Explainable Artificial Intelligence-based Edge Fuzzy Images for COVID-19 Identification and Detection

Qinhua Hu[a], Francisco Nauber B. Gois[b], Rafael Costa[c], Lijuan Zhang[d], Ling Yin[e], Naercio Magaia[f], Victor Hugo C. de Albuquerque[g]

[a]*School of Chemical Engineering and Energy Technology, Dongguan University of Technology, Dongguan 523808, China. Email: huqh@dgut.edu.cn*
[b]*Public Health School of Ceará .Av. Antônio Justa, 3161 - Meireles, , 60165-090,Fortaleza/CE, Brazil. Email: naubergois@gmail.com*
[c]*Health Department of Ceara Email: rafael.costa@alu.ufc.br*
[d]*DGUT-CNAM Institute, Dongguan University of Technology, Dongguan 523106, China. Email: zhanglijuan@dgut.edu.cn*
[e]*School of Mechanical Engineering, Dongguan University of Technology, Dongguan 523808, China. Email: yinl@dgut.edu.cn*
[f]*School of Engineering and Informatics, University of SUssex, United Kingdom. Email: N.Magaia@sussex.ac.uk*
[g]*Department of Teleinformatics Engineering, Federal University of Ceará, Fortaleza, Fortaleza/CE, Brazil*

Abstract

The COVID-19 pandemic continues to wreak havoc on the world's population's health and well-being. Successful screening of infected patients is a critical step in the fight against it, with radiology examination using chest radiography being one of the most important screening methods. For the definitive diagnosis of COVID-19 disease, the reverse-transcriptase polymerase method was given emergency approval and is now one of the most widely utilized methods for detecting covid-19 in the pandemic. By the end of the year, the CDC will have issued a withdrawal and will prescribe new ways for distinguishing influenza and covid-19.PCR is unable to produce fine-grained results, resulting in a substantial number of false positives. We propose a Multi-Input Transfer Learning COVID-Net fuzzy convolutional neural network to detect COVID-19 instances from torso X-ray, motivated by the latter and the open-source efforts in this research area. Furthermore, we use an explainability method to investigate several Convolutional Networks COVID-Net forecasts in an effort to not only gain deeper insights into critical factors associated with COVID-19 instances, but also to aid clinicians in improving screening. We show that using transfer learning and pre-trained models, we can detect it with a high degree of accuracy. Using X-ray images, we chose four neural networks to predict its probability. Finally, in order to achieve better results, we considered various methods to verify the techniques proposed here. As a result, we were able to create a model with an area under curve of 1.0 and accuracy, precision, and recall of 0.97. The model was quantized for use in Internet of Things devices and maintained a 0.95 percent accuracy.

Keywords: Soft Computing, X-Rays, COVID-19, Multi-Input Convolutional Network, Internet of Things, XAI

1. Introduction

The Coronavirus (COVID-19) is a viral disease caused by hard acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The outbreak seems to have a detrimental impact on the market and health. Many nations are challenged by the medical tools necessary for COVID-19 detection. They are looking forward to developing a low-cost, fast tool to detect and diagnose the virus efficiently. Even though a chest X-Ray (CXR) scan is a useful candidate, the images created by the scans must be analyzed, and large numbers of evaluations need to be processed. A CXR of individuals is a vital step in the struggle against COVID-19. This disease causes pulmonary opacities and bilateral parenchymal ground-glass, sometimes with a peripheral lung distribution and a morphology. Several Deep Learning (DL) techniques revealed a firmly optimistic accuracy of COVID-19 patient discovery via the use of CXRs [44] [1]. Because most hospitals have X-ray machines, it is the radiologists' first choice. Automatic diagnosis of COVID-19 from chest pictures is particularly desirable because radiologists are limited and also busy in pandemic conditions. Despite the fact that most machine learning models include a margin of error, automation can be critical for screening patients who can then be assessed with more precise tests.

Image segmentation is an essential procedure for most medical image analysis tasks. Having great segmentations will help clinicians and patients provide essential information for 2-D and 3-D visualization, surgical preparation, and early disease detection [28]. Segmentation describes regions of interest (ROIs), e.g., lung, lobes, bronchopulmonary segments, and infected areas or lesions at the CXR or computed tomography (CT) images. Segmented regions could be further used to extract features for description and other applications

[40]. Automated computer-aided diagnostic (CADx) tools powered by artificial intelligence (AI) techniques to detect and distinguish COVID-19 related nasal abnormalities must be tremendously valuable, given the significant number of patients. These tools are particularly vital in places with inadequate CT accessibility or radiological experience, and CXRs create fast, higher throughput triage in mass casualty situations. These instruments combine radiological picture processing components with computer vision to identify common disease indications and localize problematic ROIs. At the moment, recent advances in machine learning (ML), especially DL approaches using convolutional neural networks (CNNs), have demonstrated promising performance in identifying, classifying, and measuring disease patterns in medical images in CT scans and CXRs [37] [7] [29] [34] [11] [9] [10] [27] [38].

In the past decades, Fuzzy logic has represented a vital role in many research areas [9]. Fuzzy logic is an offshoot of fuzzy set theory, which reproduces reasoning and human thinking to boost the procedure's efficacy when managing uncertain or vague data [38].

With little loss in model accuracy, post-training quantization is a conversion technique that can reduce model size while improving CPU and hardware accelerator latency. You can quantize a TensorFlow floating model that has already been trained by converting it to TensorFlow Lite format with the TensorFlow Lite Converter.

As a result, the goal of this research is to use ML to solve the problem of identifying COVID-19, using X-rays. VGG16, ResNet152V2, InceptionV3, and EfficientNetB3 were chosen as the neural networks to predict disease probability. Finally, in order to obtain better results, we use several techniques proposed here, such as fuzzy filters and Multinput networks. According to a fuzzy equal relation, fuzzy rough set-based approaches find reduct directly on initial data. The difference between items is preserved by a fuzzy relation. The classification precision can be improved using a fuzzy rough set approach. As a result, we were able to produce models with an Area Under Curve (AUC) of 1.0 and several variations with very high-performance evaluation metrics like precision, accuracy, and recall. To quantize the model, we use TensorFlow Lite Converter with a 0.95 accuracy.

This study's main novelty uses a multi-input network with a combination of segmented and non-segmented images in a neural network composed of two pre-trained networks. In a nutshell, the primary contributions of this paper are:

- use a multi-input approach to CXR with COVID-19 classification;

- apply a trapezoidal membership function to generate fuzzy edge images of CXR with COVID-19;
- obtains classification models with AUC of 0.99 and recall of 100% to CXR COVID-19 detection.

2. Literature Review

In epidemic regions, COVID-19 presumed patients are in immediate demand for identification and suitable therapy. Nevertheless, medical images, mainly chest CT, include hundreds of pieces that require a very long time for those experts in diagnosing. Additionally, COVID-19, being a new virus, has comparable symptoms to several different kinds of pneumonia, which necessitates radiologists to collect many experiences for attaining a more accurate diagnostic operation. Therefore, AI-assisted diagnosis utilizing medical images is highly desirable [40]. Several studies aim to separate COVID-19 patients from non-COVID-19 subjects. The researchers distinguished pneumonia manifestations with higher specificity from that of viral pneumonia on chest CT scans. It was noted that COVID-19 pneumonia was shown to be peripherally distributed together with ground glass opacities (GGO) and vascular thickening [33]. Abdel-Basset et al.[44] propose a hybrid COVID-19 detection model based on an improved marine hunters algorithm (IMPA) to get X-Ray image segmentation. The ranking-based diversity decrease (RDR) strategy enhances the IMPA operation to achieve better alternatives from fewer iterations. The experimental results reveal that the hybrid model outperforms all other algorithms for a range of metrics. Abdul Waheed et al.[44] present a process to generate synthetic chest X-ray (CXR) images by developing an Auxiliary Classifier Generative Adversarial Network (ACGAN) [33]. The segmentation approaches in COVID-19 programs can be mostly grouped into two different classes, i.e., the lung-region-oriented approaches as well as the lung-lesion-oriented procedures. The former lung-region-oriented procedures aim to separate lung areas, i.e., entire lung and lung lobes, from other areas in CT or X-ray, which is considered as a requisite measure in COVID-19 [40]. Jin et al. [15] shows a two-stage pipeline for screening COVID-19 in CT images, where the entire lung area is first detected through an efficient segmentation network based on UNet+. Wang proposes a novel COVID-19 Pneumonia Lesion segmentation system (COPLE-Net) to better deal with the lesions with various scales and looks [45]. Chouhan et al. [7] approach extract features from images using several pre-trained neural network models. The study uses five distinct models, examined their performance, and combined outputs, which beat individual models, reaching the state-of-the-art performance in pneumonia identification. The study reached an accuracy of 96.4% with a recall of 99.62% on unseen data from the Guangzhou Women and Children's Medical Center dataset. Zheng et al. [49] developed a weakly-supervised deep learning-based software utilizing 3D CT volumes to identify COVID-19. The lung region was segmented using a pre-trained UNet; then, the segmented 3D lung region was fed into a 3D deep neural network to foretell the probability of COVID-19 infectious. The present study presents a different approach from the studies presented, using a multi-input architecture and segmented images in conjunction with non-segmented images. To extract information from fabric photos, Lin et al. propose a multi-input neural network. The segmented small-scale image and the related features collected using standard methods are the inputs. Experiments suggest that including these manually extracted features into a neural network can increase its performance to a degree [22].For identifying autism, Epalle et al. propose a multi-input deep neural network model. The architecture of the model is built to accommodate neuroimaging data that has been preprocessed using three different reference atlases. For each training example, the proposed deep neural network receives data from three alternative parcellation algorithms at the same time and learns discriminative features from the three input sets automatically. As a result of this process, learned features become more general and less reliant on a single brain parcellation approach. The study used a collection of 1,038 real participants and an augmented set of 10,038 samples to validate the model utilizing cross-validation methods. On genuine data, the study achieve a classification accuracy of 78.07 percent, and on augmented data, the model reach a classification accuracy of 79.13 percent, which is about 9% higher than previously reported results [12].

3. Background

3.1. Blind/Referenceless Image Spatial Quality Evaluator

Blind/referenceless image spatial quality evaluator (BRISQUE) [8] [35] is a reference-less quality assessment technique. The BRISQUE algorithm estimates the quality score of an image with computational performance. The algorithm selects the pointwise numbers of sectionally normalized luminance signs and measures image naturalness based on measured differences from a natural picture form. The algorithm models the incidence of pairwise statistics of neighboring normalized luminance signals, which provide deformity orientation information. Though multiscale, the version applies to calculate features making it computationally fast and time-efficient [8] [35].

