features:
  - Features reuse capability.
  - DenseNet works effectively even with limited since it utilizes high-level features.
  - Enhancing feature propagation.

- The VGG performs reasonably well compared to AlexNet due to the increasing depth of the model and the usage of a small kernel. However, it offers low accuracy compared to ResNet, DenseNet, Inception, Xception, etc., because of the vanishing gradient problem.
- AlexNet yields low performance compared to the other pre-trained architectures, such as ResNet, VGG, and DenseNet, because the depth of the network is less, and it is not easy to learn relevant features from images.
- The Inception V3 and Inception-ResNet-V2 models obtained relatively better classification accuracy than VGG due to their computational cost and usage of smaller convolutions.
- The Xception model is a good improvement in accuracy compared to Inception due to its depthwise separable convolutions.



(a) Convolution Layer 1 (b) Convolution Layer 2 (c) Convolution Layer 3

FIGURE 6. Feature maps of the first three convolution layers of the AlexNet model

TABLE 13: PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE SGD OPTIMIZER

Network	Performance Measures (%)					
	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	99.08	99	99.54	99.31	99.04	99.06
ResNet-50	98.92	100	100	99.46	99.46(3)	99.26
ResNet-101	100	99.71	99.83	99.91(1)	99.85(1)	99.89(1)
ResNet-152	99.7	100	100	99.85(2)	99.85(1)	99.79(2)
DenseNet-121	99.53	99.05	99.53	99.53	99.29	99.37
DenseNet-169	99.36	100	100	99.68(3)	99.68(2)	99.58(3)
DenseNet-201	98.73	95.37	97.64	97.05	98.18	97.58
InceptionV3	99.37	99.37	99.68	99.52	99.37	99.37
Inception-ResNet-V2	99.06	98.4	99.22	99.14	98.73	98.5
VGG-16	98.5	99.15	99.5	98.99	98.82	98.74
VGG-19	97.72	98.18	97.88	97.8	97.8	97.17
MobileNetV2	99.05	99.38	99.68	99.36	99.21	99.16
Xception	99.84	98.43	99.22	99.53	99.13	99.37

TABLE 14 PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE ADAM OPTIMIZER

Network	Performance Measures					
	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	100	98.75	99.37	99.68	99.37	99.58
ResNet-50	100	95.65	97.83	98.9	97.82	98.53
ResNet-101	99.05	99.38	99.68	99.36	99.21	99.16
ResNet-152	99.54	100	100	99.77	99.77	99.68
DenseNet-121	100	100	100	100(1)	100(1)	100(1)
DenseNet-169	99.7	99.67	99.84	99.77	99.68	99.68
DenseNet-201	99.53	96.5	98.3	98.91	98.01	98.53
InceptionV3	99.84	99.7	99.84	99.84(3)	99.77	99.8(3)

Inception-ResNet-V2	100	99.7	99.84	99.92(2)	99.85(3)	99.9(2)
VGG-16	99.24	97.94	99.09	99.16	98.5	98.85
VGG-19	97.56	100	100	98.76	98.78	98.32
MobileNetV2	99.68	99.05	99.53	99.6	99.36	99.47
Xception	99.85	100	100	99.92(2)	99.92(2)	99.9(2)

TABLE 15 PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE ADAGRAD OPTIMIZER

Network	Performance Measures					
	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	98.17	100	100	99.07	99.08	98.74
ResNet-50	98.95	99.3	99.7	99.32	99.12	99.05
ResNet-101	99.7	100	100	99.85(1)	99.85(1)	99.8(1)
ResNet-152	99.52	97.86	98.9	99.21	98.69	98.95
DenseNet-121	99.84	98.75	99.37	99.6	99.3	99.48
DenseNet-169	99.68	99.37	99.68	99.68(2)	99.52(2)	99.58(2)
DenseNet-201	96.46	86.90	93.6	95.01	91.68	92.97
InceptionV3	97.99	97.7	98.91	98.45	97.84	97.9
Inception-ResNet-V2	98.7	99.11	99.51	99.1	98.9	98.74
VGG-16	99.54	98.37	99.23	99.38	98.95	99.16
VGG-19	99.38	99.02	99.53	99.45	99.2	99.26
MobileNetV2	99.21	100	100	99.60(3)	99.6(3)	99.47(3)
Xception	100	98.25	99.02	99.51	99.12	99.37