The BRISQUE model utilized a spatial method. First, a locally normalized luminance, also known as Mean Subtracted Contrast Normalized (MSCN) [8], is calculated as the following equation:

$$I(m,n) = \frac{I(m,n) - \mu(m,n)}{\sigma(m,n) + c} \tag{1}$$

where $\mu(m,n)$ is the local mean,I(m,n) is the intensity image, and normalizes using local variance σ(m,n). N are spatial indices, M and N are the image height and width, respectively, to avoid a zero denominator (variance). The local mean μ(m,n) and local variance δ(m,n) is calculated using the following equations:

$$\mu(m,n) = \sum_{k}^{k} = -k \sum_{l}^{L} =_{-L} \ w_{k,l} I_{k,l}(m,n) \tag{2}$$

$$\sigma(m,n) = \sqrt{\sum\nolimits_{K}^{K} =_{-K} \ \sum_{l}^{L} =_{-L} \ w_{k,l}(I_{k,l}(m,n) - \mu(m,n))^2} \tag{3}$$

where $w = \{w_{k,l} | k = -K,..,K, l = -L,..,L\}$

3.2. K-means segmentation method

K-means clustering is a common segmentation technique in pixel-based methods. Clustering pixel-based approaches have low complexity in comparison to other region-based approaches. K-means clustering is adequate for image segmentation because the amount of clusters is usually known for images of particular areas of the body. K-means is a clustering algorithm to partition data. Clustering is the procedure for grouping data points with similar feature vectors into several clusters. Let the feature vectors obtained from l clustered data be $X = \{x_i | i = 1, 2, ..., /\}$. The generalized algorithm starts k cluster centroids $C = \{c_j | j = 1, 2..., k\}$
by randomly choosing k characteristic vectors from X. Next, the feature vectors are grouped into k clusters using a chosen distance measure, such as Euclidean distance [46].

3.3. Fuzzy edge images using trapezoidal membership functions

Edge detection is the strategy used most often for segmenting images based on fluctuations in intensity. Edge detection is a requirement for image segmentation because it usually allows the image to be represented by black and white colors. Edge detection identifies the size and shape of an item. A better edge detection method is very likely to be a valuable tool for several applications. A digital image is a discrete description of reality. The image is composed of the color of the pixels and the position of these objects. Any potential treatment of the picture will have to account for the image's discretization issues. For instance, at times, it is not possible to discern which pixel belongs to which item. Even a human has some difficulty with the place of the edges on an image. Conventional segmentation techniques such as watershed, region growing, and thresholding are suitable for segmenting regions with clear boundaries. However, for cases with boundaries and inhomogeneity, these methods cannot help segment the areas. Therefore, the fuzzy logic appears as a suitable choice for tackling these edges' representation [24] [21] [14] [25] [2].

Fuzzy systems are an option to the classic boolean logic that only has two states: false or true. The membership values have been signaled by either 0 absolute false or 1 for complete accuracy and range. Fuzzy systems overcome the uncertainties in the information and solve image processing [39]. In a fuzzy inference system (FIS)[21], a fuzzy set declares each fuzzy number and requires a predetermined range of crisp with a grade of membership. The fuzzy sets of input membership functions transfer crisp inputs into fuzzy inputs. The set is explained as follows $X = x_1, x_2, ..., x_n$ where, x is an element μ in the set X. A membership worth expresses the grade of membership linked to each element x_i in a fuzzy set A, which reveals a combination $A = \mu_1(x_1), \mu_2(x_2), ..., \mu_n(x_n)$.

3.3.1. Fuzzy trapezoidal membership function

Membership function (MF) is a curve that defines how every pixel from the input is redirected to a membership value between 1 and 0. The MF curve is a function of a vector x and is determined by four scalar parameters b, a, c, and d [17] [18].

3.4. Transfer Learning

The use of DL and CNN methodologies in various computer vision software has been grown quickly. DL draws its power to optimize multiple neuron layers connected as a system that includes operators and linear.

A convolutional neural network is a type of feed-forward neural network broadly employed for picture-based classification, object detection, and recognition. The fundamental principle is using convolution, which generates the filtered characteristic maps piled over each other [26].

A CNN is a structure of DL that measures the convolution between the weights along with a picture input. It selects attributes from the input data as opposed to conventional ML methods. During the learning process, the optimal values to the convolution coefficients, using a pre-defined price function, are discovered, based on which the characteristics are automatically determined. Convolution is a method that takes a little matrix of numbers (known as kernel or filter), pass it on an image and change it based on the values from the filter. Subsequent attribute map values are calculated according to the following formula [48]:

$$G[m,n] = (f * h)[m,n] = \sum_{i} \sum_{k} h[j,k] f[m-j, n-k] \tag{4}$$

The convolutional layer gives a convolved characteristic map; as a result signal after applying the dot product between a small region of input and the filter weights to which they all are connected. Then, the pooling layer performs a downsampling operation. In the convolutional neural system, the size of pooling layer output can be measured using the following formula [26]:

$$\frac{W - F + 2P}{S} - 1 \tag{5}$$

where W is the input size; F is the convolutional kernel size; P is the padding value and S is the step size.

Transfer learning has brought considerable importance since it can work with little or no information in the training phase. That is, data that is well-established is adjusted by move learning from one domain to another. Transfer learning is well-suited to scenarios where a version performs poorly due to obsolete data or scant[23] [50]. This form of transfer learning used in DL is known as an inductive transfer. This is where the reach of feasible models, i.e., model bias, is narrowed in a practical way using a model match on a different but related task.

Since AlexNet won the ImageNet competition, CNNs are utilized for a broad selection of DL applications. From 2012 to the current, researchers are attempting to apply CNN on several different tasks [31].

3.4.1. VGG16

The VGG16 system is fashioned of 3 x 3 convolutional layers, 13 convolutional layers, and three fully-connected layers and can be attached to the pooling layer after every phase. The max-pooling layer follows some convolutional layers. The stride is set to 1 pixel. The five max-pooling layers use a determined stride of 2 and a 2 x 2-pixel filter. A padding of 1 pixel is done for the 3 x 3 convolutional layers—all the layers of the network use ReLU as the activation function [32].

3.4.2. ResNet

Deep Residual Network (ResNet) is an Artificial Neural Network (ANN) that overcomes reduced precision when developing a plain ANN using a deeper layer compared to a shallower ANN. This Deep Residual Network's purpose would be to earn ANN with layers with higher precision. The idea of it would be to create ANN that may upgrade the weight into a shallower layer, i.e., decrease degradation gradient [4]. Residual Networks (ResNet) improving DL introduces the notion of restructuring the layers in order for residual functions to be learned, which are relative to the inputs of these layers instead of learning capabilities that have no reference to the layer inputs. This restructuring solved the vanishing gradient problem in CNNs and allowed the training of considerably deeper neural networks. ResNet-152 includes 152 layers that are 8×VGGNet's depth, nevertheless has lower complexity. An ensemble of ResNet-152 models attained 3.57% precision error on the ImageNet test dataset and won first place in the ILSVRC 2015 classification challenge.

3.4.3. InceptionV3

Google's Inception V3 is the variant of the DL Architectures series Inception V3 trained with 1000 classes using the first ImageNet dataset with more than 1 million pictures [20].

3.4.4. EfficientNets

EfficientNet is a DL family of models with fewer parameters than the state-of-the-art versions. The model improves performance by a smart mixture of depth, width, and resolution. EfficientNet scales width and resolution using a compound coefficient. The advantage of EfficientNets in comparison to CNN is related to the decrease in the number of FLOPS along with which parameters, increasing precision. The classification accuracy for EffficientNet can also be better than those that have similar complexities [19] [31] [41].

3.5. Class Activation Map

The convolutional layer can be utilized as object sensors without giving an object's annotated bounding box to the practice sample. CNN lost this ability when connected to a fully connected layer. Unlike a traditional CNN, whereby looking at the image, the goal is to identify the picture class, the class activation map produces a heatmap to show the significant area of the image classified. Class activation mapping is a method to generate a specified category representing the discriminative areas that link the class of the object [6] [3].

3.6. Quantization

Deep learning has a long track record of success, but the use of heavy algorithms on large graphical processing units isn't ideal. In response to this disparity, a new class of deep learning methods known as quantization has emerged. Quantization is used to reduce the size of the neural network while maintaining high performance accuracy. This is particularly important for on-device applications, where memory and computation capacity are constrained. The process of approximating a neural network that uses floating-point numbers by a neural network with low bit width numbers is known as quantization for deep learning. The memory requirements and computational costs of using neural networks are drastically reduced as a result of this.

4. Proposed Method

The proposed method consists of using a multi-input network with two input images: a non-segmented image and the second segmented image using a fuzzy trapezoidal membership function or a K-means cluster segmentation. The proposal consists of four stages. The first one reads the set of images of the two datasets previously presented and performs the data shuffling. The second phase is to apply the fuzzy filter and the segmenter separately using K-means to compare which achieves the best accuracy. The fuzzy filter applies a trapezoidal fuzzy number presented as:

$$trapezoidal:\ f(x,a,b,c,d) = \begin{cases} 0, x \leqslant a & \\ x-a/b-a & a \leqslant x \leqslant d \\ 1, & b \leqslant x \leqslant c \\ d-x/d-c, & c \leqslant x \leqslant d \\ 0, & d \leqslant x \end{cases} \tag{6}$$

(7)

We use the Brisque score to select the best parameters (a,b,c,d) for the fuzzy filter. Initially, images treated with a diffuse filter and a cluster are used to train four networks using transfer of learning, namely VGG16, InceptionV3, ResNet152V2, and EfficientNetB3. This test aims to compare the performance between the applied diffuse filter and the cluster averages. We evaluate the tests using the AUC, accuracy, precision and recall metrics.

The third phase consists of applying tests with a multi-input network with two pre-trained networks. We altered the networks by substituting the last layer, using a fully connected layer, with 20 nodes divided into another fully connected sigmoid layer with a single node. The criteria are applied in 12 combinations between VGG16, InceptionV3, ResNet152V2, and EfficientNetB3. These tests were executed on ten epochs. The combination with a better AUC score was chosen to be tunning and evaluated. The fourth phase of the process consists of tunning the best model chosen. We train the best model in 100 epochs, and the metrics of the ROC curve, f1-score, and recall by epoch are presented. The last step consists of using class activation maps to compare explainable ML practice with and without the fuzzy filter.

4.0.1. Training and implementation details

We use Adaptive Moment Estimation (Adam) as the optimizer and Binary Cross Entropy as the loss function and Sigmoid as the networks' activation function. The initial learning rate was 0.001. We chose the simple learning speed schedule of decreasing the learning rate by a constant element when operation metric plateaus about the validation/test place (commonly called Reduce learning rate on plateau). We configure Reduce learning rate on plateau to monitor validation loss with factor parameter equals 0.2 and patience with value 2. The best-chosen combination was obtained after 100 epochs.

4.1. The Dataset

We use two datasets to train the proposed COVID-Net. The first dataset was available on `https://github.com/ieee8023/covid-chestxray-dataset` and approved by the University of Montreal's Ethics Committee (Fig. 4). The dataset is a collection of CXR of Healthy vs. Pneumonia (Corona) affected patients, infected patients, along with few other categories such as SARS (Severe Acute Respiratory Syndrome), Streptococcus and ARDS (Acute Respiratory Distress Syndrome). The second dataset is available on the Kaggle platform on `https://www.kaggle.com/nabeelsajid917/covid-19-x-ray-10000-images` and was used to test the model. Fig. 4.1 shows the diversity of patient cases from the dataset. The first dataset contains 5888 images, where 1.576 do not have any pneumonia, 4.265 have pneumonia of unknown cause, 58 cases of COVID-19, and 4 SARS images. We use a third dataset available on `https://www.kaggle.com/alifrahman/covid19-chest-xray-image-dataset` to improve the test results.

4.2. *Model Analysis*

We use the AUC, accuracy, precision, and recall to compare the models.

4.2.1. *The AUC metric*

The trapezoidal rule is used to ascertain the AUC. The resulting area is equal to this Mann-Whitney U statistic divided by $N_1 * N_2$, where N_1 and N_2 are the number of instances in C_1 and C_2, respectively. The AUC can be described as the chance to correctly identify the C_1 case when confronted with a randomly selected case from each class.

4.2.2. *Confusion Matrix*

Let I(x,y) : $\mathbb{R}^2 \rightarrow \mathbb{R}$ be a medical image and S(I(x,y)):$\mathbb{R}^2 \rightarrow \Omega, \Omega = 0, 1$ a binary decision of picture I(x,y). According to [30], the gold standard as G and the result as R, each fold can be classified as:

- true positive: $G(x, y) = 1 \wedge R(x, y) = 1$,
- false positive: $G(x, y) = 0 \wedge R(x, y) = 1$,
- true negative: $G(x, y) = 0 \wedge R(x, y) = 0$,
- false negative: $G(x, y) = 1 \wedge R(x, y) = 0$,

4.2.3. *Precision*

The precision is given by

$$P = \frac{TP}{TF + TP} \tag{8}$$

4.2.4. *Recall*

The recall is given by:

$$R = \frac{TP}{TP + FN} \tag{9}$$

where TP is the number of true positives and FN the number of false negatives, where R is Recall, the recall is the capability of the classifier to find all the samples. The best value is 1, and the worst value is 0.

4.2.5. Accuracy

Accuracy is a metric for deciding the models' performance in categorizing positive and negative classes. Assessing all detailed data with all data calculates the rating. It is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

4.3. Big O for convolutional networks

The number of features in each feature map in a CNN is at most a constant times the number of input pixels n (usually 1). Because each output is merely the sum product of k pixels in the picture and k weights in the filter, and k doesn't vary with n, convolving a fixed size filter across an image with n pixels requires O(n) time.

4.4. Pseudo-code and Source code

The process of building the single input model is described in the algorithm in the listing 1. Three parameters are required by the method: The weight of adjacent layers of neural networks (the model uses transfer learning, including additional layers to a pre-trained network); the Dropout value used for network generalization and the network to be created (the model uses transfer learning, including additional layers to a pre-trained network); and the weight of adjacent layers of neural networks (the model uses transfer learning, including additional layers to a pre-trained network). BatchNormalization is used in this method. Batch normalization is a transformation that keeps the mean output close to 0 and the standard deviation of the output close to 1. The algorithm in the listing 2 describes the process of creating the Multi-input model. The approach requires the same three parameters that were used to generate the single input model. The last parameter is an array containing a list of networks that will be used to build the model. Model(INPUTs=[model1.INPUT,model2.INPUT], outputs=x) concatenates the output of the two models, while concatenate([model1.output, model2.output]) concatenates the input of the two models. The pseudo-code for experiment execution is shown in the listing 3. There are two network lists, and the algorithm combines the networks and the layer weights. The model is trained with original and fuzzy images using the Adam optimizer and binary cross entropy loss function for each combination.

The Listing 4 show the main imports.The 5 and 6 listings describe methods for creating single and multi-input models, respectively. Listing 7 shows the code for training and evaluating several neural network configurations. The code for executing the predictions is listed in Listing 8. The procedure of applying the fuzzy transformation, which is provided through the python API skfuzzy, to each image is shown in Listing 9. The fuzzy method's parameters were chosen using the brisque score technique.

```
DEFINE FUNCTION create_single_model(weights,dropout_var,network):

    INPUT_shape=(256,256,3)

    SET model1 TO VGG16(include_top=False,INPUT_shape=(256, 256, 3))

    IF network=='VGG16':

        SET model1 TO VGG16(include_top=False,INPUT_shape=(256, 256, 3))

    IF network=='ResNet152V2':

        SET model1 TO ResNet152V2(include_top=False,
            INPUT_shape=(256, 256, 3))

    IF network=='InceptionV3':

        SET model1 TO InceptionV3(include_top=False,
            INPUT_shape=(256, 256, 3))

    IF network=='Xception':

        SET model1 TO Xception(include_top=False, INPUT_shape=(256, 256, 3))


    IF network=='EfficientNetB3':

        SET model1 TO EfficientNetB3(weights="imagenet",

            include_top=False, INPUT_shape=INPUT_shape)


    SET flat1_ TO Flatten() (model1.output)

    SET dropout_ TO Dropout(dropout_var)(flat1_)

    SET dense1_ TO Dense(weights, activation='relu')(dropout_)

    SET batch_ TO  BatchNormalization()(dense1_)

    SET dense2_ TO Dense(weights, activation='relu')(batch_)

    SET dropout_ TO Dropout(dropout_var)(dense2_)


    SET dense3_ TO Dense(4, activation="relu")(dropout_ )

    SET output TO Dense(1, activation="sigmoid")(dense3_)


    model= Model(INPUTs=model1.INPUT, outputs=output)


    RETURN model
```
Listing 1: Pseudo-code method to create Single Input Model

```
DEFINE FUNCTION create_model(weights,dropout_var,network):
    INPUT_shape=(256,256,3)
    SET model1 TO VGG16(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[0]=='VGG16':
        SET model1 TO VGG16(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[0]=='ResNet152V2':
        SET model1 TO ResNet152V2(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[0]=='InceptionV3':
        SET model1 TO InceptionV3(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[0]=='Xception':
        SET model1 TO Xception(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[0]=='EfficientNetB3':
        SET model1 TO EfficientNetB3(weights="imagenet",
            include_top=False, INPUT_shape=INPUT_shape)
    SET flat1 TO Flatten() (model1.output)
    SET dropout1 TO Dropout(dropout_var)(flat1)
    SET dense1 TO Dense(weights, activation='relu')(dropout1)
    SET batch1 TO  BatchNormalization()(dense1)
    SET dense12 TO Dense(weights, activation='relu')(batch1)
    SET dropout12 TO Dropout(dropout_var)(dense12)

    SET model1 TO Model(INPUTs=model1.INPUT, outputs=dropout12)

    SET model2 TO Net7(weights="imagenet",
        include_top=False, INPUT_shape=INPUT_shape)

    IF network[1]=='VGG16':
        SET model2 TO VGG16(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[1]=='ResNet152V2':
        SET model2 TO ResNet152V2(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[1]=='InceptionV3':
        SET model2 TO InceptionV3(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[1]=='Xception':
        SET model2 TO Xception(include_top=False, INPUT_shape=(256, 256, 3))
    IF network[1]=='EfficientNetB3':
        SET model2 TO EfficientNetB3(weights="imagenet",
            include_top=False, INPUT_shape=INPUT_shape)

    SET flat1_ TO Flatten() (model2.output)
    SET dropout_ TO Dropout(dropout_var)(flat1_)
    SET dense1_ TO Dense(weights, activation='relu')(dropout_)
    SET batch_ TO  BatchNormalization()(dense1_)
    SET dense2_ TO Dense(weights, activation='relu')(batch_)
    SET dropout_ TO Dropout(dropout_var)(dense2_)

    model2= Model(INPUTs=model2.INPUT, outputs=dropout_)

    SET modeld TO Sequential()
    modeld.add(Dense(14, INPUT_dim=4, activation="relu"))
    modeld.add(Dense(28,  activation="relu"))
    modeld.add(Dropout(dropout_var))
    modeld.add(Dense(4, activation="relu"))
    SET model2.trainable TO False
    SET combinedInput TO concatenate([model1.output,model2.output])
    SET x TO Dense(4, activation="relu")(combinedInput)
    SET x TO Dense(1, activation="sigmoid")(x)
    SET modelall TO Model(INPUTs=[model1.INPUT,model2.INPUT], outputs=x)
```

```
    RETURN modelall
```

Listing 2: Pseudo-code method to create Multi-Input Model

```
networks=['VGG16','ResNet152V2','InceptionV3','EfficientNetB3']
networks2=['VGG16','ResNet152V2','InceptionV3','EfficientNetB3']
best_auc=-10

FOR weights IN range(1,20,1):
    FOR network IN networks:
        FOR network2 IN networks2:
            IF network!=network2:
                OUTPUT('Testing with  {} and {} and {} '
                    .format(network,weights,network2))
                OUTPUT('Creating model')
                model=create_model(weights,0.5,[network,network2])

                SET opt TO Adam(lr=1e-3, decay=1e-3 / 100)

                OUTPUT('Model created')

                SET losses TO tf.keras.losses.BinaryCrossentropy()

                model.compile(loss="binary_crossentropy" ,
                    optimizer=opt,metrics=[
                    tf.keras.metrics.Precision(name='precision_2'),
                    tf.keras.metrics.Recall(name='recall_2')
                    ,tf.keras.metrics.AUC(name='auc'),'accuracy'])

                OUTPUT('Model compiled')

                history=model.fit(
                     x=[train_images,np.array(images_fuzzy)], y=y,
                     validation_data=([valid_images,
                        np.array(images_valid_fuzzy)],y_valid),
                     epochs=10, batch_size=10,callbacks=callbacks)

                auc=max(history.history['val_auc'])

                IF auc>best_auc:

                     best_auc=auc

                     best_network=network

                     best_weight=weights



                OUTPUT('val_auc',auc)

                IF (auc)>0.98:
                     OUTPUT('find')

                df_history=pd.DataFrame.from_dict(history.history)

                df_history.to_csv('History to {} and {} with weight of  {} '

                    .format(network,network2,weights)+'.csv')
```