TABLE 16 PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE ADADELTA OPTIMIZER

Network	Performance Measures					
	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	94.63	99.67	99.84	97.16	97.15	96.22
ResNet-50	98.55	98.79	99.35	98.95	98.67(3)	98.64(3)
ResNet-101	98.56	99.08	99.52	99.04(2)	98.82(2)	98.74(2)
ResNet-152	97.53	98.68	99.37	98.44	98.1	97.9
DenseNet-121	99.08	97.32	98.78	98.93	98.2	98.53
DenseNet-169	99.37	99.68	99.84	99.6(1)	99.52(1)	99.47(1)
DenseNet-201	96.94	95.32	97.84	97.38	96.13	96.43
InceptionV3	99.38	97.08	98.62	98.99(3)	98.23	98.64(3)
Inception-ResNet-V2	96.09	91.72	95.94	96.01	93.90	94.65
VGG-16	98.31	96.37	98.31	98.31	97.34	97.7

VGG-19	98.6	95.47	97.85	98.22	97.03	97.6
MobileNetV2	97.42	93.87	97.27	97.34	95.64	96.33
Xception	98.42	86.94	93.84	96.07	92.68	94.65

TABLE 17: PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE ADAMAX OPTIMIZER

Network	Performance Measures					
	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	98.92	100	100	99.46	99.46	99.26
ResNet-50	99.68	99.68	99.84	99.76(3)	99.68	99.68(3)
ResNet-101	100	99.66	99.85	99.92(1)	99.83(1)	99.9(1)
ResNet-152	99.83	100	100	99.91(1)	99.91(1)	99.9(1)
DenseNet-121	100	99.68	99.84	99.92(1)	99.92(1)	99.9(1)
DenseNet-169	99.54	100	100	99.76(3)	99.77(3)	99.68(3)
DenseNet-201	99.68	95.37	97.66	98.66	97.52	98.22
InceptionV3	99.48	99.38	99.68	99.58	99.4	99.68(3)
Inception-ResNet-V2	99.84	99.7	99.84	99.84(2)	99.77(2)	99.8(2)
VGG-16	99.68	98.12	99.06	99.37	98.9	99.16
VGG-19	99.33	99.43	99.67	99.5	99.38	99.37
MobileNetV2	99.22	96.75	98.46	98.84	97.98	98.43
Xception	99.84	99.1	99.52	99.68	99.47	99.58

TABLE 18: PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE NADAM OPTIMIZER

Network	Performance Measures					
	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	99.37	97.47	98.75	99.06	98.42	98.74
ResNet-50	98.73	99.38	99.68	99.35	99.05	98.95
ResNet-101	99.51	99.7	99.84	99.67(3)	99.6	99.58
ResNet-152	100	98.32	99.24	99.62	99.16	99.47
DenseNet-121	99.84	100	100	99.92(1)	99.92(1)	99.9(1)
DenseNet-169	98.93	95.64	98.04	98.48	97.28	97.9
DenseNet-201	99.54	97.98	99.09	99.31	98.76	99.05
InceptionV3	100	99.33	99.7	99.85(2)	99.66(3)	99.8(2)
Inception-ResNet-V2	100	99.68	99.84	99.92(1)	99.84(2)	99.9(1)
VGG-16	99.68	99.07	99.52	99.6	99.37	99.47(3)
VGG-19	99.7	98.37	99.23	99.46	99.03	99.26
MobileNetV2	99.23	99.02	99.53	99.38	99.12	99.16
Xception	98.77	97.7	98.92	98.84	98.23	98.43