Listing 3: Experiment execution pseudo-code

```
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications import ResNet152V2
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications import Xception
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import concatenate
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalMaxPooling2D
from keras.optimizers import Adam
from tensorflow.keras.applications import DenseNet121

import sys
import subprocess
import pkg_resources
required = {'efficientnet'}
installed = {pkg.key for pkg in pkg_resources.working_set}
missing = required - installed
if missing:
    !pip install -U efficientnet
import efficientnet.keras as efn

from efficientnet.keras import EfficientNetB0 as Net
from efficientnet.keras import EfficientNetB3
from efficientnet.keras import EfficientNetB3 as Net7
```
Listing 4: Imports

```
def create_single_model(weights,dropout_var,network):

    input_shape=(256,256,3)

    model1 = VGG16(include_top=False,input_shape=(256, 256, 3))

    if network[0]=='VGG16':
        model1 = VGG16(include_top=False,input_shape=(256, 256, 3))

    if network[0]=='ResNet152V2':
        model1 = ResNet152V2(include_top=False, input_shape=(256, 256, 3))


    if network[0]=='InceptionV3':
        model1 = InceptionV3(include_top=False, input_shape=(256, 256, 3))

    if network[0]=='Xception':
        model1 = Xception(include_top=False, input_shape=(256, 256, 3))



    if network[0]=='EfficientNetB3':
        model1 = EfficientNetB3(weights="imagenet",
            include_top=False, input_shape=input_shape)

    flat1_ = Flatten() (model1.output)
    dropout_ = Dropout(dropout_var)(flat1_)
    dense1_ = Dense(weights, activation='relu')(dropout_)
    batch_ =  BatchNormalization()(dense1_)
    dense2_ = Dense(weights, activation='relu')(batch_)
    dropout_ = Dropout(dropout_var)(dense2_)

    dense3_ = Dense(4, activation="relu")(dropout_ )
    output = Dense(1, activation="sigmoid")(dense3_)

    model= Model(inputs=model1.input, outputs=output)

    return model
```

Listing 5: Method to create Single Input Model

```
def create_model(weights,dropout_var,network):

    input_shape=(256,256,3)

    model1 = VGG16(include_top=False, input_shape=(256, 256, 3))
    if network[0]=='VGG16':
        model1 = VGG16(include_top=False, input_shape=(256, 256, 3))
    if network[0]=='ResNet152V2':
        model1 = ResNet152V2(include_top=False, input_shape=(256, 256, 3))
    if network[0]=='InceptionV3':
        model1 = InceptionV3(include_top=False, input_shape=(256, 256, 3))
    if network[0]=='Xception':
        model1 = Xception(include_top=False, input_shape=(256, 256, 3))
    if network[0]=='EfficientNetB3':
        model1 = EfficientNetB3(weights="imagenet",
            include_top=False, input_shape=input_shape)

    flat1 = Flatten() (model1.output)
    dropout1 = Dropout(dropout_var)(flat1)
    dense1 = Dense(weights, activation='relu')(dropout1)
    batch1 =  BatchNormalization()(dense1)
    dense12 = Dense(weights, activation='relu')(batch1)
    dropout12 = Dropout(dropout_var)(dense12)
    model1 = Model(inputs=model1.input, outputs=dropout12)
    model2 = Net7(weights="imagenet",
        include_top=False, input_shape=input_shape)
    if network[1]=='VGG16':
        model2 = VGG16(include_top=False, input_shape=(256, 256, 3))
    if network[1]=='ResNet152V2':
        model2 = ResNet152V2(include_top=False, input_shape=(256, 256, 3))
    if network[1]=='InceptionV3':
        model2 = InceptionV3(include_top=False, input_shape=(256, 256, 3))
    if network[1]=='Xception':
        model2 = Xception(include_top=False, input_shape=(256, 256, 3))
    if network[1]=='EfficientNetB3':
        model2 = EfficientNetB3(weights="imagenet",
            include_top=False, input_shape=input_shape)
    flat1_ = Flatten() (model2.output)
    dropout_ = Dropout(dropout_var)(flat1_)
    dense1_ = Dense(weights, activation='relu')(dropout_)
    batch_ =  BatchNormalization()(dense1_)
    dense2_ = Dense(weights, activation='relu')(batch_)
    dropout_ = Dropout(dropout_var)(dense2_)
    model2= Model(inputs=model2.input, outputs=dropout_)


    modeld = Sequential()
    modeld.add(Dense(14, input_dim=4, activation="relu"))
    modeld.add(Dense(28,  activation="relu"))
    modeld.add(Dropout(dropout_var))
    modeld.add(Dense(4, activation="relu"))

    model2.trainable = False
    combinedInput = concatenate([model1.output,model2.output])

    x = Dense(4, activation="relu")(combinedInput)
    x = Dense(1, activation="sigmoid")(x)
    modelall = Model(inputs=[model1.input,model2.input], outputs=x)

    return modelall
```

Listing 6: Method to create Multi-Input Model

```
networks=['VGG16','ResNet152V2','InceptionV3','EfficientNetB3']
networks2=['VGG16','ResNet152V2','InceptionV3','EfficientNetB3']
best_auc=-10

for weights in range(1,20,1):
    for network in networks:
        for network2 in networks2:
            if network!=network2:
                print('Testing with  {} and {} and {} '
                    .format(network,weights,network2))

                print('Creating model')
                model=create_model(weights,0.5,[network,network2])
                opt = Adam(lr=1e-3, decay=1e-3 / 100)
                print('Model created')
                losses = tf.keras.losses.BinaryCrossentropy()
                model.compile(loss="binary_crossentropy" ,
                    optimizer=opt,metrics=[
                    tf.keras.metrics.Precision(name='precision_2'),
                    tf.keras.metrics.Recall(name='recall_2')
                    ,tf.keras.metrics.AUC(name='auc'),'accuracy'])
                print('Model compiled')
                history=model.fit(
                     x=[train_images,np.array(images_fuzzy)], y=y,
                     validation_data=([valid_images,
                         np.array(images_valid_fuzzy)],y_valid),
                     epochs=10, batch_size=10,callbacks=callbacks)
                auc=max(history.history['val_auc'])
                if auc>best_auc:
                     best_auc=auc
                     best_network=network
                     best_weight=weights

                print('val_auc',auc)
                if (auc)>0.98:
                     print('find')

                df_history=pd.DataFrame.from_dict(history.history)
                df_history.to_csv('History to {} and {} with weight of  {} '
                    .format(network,network2,weights)+'.csv')
```

Listing 7: Run experiments with Multi-Input networks and Fuzzy data

```
#Make predictions
predictions=model.predict([valid_images,valid_images])

#Apply threshold on predictions
ypred=[1 if x>0.5 else 0 for x in predictions]

from sklearn.metrics import classification_report

print(classification_report(y_valid,ypred))
```

Listing 8: Make predictions and classification report

```
from skimage import transform,io
import skfuzzy as fuzz
import cv2

def fuzzy(image,a,b,c,d):

    mfx = fuzz.trapmf(image.flatten(),  [0.4, 0.6,200,200])
    return mfx.reshape(256,256,3)

images_fuzzy=[]
for image,name in zip(train_images,filenames):
    fuzzy_image=fuzzy(image,best_a,best_a+best_b,best_c,best_d)
    images_fuzzy.append(fuzzy_image)

images_valid_fuzzy=[]
for image in valid_images:
    fuzzy_image=fuzzy(image,best_a,best_a+best_b,best_c,best_d)
    images_valid_fuzzy.append(fuzzy_image)
```
Listing 9: Apply fuzzy transformation for each image

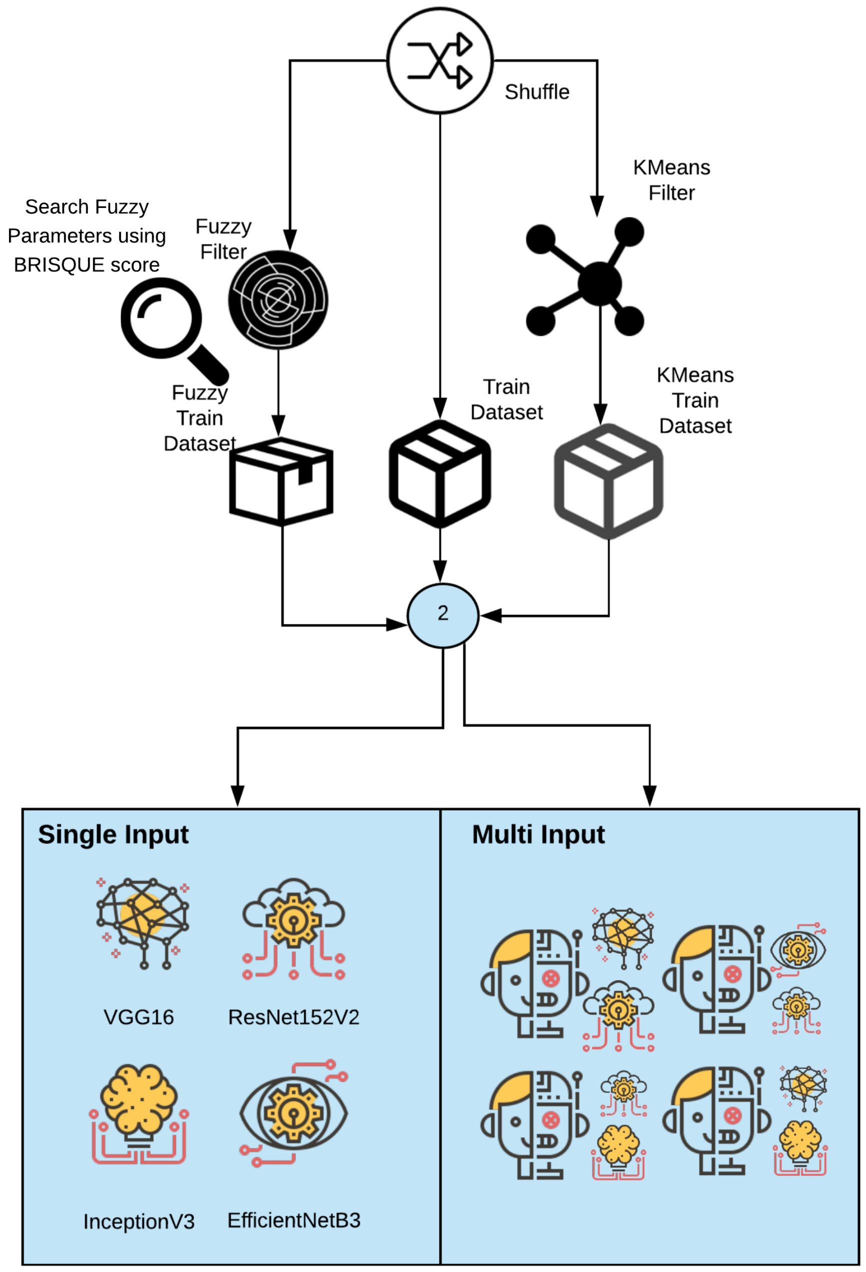


Figure 1: The second phase of proposed method

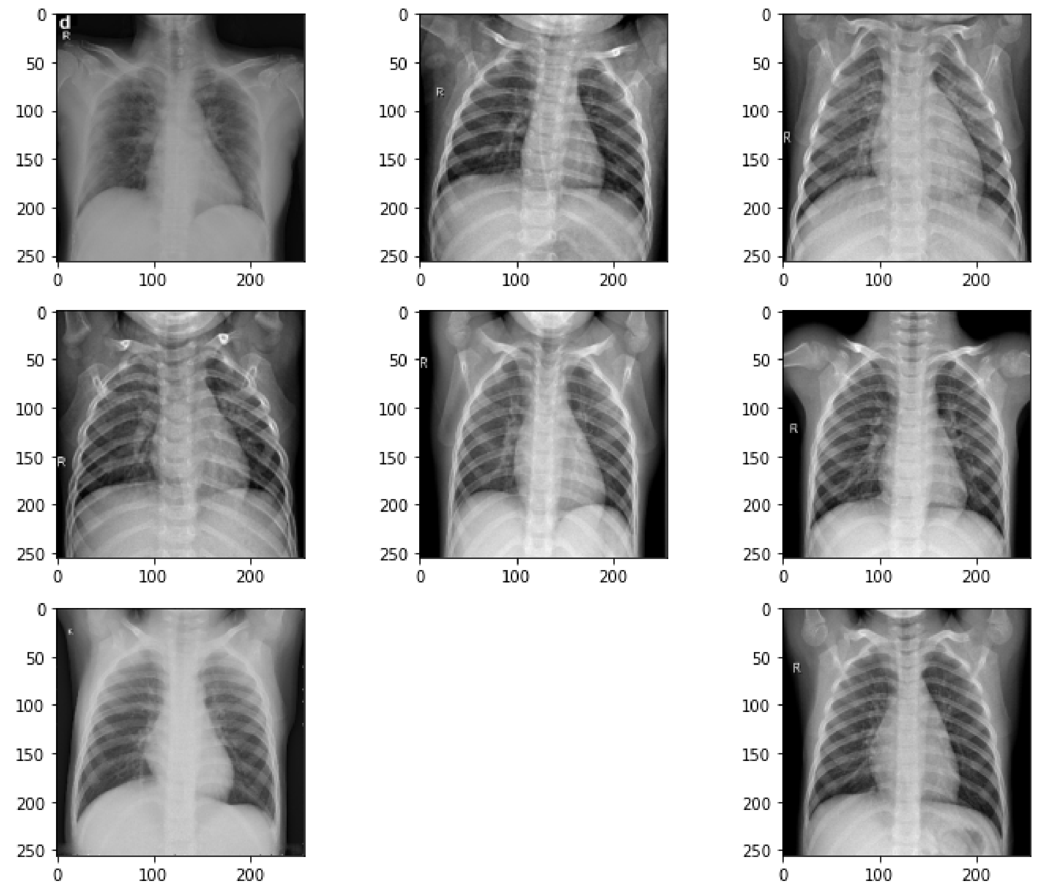
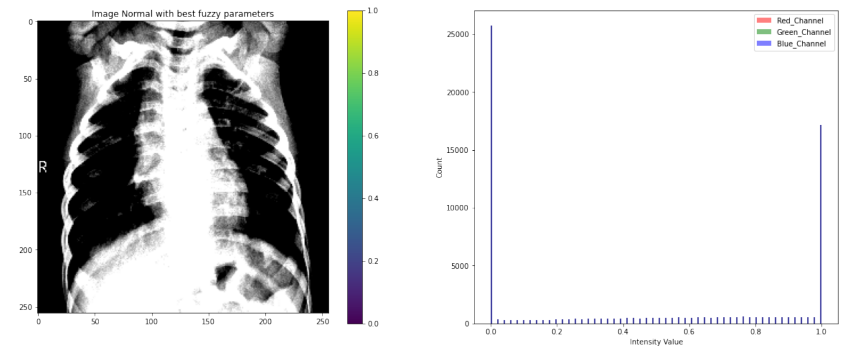
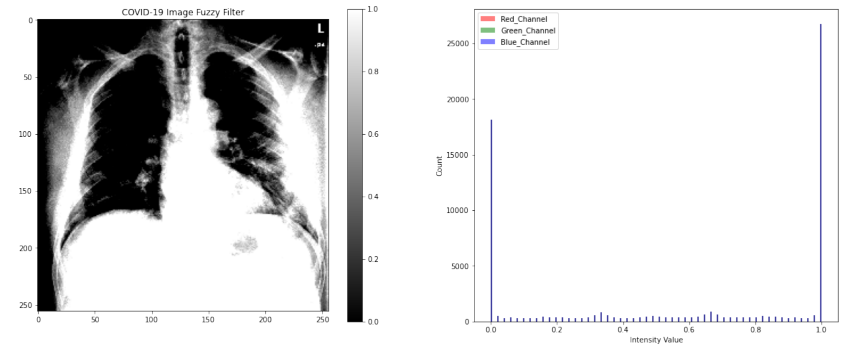


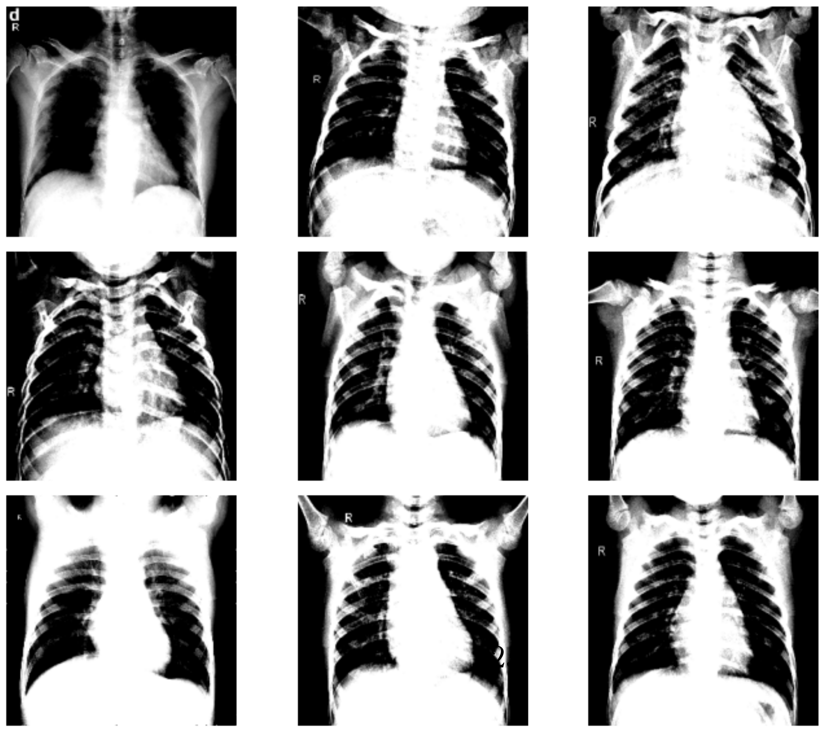
Figure 2: Image samples of the chosen datasets



(a) Histogram of non SARS sample X-Ray figure with the best fuzzy parameters



(b) Histogram of SARS sample X-Ray figure with the best fuzzy parameters



(c) Histogram of sample X-Rays figures with Brisque score

Figure 3: X-Ray samples with fuzzy filter

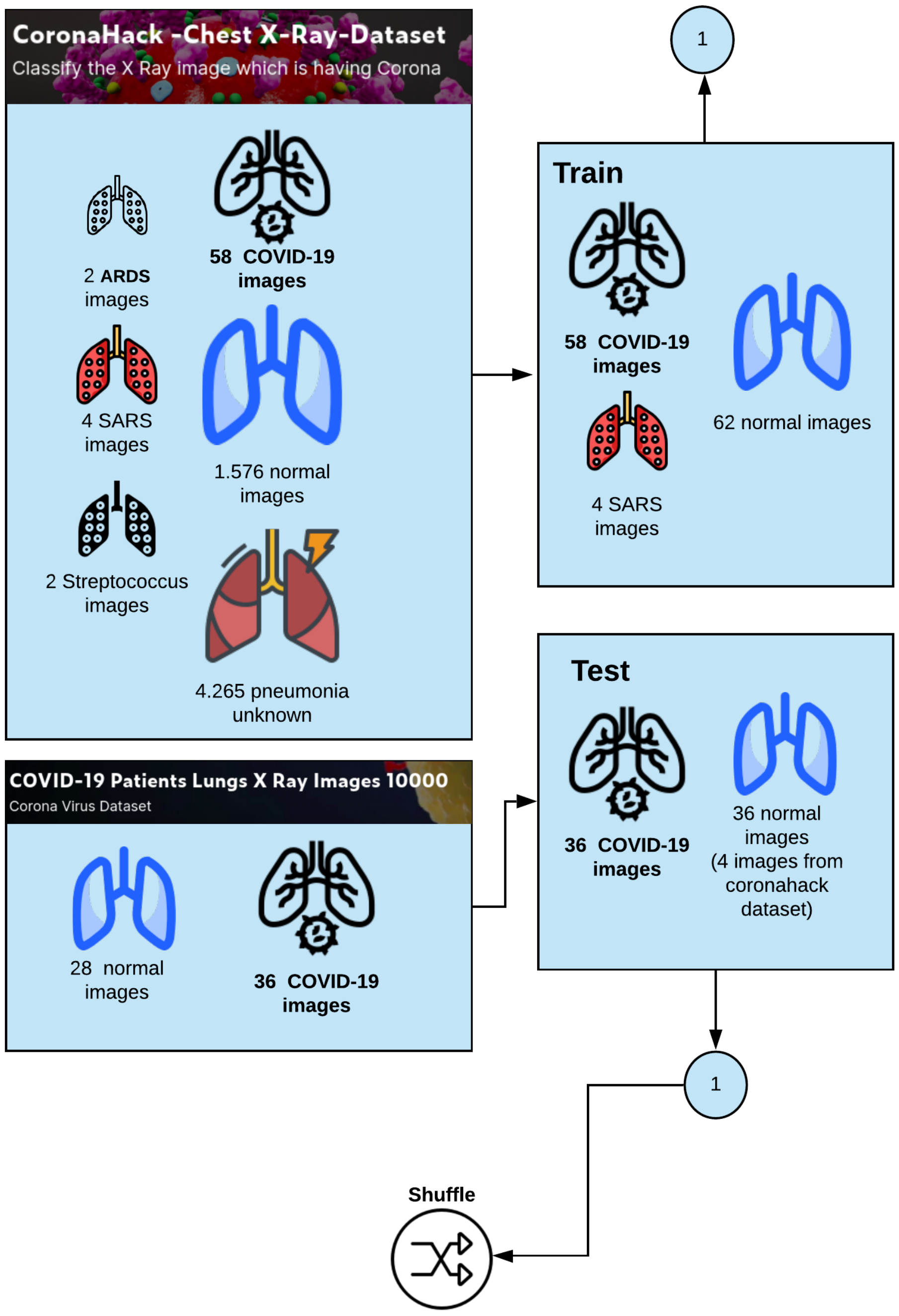


Figure 4: Description of the dataset used in this study

5. Results and Discussion

This study's primary goal is to present a monitoring model and reduce human errors in COVID-19 diagnosis. The proposed model's performance was measured using the Area Under the receiver operator characteristic curve (AUC). The code used in these experiments and datasets is available in `https://www.kaggle.com/naubergois/covid-xray-classification-with-fuzzy-1-0-recall` and `https://www.kaggle.com/naubergois/fork-of-covid-xray-classification-with-fuzzy-1-0-r`. The model requires O(n) time. We use Brisque values to tunning the Fuzzy filter parameters and obtain a=0.2, b=0.4, c=200, and d=200. Fig. 3(c) shows samples of the X-Rays with respective Brisque scores. Fig. 3(a) presents a non-SARS sample with the best fuzzy parameters. Fig. 3(b) presents a non-SARS sample with the best fuzzy parameters.