TABLE 19 PERFORMANCE ANALYSIS OF THE PROPOSED FRAMEWORK WITH THE RMSPROP OPTIMIZER

Network	Performance Measures

	TPR	TNR	PPV	F-Score	AUC	Accuracy
AlexNet	100	98	99.09	99.54	99	99.37
ResNet-50	99.07	100	100	99.53	99.53	99.37
ResNet-101	98.55	88.25	94.02	96.23	93.4	94.97
ResNet-152	99.53	99.68	99.84	99.68(3)	99.61(2)	99.58(3)
DenseNet-121	99.68	99.05	99.53	99.6	99.36	99.47
DenseNet-169	98.6	95.83	97.98	98.28	97.21	97.69
DenseNet-201	99.41	96.03	98.39	98.89	97.72	98.43
InceptionV3	99.67	93.8	97.35	98.5	96.73	97.90
Inception-ResNet-V2	99.85	99.32	99.7	99.77(2)	99.58(3)	99.68(2)
VGG-16	99.36	99.08	99.52	99.44	99.22	99.26
VGG-19	99.68	96.94	98.42	99.04	98.31	98.74
MobileNetV2	98.9	99.06	99.52	99.21	98.98	98.95
Xception	100	99.4	99.68	99.84(1)	99.7(1)	99.8(1)

C. Comparison with Existing Models

The classification performance of the implemented framework was compared with the state-of-the-art approaches, and their outcomes are represented in Table 20. From this, we observed that the adopted DenseNet-121 model with transfer learning achieved an accuracy of 100% on the Adam optimizer with a learning rate of 0.002. Hence, the presented deep transfer learning architecture can be used as a predictive tool in clinical analysis to assist doctors in identifying COVID-19 from LUS imagery data.

TABLE 20 COMPARATIVE ANALYSIS OF THE PROPOSED AND EXISTING ARCHITECTURES

Reference	Method	Optimizer	Learning Rate	Accuracy
Michael et al. [14]	VGG-16	-	0.0001-0.00001	98.54%
Julia et al. [17]	Inception-V3	Adam	0.0001	91.5%
Hui et al. [18]	Multi-Scale RCNN	Adam	0.00001	95.11%
The Proposed	**DenseNet-121**	**Adam**	**0.002**	**100%**

V. Conclusion and Future Scope

Researchers have developed a deep learning-based COVID-19 automatic screening tool using LUS data instead of CT and X-ray images. The DenseNet-121 model achieved 100% accuracy, F-Score, and AUC in identifying COVID-19 samples from non-COVID samples. However, the project has limitations due to a limited LUS database. Future work aims to incorporate the model's results into a larger database and extend it to multiclass classification for bacterial pneumonia and non-COVID samples.