The first results concern the comparison between the use of the fuzzy filter and the segmentation with K-means. Table 1 show the results obtained with the use of K-means, each table show AUC, accuracy, precision and recall of each model. EfficientNetB3 obtain the better precision and InceptionV3 the better recall result. Tables 2 and 3 show the results for all pre-trained networks used with and without the fuzzy filter. We can see that the fuzzy filter has considerably improved the results of the ResNet152V2 model and for the AUC of all models. On the other hand, fuzzy filters reduce the recall of VGG16 and EfficientNet B3. The use of the fuzzy filter surpassed the use of the K-means cluster in all experiments. Fig 5(a) present the loss by an epoch of the best single input models. Fig 5(b) present the AUC by an epoch of the best single input models.

The tables 6 and 5 present the results obtained through the MultiInput technique using and not using the fuzzy filter. We obtained a higher AUC in all cases with the fuzzy filter. Therefore, it is easy to conclude that the fuzzy filter contributes to distinguishing positive or negative for COVID. It is also verified that in the same way that the EfficientNetB3 shows a decrease in accuracy using a single input, there is no difference in the multi-input technique.

The table 7 shows the comparison of the results between multi-input and single-input networks. Except for the EfficientNetB3 model, multi-input models obtained better value for AUC and accuracy.

The combination of the VGG16 and ResNet152V2 network achieved the best result. This combination was trained in 100 epochs. The VGG16 network received the images with a fuzzy filter and ResNet152v2 received the images

Table 1: Results of experiment single input classification with K-Means cluster

Model	AUC	Accuracy	Precision	Recall
VGG16	0.861497	0.763889	0.711111	0.888889
ResNet152V2	0.700231	0.527778	0.517241	0.833333
InceptionV3	0.870370	0.500000	0.500000	1.000000
EfficientNetB3	0.756944	0.680556	1.000000	0.361111

Table 2: Results of experiment single input classification with Fuzzy

Modell	AUC	Accuracy	Precision	Recall
VGG16	0.900463	0.597222	1.000000	0.194444
ResNet152V2	0.951389	0.777778	0.692308	1.000000
InceptionV3	0.912037	0.875000	0.829268	0.944444
EfficientNetB3	0.992670	0.736111	1.000000	0.472222

Table 3: Results of experiment single input classification without Fuzzy

Modell	AUC	Accuracy	Precision	Recall
VGG16	0.989197	0.972222	0.972222	0.972222
ResNet152V2	0.611111	0.611111	0.562500	1.000000
InceptionV3	0.852623	0.847222	0.765957	1.000000
EfficientNetB3	0.977238	0.916667	0.941176	0.888889

Table 4: Comparative results between using images with and without fuzzyfilter

Fuzzy	AUC	Accuracy
ResNet152V2	0.951389	0.777778
InceptionV3	0.912037	0.875000
EfficientNetB3	0.992670	0.736111
No Fuzzy	**AUC**	**Accuracy**
ResNet152V2	0.611111	0.611111
InceptionV3	0.852623	0.847222
EfficientNetB3	0.977238	0.916667

Table 5: Results of experiment MultiInput classification without Fuzzy

Model1	Model2	AUC	Acc	Prec.	Recall
VGG16	ResNet152V2	0.99	0.94	0.97	0.92
VGG16	InceptionV3	0.98	0.94	0.97	0.92
VGG16	EffNetB3	0.97	0.93	0.92	0.94
ResNet152V2	VGG16	0.99	0.94	0.97	0.92
ResNet152V2	InceptionV3	1.00	0.97	0.95	1.00
ResNet152V2	EffNetB3	0.99	0.88	0.80	1.00
InceptionV3	VGG16	0.98	0.94	0.97	0.92
InceptionV3	ResNet152V2	1.00	0.97	0.95	1.00
InceptionV3	EffNetB3	0.99	0.96	0.97	0.94
EffNetB3	VGG16	0.97	0.93	0.92	0.94
EffNetB3	ResNet152V2	0.99	0.88	0.80	1.00
EffNetB3	InceptionV3	0.99	0.96	0.97	0.94

Table 6: Results of experiment MultiInput classification with Fuzzy

Model1	Model2	AUC	Acc	Prec.	Recall
VGG16	ResNet152V2	1.00	0.94	1.00	0.89
VGG16	InceptionV3	1.00	0.97	0.97	0.97
VGG16	EfficientNetB3	0.88	0.72	0.64	1.00
ResNet152V2	VGG16	1.00	0.94	1.00	0.89
ResNet152V2	InceptionV3	0.99	0.96	0.92	1.00
ResNet152V2	EfficientNetB3	0.98	0.92	0.94	0.89
InceptionV3	VGG16	1.00	0.97	0.97	0.97
InceptionV3	ResNet152V2	0.99	0.96	0.92	1.00
InceptionV3	EfficientNetB3	0.96	0.94	0.97	0.92
EfficientNetB3	VGG16	0.88	0.72	0.64	1.00
EfficientNetB3	ResNet152V2	0.98	0.92	0.94	0.89
EfficientNetB3	InceptionV3	0.96	0.94	0.97	0.92

Table 7: Comparative of experiment results with multiInput and single input classification with fuzzy (mean values by network)

Multi input	**AUC**	**Acc**	**Precision**	**Recall**
VGG16	0.97	0.90	0.90	0.96
ResNet152V2	0.99	0.94	0.95	0.94
InceptionV3	0.99	0.96	0.96	0.97
EfficientNetB3	0.95	0.88	0.88	0.93
Single Input	**AUC)**	**Acc**	**Precision**	**Recall**
VGG16	0.90	0.60	1.00	0.19
ResNet152V2	0.95	0.78	0.69	1.00
InceptionV3	0.91	0.88	0.83	0.94
EfficientNetB3	0.99	0.74	1.00	0.47

without the filter. The table 8 show the f1_score achieved by the model. The model had a recall value of 1.

Fig. 6(a) shows the model's confusion matrix, and it is important to note that no COVID-19 was misclassified. Fig 7(a) and 7(b) present the class activation maps with and without fuzzy filter. We can observe that the regions of the map are well delimited with fuzzy filter images.

Several approaches to discover COVID-19 are trained to find out Pneumonia. Pneumonia is a possibly life-threatening illness caused by several pathogens. In common practice, most research proposes to classify the presence of Pneumonia associated with COVID-19. COVID-19 and pneumonia are both respiratory disorders that share many of the same symptoms. However, they are much more intimately connected. As a result of the viral infection that causes COVID-19 or the flu, some patients acquire pneumonia. Many times, pneumonia develops in both lungs in COVID-19 patients, putting the patient at serious danger of respiratory problems. Even if you don't have COVID-19 or the flu, you can get pneumonia from bacteria, fungus, and other microorganisms. However, COVID-19 pneumonia is a unique infection with unusual characteristics [13]. Some studies attempt to distinguish between common pneumonia and COVID-19 pneumonia [47][43][16]. Some datasets with X-ray pictures of cases (pneumonia or COVID-19) and controls have been made accessible in order to develop machine-learning-based algorithms to aid in illness diagnosis. These datasets, on the other hand, are primarily made up of different sources derived from pre-COVID-19 and COVID-19 datasets. Some studies have discovered significant bias in some of the publicly available datasets used to train and test

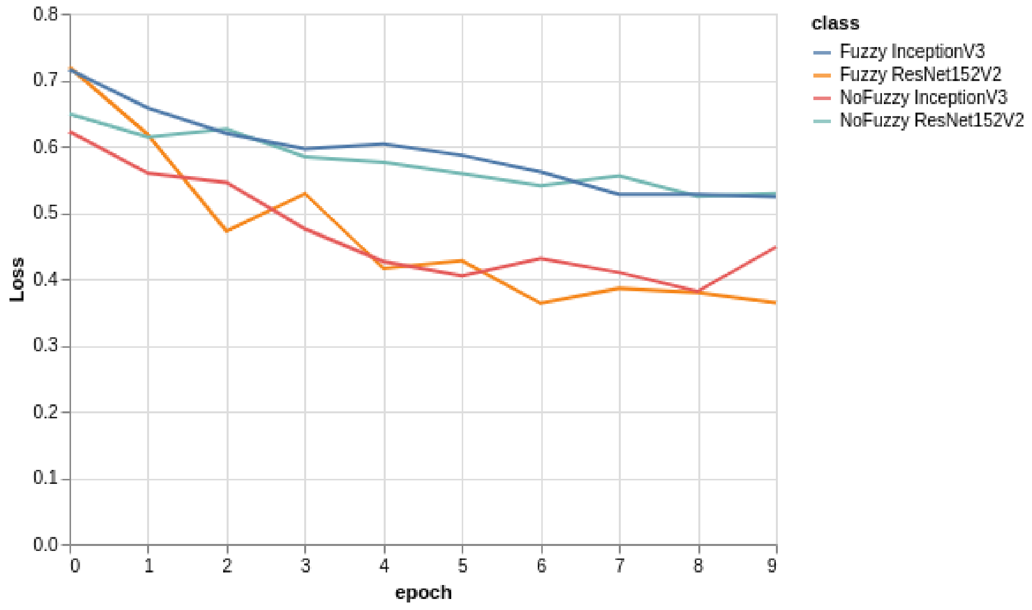
Table 8: f1 score of the ResNet152V2-VGG16 model

Class	Precision	Recall	F1-score	Support
0	1	0.94	0.97	36
1	0.95	1	0.97	36
accuracy			0.97	72
macro avg	0.97	0.97	0.97	72
weighted avg	0.97	0.97	0.97	72

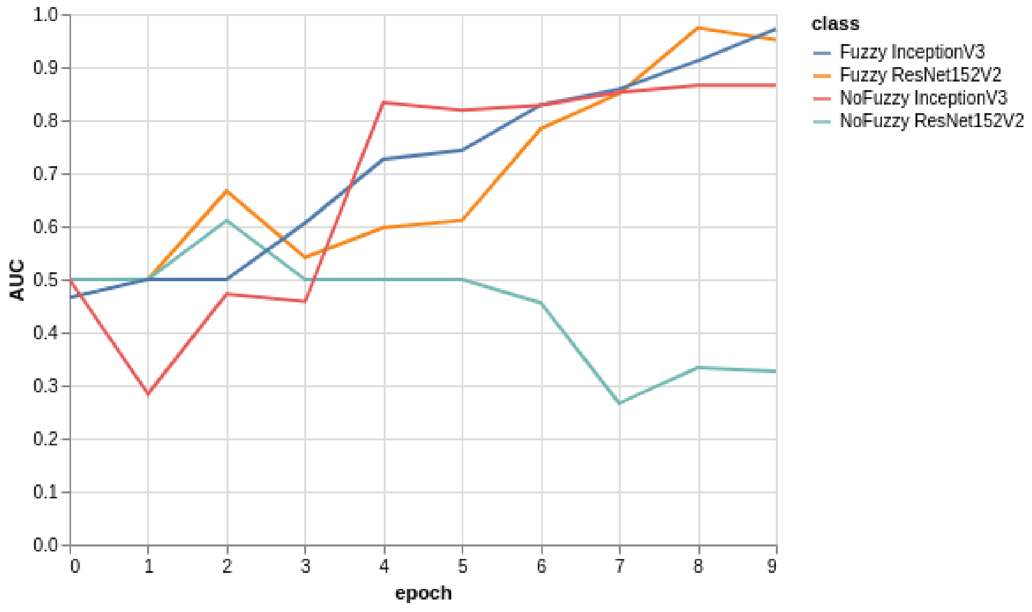
diagnostic systems, implying that the published results are optimistic and may overestimate the approaches' real predictive capacity [5][36][42]. This study does not intend to validate the differences or distinguish between usual pneumonia and pneumonia caused by COVID at this time, but that is a goal for future research.

This research presents an approach where transfer learning in conjunction with fuzzy filters allows the classification of CXRs. This study has an AUC value more significant than the presented researches in the literature review. Jin et al. show an AUC value of 0.979. Zheng et al. obtained an AUC value of 0.959. Chouhan achieved a recall value of 99.62%. Abdul et al. get an accuracy of 85%. However, the given solution must be validated in a more extensive sample set and clinical tests before using it in the clinical environment. A fuzzy filter improves the AUC and precision results. Future research developments must apply the techniques presented in this study in the segmentation of computed tomography scans.

Finally, we quantized the model with the quantize_model method (tfmot. quantization. keras .quantizemodel) in the TensorFlow framework, with an accuracy of 0.95.

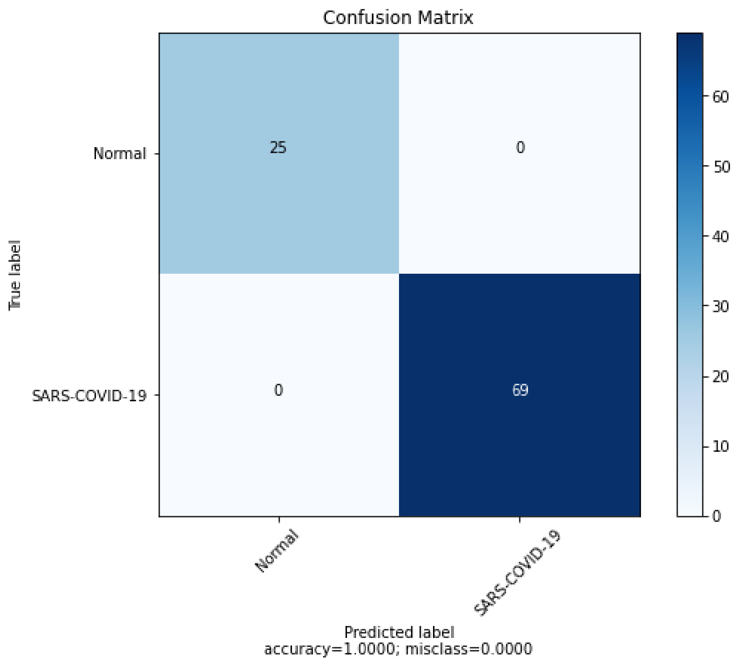


(a) Loss by epoch of best single input models



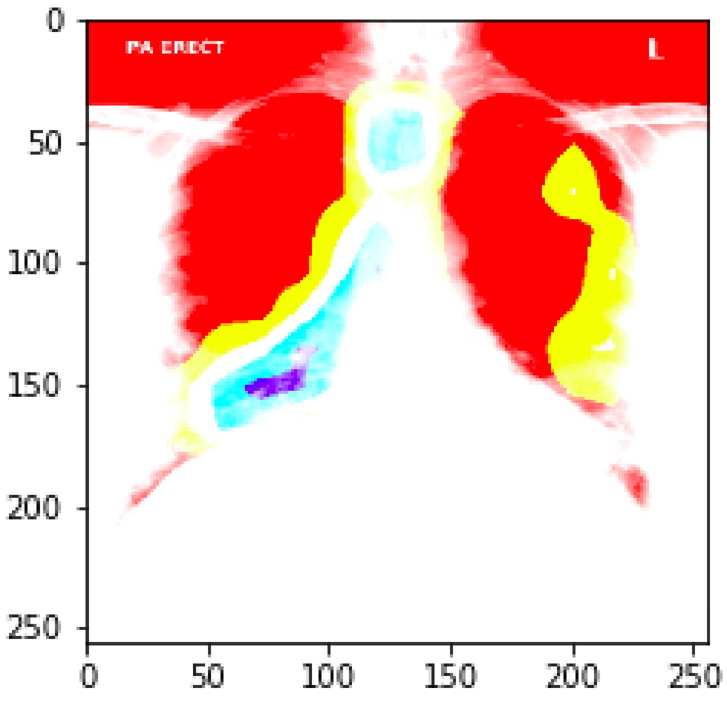
(b) AUC by epoch of best single input models

Figure 5: Loss and AUC by epoch results

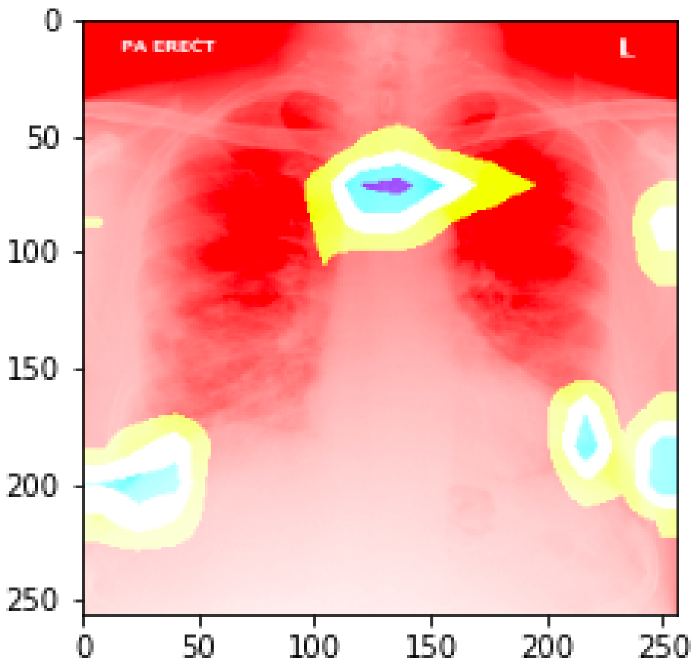


(a) Confusion Matrix of the ResNet152V2-VGG16 model

Figure 6: Loss and AUC by epoch results



(a) Class Activation Map with Fuzzy Filter



(b) Class Activation Map without Fuzzy Filter

Figure 7: Class Activation Map of of SARS sample X-Ray

6. Conclusion

In this paper, we show that by using transfer learning and leveraging pre-trained models, we can achieve very high accuracy in detecting COVID-19. Also, together with the fuzzy filter, this study shows that it is possible to achieve a recall of 1.0 with more than one pre-trained model. The best model was a combination of VGG16 and ResNet152V2. Finally, using quantization technology, we achieve an accuracy of 0.95. Despite the fact that we achieved good COVID-19 detection accuracy, sensitivity, and specificity, this does not imply that the solution is ready for production, especially given the small number of photos currently accessible about COVID-19 cases. The aim of this analysis is to provide radiologists, data scientists, and the research community with a multi-input CNN model that may be used to diagnose COVID-19 early, with the aim that it will be expanded upon to speed up research in this area.

Acknowledgment

QH received support from the Guangdong Province Science and Technology Plan Project (2018A050506086). VHCA received support from the Brazilian National Council for Research and Development (CNPq, Grant 304315/2017-6 and 430274/2018-1). Nauber Gois received support from FUNCAP (Project 06629650/2021). LZ received support from the Research Start-up Funds of DGUT (GC300502-60). LY received support from the Guangdong Regular Institutions of Characteristic Innovation Project (2017KTSCX176) and the Key Laboratory of Robotics and Intelligent Equipment of Guangdong Regular Institutions of Higher Education (2017KSYS009).

References

[1] Abdel-Basset M, Mohamed R, Elhoseny M, Chakrabortty RK, Ryan M (2020) A Hybrid COVID-19 Detection Model Using an Improved Marine Predators Algorithm and a Ranking-Based Diversity Reduction Strategy. IEEE Access 8:79,521–79,540, DOI 10.1109/ACCESS.2020.2990893

[2] Albashah NLSB, Asirvadam VS, Dass SC, Meriaudeau F (2019) Segmentation of blood clot MRI images using intuitionistic fuzzy set theory. 2018 IEEE EMBS Conference on Biomedical Engineering and Sciences, IECBES 2018 - Proceedings pp 533–538, DOI 10.1109/IECBES.2018.8626678

[3] Bhaswara ID, Marsiano AFD, Suryani OF, Adji TB, Ardiyanto I (2019) Class Activation Mapping-Based Car Saliency Region and Detection for In-Vehicle Surveillance. IES 2019 - International Electronics Symposium: The Role of Techno-Intelligence in Creating an Open Energy System Towards Energy Democracy, Proceedings (1):349–353, DOI 10.1109/ELECSYM.2019.8901604

[4] Budhiman A, Suyanto S, Arifianto A (2019) Melanoma Cancer Classification Using ResNet with Data Augmentation. 2019 2nd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2019 pp 17–20, DOI 10.1109/ISRITI48646.2019.9034624

[5] Catalá ODT, Igual IS, Pérez-Benito FJ, Escrivá DM, Castelló VO, Llobet R, Peréz-Cortés JC (2021) Bias analysis on public x-ray image datasets of pneumonia and covid-19 patients. IEEE Access 9:42,370–42,383

[6] Charuchinda P, Kasetkasem T, Kumazawa I, Chanwimaluang T (2019) On the use of class activation map for land cover mapping. Proceedings of the 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2019 pp 653–656, DOI 10.1109/ECTI-CON47248.2019.8955345

[7] Chouhan V, Singh SK, Khamparia A, Gupta D, Tiwari P, Moreira C, Damaševičius R, de Albuquerque VHC (2020) A novel transfer learning based approach for pneumonia detection in chest X-ray images. Applied Sciences (Switzerland) 10(2), DOI 10.3390/app10020559

[8] Chow LS, Rajagopal H (2017) Modified-BRISQUE as no reference image quality assessment for structural MR images. Magnetic Resonance Imaging 43:74–87, DOI 10.1016/j.mri.2017.07.016

[9] De Souza RWR, De Oliveira JVC, Passos LA, Ding W, Papa JP, Albuquerque V (2019) A Novel Approach for Optimum-Path Forest Classification Using Fuzzy Logic. IEEE Transactions on Fuzzy Systems 6706(c):1–1, DOI 10.1109/tfuzz.2019.2949771

[10] Ding W, Abdel-Basset M, Eldrandaly KA, Abdel-Fatah L, de Albuquerque VHC (2020) Smart supervision of cardiomyopathy based on fuzzy harris hawks optimizer and wearable sensing data optimization: A new model. IEEE Transactions on Cybernetics

[11] Dourado CM, Da Silva SPP, Da Nóbrega RVM, Rebouças Filho PP, Muhammad K, De Albuquerque VHC (2020) An open ioht-based deep learning framework for online medical image recognition. IEEE Journal on Selected Areas in Communications

[12] Epalle TM, Song Y, Liu Z, Lu H (2021) Multi-atlas classification of autism spectrum disorder with hinge loss trained deep architectures: Abide i results. Applied Soft Computing 107:107,375

[13] Gattinoni L, Chiumello D, Rossi S (2020) Covid-19 pneumonia: Ards or not?