References

1. Buonsenso, Danilo, Davide Pata, and Antonio Chiaretti. "COVID-19 outbreak: less stethoscope, more ultrasound." The Lancet Respiratory Medicine 8.5 (2020): e27.
2. Memish, Ziad A., et al. "Middle East respiratory syndrome." The Lancet 395.10229 (2020): 1063-1077.
3. https://www.worldometers.info/coronavirus/
4. https://www.cdc.gov/coronavirus/2019-ncov/variants/variant-classifications.html
5. https://www.who.int/health-topics/coronavirus#tab=tab_3
6. https://www.mayoclinic.org/tests-procedures/covid-19-diagnostic-test/about/pac-20488900
7. https://www.upmc.com/coronavirus/covid-19
8. https://www.health.harvard.edu/diseases-and-conditions/preventing-the-spread-of-the-coronavirus
9. https://community.nasscom.in/communities/covid-19/effect-of-covid-19-on-hospitality-industry.html
10. Susairaj, A. Xavier, A. Salaijayamani, and A. Premkumar. "Impact of Covid-19 on Automobile Industries in India."
11. Mamillapalli, Raja Sekhar. (2021). IMPACT OF COVID -19 ON REAL ESTATE SECTOR IN INDIA ABSTRACT. Journal of Global Information Management.
12. Zhao, Lingyi, and Muyinatu A. Lediju Bell. "A review of deep learning applications in lung ultrasound imaging of COVID-19 patients." BME Frontiers 2022 (2022).
13. Bourcier, Jean-Eudes, Sergiu Braga, and Didier Garnier. "Lung ultrasound will soon replace chest radiography in the diagnosis of acute community-acquired pneumonia." Current infectious disease reports 18.12 (2016): 1-6.
14. Horry, Michael J., et al. "COVID-19 detection through transfer learning using multimodal imaging data." Ieee Access 8 (2020): 149808-149824.
15. Born, Jannis, et al. "POCOVID-Net: automatic detection of COVID-19 from a new lung ultrasound imaging dataset (POCUS)." arXiv preprint arXiv:2004.12084 (2020).
16. Zhang, Jiaqi, et al. "Detection and classification of pneumonia from lung ultrasound images." 2020 5th International Conference on Communication, Image and Signal Processing (CCISP). IEEE, 2020.
17. Diaz-Escobar, Julia, et al. "Deep-learning based detection of COVID-19 using lung ultrasound imagery." Plos one 16.8 (2021): e0255886.
18. Che, Hui, et al. "Multi-feature multi-scale CNN-derived COVID-19 classification from lung ultrasound data." 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, 2021.
19. Awasthi, Navchetan, et al. "Mini-COVIDNet: efficient lightweight deep neural network for ultrasound based point-of-care detection of COVID-19." IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 68.6 (2021): 2023-2037.
20. La Salvia, Marco, et al. "Deep learning and lung ultrasound for Covid-19 pneumonia detection and severity classification." Computers in biology and medicine 136 (2021): 104742.
21. Born, Jannis, et al. "Accelerating detection of lung pathologies with explainable ultrasound image analysis." Applied Sciences 11.2 (2021): 672.
22. Muhammad, Ghulam, and M. Shamim Hossain. "COVID-19 and non-COVID-19 classification using multi-layers fusion from lung ultrasound images." Information Fusion 72 (2021): 80-88.
23. Barros, Bruno, et al. "Pulmonary COVID-19: Learning spatiotemporal features combining cnn and lstm networks for lung ultrasound video classification." Sensors 21.16 (2021): 5486.
24. Santurkar, Shibani, et al. "How does batch normalization help optimization?." Advances in neural information processing systems 31 (2018).
25. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.
26. Montavon, Gregoire, Wojciech Samek, and Klaus-Robert Muller. "Methods for interpreting and understanding deep neural networks." Digital Signal Processing 73 (2018): 1-15
27. NarasingaRao, M. R., et al. "A survey on prevention of overfitting in convolution neural networks using machine learning techniques." International Journal of Engineering and Technology (UAE) 7.2.32 (2018): 177-180.
28. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.
29. Khan, Asifullah, et al. "A survey of the recent architectures of deep convolutional neural networks." Artificial intelligence review 53.8 (2020): 5455-5516.
30. Kirzhevsky, A., Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.
31. Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115.3 (2015): 211-252.
32. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
33. Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.
34. Philipp, George, Dawn Song, and Jaime G. Carbonell. "Gradients explode-deep networks are shallow-resnet explained." (2018).
35. Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

36. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
37. Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
38. Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
39. Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
40. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
41. Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." Thirty-first AAAI conference on artificial intelligence. 2017.
42. https://www.indusmic.com/post/transfer-learning-and-fine-tuning-of-neural-networks
43. Sun, Shiliang, et al. "A survey of optimization methods from a machine learning perspective." IEEE transactions on cybernetics 50.8 (2019): 3668-3681.
44. Robbins, Herbert, and Sutton Monro. "A stochastic approximation method." The annals of mathematical statistics (1951): 400-407.
45. Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12.7 (2011).
46. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
47. Igel, Christian, and Michael Hüsken. "Improving the Rprop learning algorithm." Proceedings of the second international ICSC symposium on neural computation (NC 2000). Vol. 2000. 2000.
48. Zeiler, Matthew D. "Adadelta: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).
49. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
50. Dozat, Timothy. "Incorporating nesterov momentum into adam." (2016).
51. Li, Chun, et al. "Transfer learning for establishment of recognition of COVID-19 on CT imaging using small-sized training datasets." Knowledge-Based Systems 218 (2021): 106849.