[14] Jabbar SI, Day CR, Chadwick EK (2019) Using Fuzzy Inference system for detection the edges of Musculoskeletal Ultrasound Images. IEEE International Conference on Fuzzy Systems 2019-June:1–7, DOI 10.1109/FUZZ-IEEE.2019.8858971

[15] Jin S, Wang B, Xu H, Luo C, Wei L, Zhao W, Hou X, Ma W, Xu Z, Zheng Z, Sun W, Lan L, Zhang W, Mu X, Shi C, Wang Z, Lee J, Jin Z, Lin M, Jin H, Zhang L, Guo J, Zhao B, Ren Z, Wang S, You Z, Dong J, Wang X, Wang J, Xu W (2020) AI-assisted CT imaging analysis for COVID-19 screening: Building and deploying a medical AI system in four weeks. medRxiv pp 1–22, DOI 10.1101/2020.03.19.20039354

[16] Jin W, Dong S, Dong C, Ye X (2021) Hybrid ensemble model for differential diagnosis between covid-19 and common viral pneumonia by chest x-ray radiograph. Computers in biology and medicine 131:104,252

[17] Kala R, Deepa P (2017) Removal of rician noise in MRI images using bilateral filter by fuzzy trapezoidal membership function. 2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS 2017 DOI 10.1109/ICACCS.2017.8014648

[18] Kala R, Deepa P (2018) Intuitionistic Fuzzy C-Means Clustering Using Rough set for MRI Segmentation. Proceedings of the 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT 2018 pp 1–9, DOI 10.1109/ICCTCT.2018.8550853

[19] Karim Z, Van Zyl T (2020) Deep learning and transfer learning applied to sentinel-1 DInSAR and Sentinel-2 optical satellite imagery for

change detection. 2020 International SAUPEC/RobMech/PRASA Conference, SAUPEC/RobMech/PRASA 2020 DOI 10.1109/SAUPEC/RobMech/PRASA48453.2020.9041139

[20] Kristiani E, Yang CT, Huang CY (2020) ISEC: An Optimized Deep Learning Model for Image Classification on Edge Computing. IEEE Access 8:27,267–27,276, DOI 10.1109/ACCESS.2020.2971566

[21] Kumar EB, Sundaresan M (2014) Edge detection using trapezoidal membership function based on fuzzy's mamdani inference system. 2014 International Conference on Computing for Sustainable Global Development, INDIACom 2014 pp 515–518, DOI 10.1109/IndiaCom.2014.6828012

[22] Lin J, Wang N, Zhu H, Zhang X, Zheng X (2021) Fabric defect detection based on multi-input neural network. In: 2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), IEEE, pp 458–463

[23] Liu X, Liu Z, Wang G, Cai Z, Zhang H (2017) Ensemble Transfer Learning Algorithm. IEEE Access 6:2389–2396, DOI 10.1109/ACCESS.2017.2782884

[24] Lopez-Molina C, Bustince H, Fernandez J, De Baets B (2011) Generation of fuzzy edge images using trapezoidal membership functions. Proceedings of the 7th Conference of the European Society for Fuzzy Logic and Technology, EUSFLAT 2011 and French Days on Fuzzy Logic and Applications, LFA 2011 1(1):327–333, DOI 10.2991/eusflat.2011.73

[25] Madrid-Herrera L, Chacon-Murguia MI, Posada-Urrutia DA, Ramirez-Quintana JA (2019) Human image complexity analysis using a fuzzy inference system. In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, pp 1–6

[26] Mehta S, Paunwala C, Vaidya B (2019) CNN based traffic sign classification using adam optimizer. 2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019 (Iciccs):1293–1298, DOI 10.1109/ICCS45141.2019.9065537

[27] Muhammad K, Khan S, Del Ser J, de Albuquerque VHC (2020) Deep learning for multigrade brain tumor classification in smart healthcare systems: A prospective survey. IEEE Transactions on Neural Networks and Learning Systems

[28] Ng HP, Ong SH, Foong KWC, Goh PS, Nowinski WL (2001) Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm View project Neuro vasculature modeling View project MEDICAL IMAGE SEGMENTATION USING K-MEANS CLUSTERING AND IMPROVED WATERSHED ALGORITHM. 2006 IEEE Southwest Symposium on Image Analysis and Interpretation (February 2001):61–65, DOI 10.1109/SSIAI.2006.1633722

[29] Ohata EF, Bezerra GM, das Chagas JVS, Neto AVL, Albuquerque AB, de Albuquerque VHC, Reboucas Filho PP (2020) Automatic detection of covid-19 infection using chest x-ray images through transfer learning. IEEE/CAA Journal of Automatica Sinica

[30] Popovic A, de la Fuente M, Engelhardt M, Radermacher K (2007) Statistical validation metric for accuracy assessment in medical image segmentation. International Journal of Computer Assisted Radiology and Surgery 2(3-4):169–181, DOI 10.1007/s11548-007-0125-1

[31] Pour AM, Seyedarabi H, Jahromi SHA, Javadzadeh A (2020) Automatic Detection and Monitoring of Diabetic Retinopathy using Efficient Convolutional Neural Networks and Contrast Limited Adaptive Histogram Equalization. IEEE Access pp 1–1, DOI 10.1109/access.2020.3005044

[32] Qu Z, Mei J, Liu L, Zhou DY (2020) Crack detection of concrete pavement with cross-entropy loss function and improved VGG16 network model. IEEE Access 8:54,564–54,573, DOI 10.1109/ACCESS.2020.2981561

[33] Rajaraman S, Siegelman J, Alderson PO, Folio LS, Folio LR, Antani SK (2020) Iteratively Pruned Deep Learning Ensembles for COVID-19 Detection in Chest X-rays. IEEE Access pp 1–1, DOI 10.1109/access.2020.3003810

[34] Rodrigues MB, Da Nôbrega RVM, Alves SSA, Rebouças Filho PP, Duarte JBF, Sangaiah AK, De Albuquerque VHC (2018) Health of things algorithms for malignancy level classification of lung nodules. IEEE Access 6:18,592–18,601

[35] Sandilya M, Nirmala SR (2018) Determination of reconstruction parameters in Compressed Sensing MRI using BRISQUE score. 2018 International Conference on Information, Communication, Engineering and Technology, ICICET 2018 15:1–5, DOI 10.1109/ICICET.2018.8533865

[36] Santa Cruz BG, Bossa MN, Sölter J, Husch AD (2021) Public covid-19 x-ray datasets and their impact on model bias–a systematic review of a significant problem. Medical image analysis 74:102,225

[37] Santos MA, Munoz R, Olivares R, Rebouças Filho PP, Del Ser J, de Albuquerque VHC (2020) Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook. Information Fusion 53:222–239

[38] Selvachandran G, Quek SG, Lan LTH, Son LH, Long Giang N, Ding W, Abdel-Basset M, Albuquerque VHC (2019) A New Design of Mamdani Complex Fuzzy Inference System for Multi-attribute Decision Making Problems. IEEE Transactions on Fuzzy Systems 6706(c):1–1, DOI 10.1109/tfuzz.2019.2961350

[39] Sharma T, Singh V, Sudhakaran S, Verma NK (2019) Fuzzy based Pooling in Convolutional Neural Network for Image Classification. IEEE International Conference on Fuzzy Systems 2019-June:1–6, DOI 10.1109/FUZZ-IEEE.2019.8859010

[40] Shi F, Wang J, Shi J, Wu Z, Wang Q, Tang Z, He K, Shi Y, Shen D (2020) Review of Artificial Intelligence Techniques in Imaging Data Acquisition, Segmentation and Diagnosis for COVID-19. IEEE Reviews in Biomedical Engineering 3333(c):1–13, DOI 10.1109/RBME.2020.2987975

[41] Sun Y, Binti Hamzah FA, Mochizuki B (2020) Optimized Light-Weight Convolutional Neural Networks for Histopathologic Cancer Detection. LifeTech 2020 - 2020 IEEE 2nd Global Conference on Life Sciences and Technologies pp 11–14, DOI 10.1109/LifeTech48969.2020.1570619224

[42] Vaid S, Kalantar R, Bhandari M (2020) Deep learning covid-19 detection bias: accuracy through artificial intelligence. International Orthopaedics 44(8):1539–1542

[43] Varela-Santos S, Melin P (2021) A new approach for classifying coronavirus covid-19 based on its manifestation on chest x-rays using texture features and neural networks. Information sciences 545:403–414

[44] Waheed A, Goyal M, Gupta D, Khanna A, Al-Turjman F, Pinheiro PR (2020) CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for

Improved Covid-19 Detection. IEEE Access 8:91,916–91,923, DOI 10.1109/ACCESS.2020.2994762

[45] Wang G, Liu X, Li C, Xu Z, Ruan J, Zhu H, Meng T, Li K, Huang N, Zhang S (2020) A Noise-robust Framework for Automatic Segmentation of COVID-19 Pneumonia Lesions from CT Images. IEEE Transactions on Medical Imaging 1(c):1–1, DOI 10.1109/tmi.2020.3000314

[46] Wu MN, Lin CC, Chang CC (2007) Brain tumor detection using color-based K-means clustering segmentation. Proceedings - 3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2007 2:245–248, DOI 10.1109/IIHMSP.2007.4457697

[47] Yan T, Wong PK, Ren H, Wang H, Wang J, Li Y (2020) Automatic distinction between covid-19 and common pneumonia using multi-scale convolutional neural network on chest ct scans. Chaos, Solitons & Fractals 140:110,153

[48] Yemini M, Zigel Y, Lederman D (2019) Detecting Masses in Mammograms using Convolutional Neural Networks and Transfer Learning. 2018 IEEE International Conference on the Science of Electrical Engineering in Israel, ICSEE 2018 pp 1–4, DOI 10.1109/ICSEE.2018.8646252

[49] Zheng C, Deng X, Fu Q, Zhou Q, Feng J, Ma H, Liu W, Wang X (2020) Deep Learning-based Detection for COVID-19 from Chest CT using Weak Label. medRxiv p 2020.03.12.20027185, DOI 10.1101/2020.03.12.20027185

[50] Zuo H, Lu J, Zhang G, Liu F (2019) Fuzzy Transfer Learning Using an Infinite Gaussian Mixture Model and Active Learning. IEEE Transactions on Fuzzy Systems 27(2):291–303, DOI 10.1109/TFUZZ.2018.2857725