Abstract

COVID-19 is a severe respiratory tract infections. COVID-19 caused by SARS-CoV-2 can readily spread through a contact with an infected person. Monotonically increasing SARS-CoV-2 infections have not only wasted lives but also severely damaged the financial systems of both developing and developed countries. This high spread rate pressure on the health care systems which rise the need to fast methods for diagnosing this disease. Convolutional Neural Networks (CNN) show a great success for various computer vision tasks. However, CNN is a scale-variant model and computationally expensive. In this Thesis, a novel architectures are proposed for multiscale feature extraction and classification and lightweight architecture for COVID-19 diagnosing. The proposed I which is a lightweight CNN model exploits spatial kernel separability to reduce the number of the training parameters to a large extent and regularize the model to only learns linear kernel. Furthermore, This model uses residual connection and batch normalization extensively to maintain the network stability during the training process and provide the model with the regularization effect to reduce the overfitting. Proposed CNN II learns multiscale features using a pyramid of shared convolution kernels with different atrous rates. This scale invariant CNN uses attention based mechanism that is used to guide and select correct scale for each input. Proposed CNN II is an end-to-end trainable network and exploit a novel augmentation technique, Texture Augmentation, to reduce the overfitting. The lightweight architecture is trained using QaTa-Cov19 benchmark dataset achieving 100% for accuracy, sensitivity, precision and F1-score with a very low parameter count (150K) compared with the other methods in the literature. Proposed method II achieved a 0.9929 for $F1 - score$ tested on QaTa-Cov19 benchmark dataset with a total of $5,040,571$ trainable parameters.

Chapter 1

Proposed Methodology I

In this chapter, a novel architecture is introduced for detecting COVID-19 that is designed to be lightweight. The architecture is based on two key components: spatial kernel separability and residual connection. By exploiting spatial kernel separability, the number of training parameters is significantly reduced, making the model more computationally efficient. Additionally, residual connections are used extensively to maintain network stability during the training process and to provide the model with regularization effects that reduce overfitting. This combination of spatial kernel separability and residual connections creates a lightweight architecture that is highly effective at detecting COVID-19. Overall, this chapter provides a valuable contribution to the field of COVID-19 detection by introducing a novel and efficient architecture that can help to identify the disease quickly and accurately.

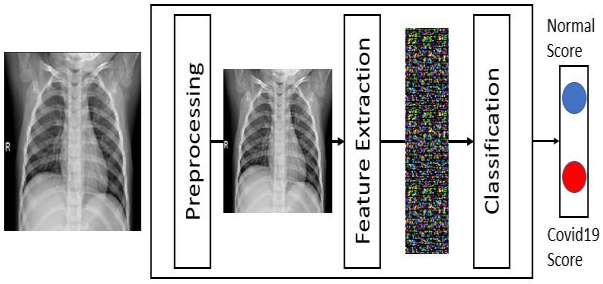

FIGURE 1.1: The phases of the proposed method I.

1.1 Methodology I

In this section, a proposed method I to detect COVID-19 disease from chest X-ray images is presented. The proposed method exploits the CNN model to classify the input chest X-ray image into one of two categories; normal case or

Covid-19 case. The proposed method I consists of three phases: preprocessing, feature extraction, and classification. The proposed method phases are shown in Fig. 1.1.

1.1.1 Preprocessing Phase

The preprocessing phase is responsible for resizing and normalizing the input chest X-ray images. The pre-processing phase is employed to maintain the numerical stability of the model and reduce the co-variance shift [1]. In addition, this phase leads the learning model of the CNN model to reduce the required overhead to adapt to the different scales of different features of the input data. Reshaping size is determined empirically. The input chest X-ray image is re-sized and then adapted and normalized to a normal distribution as follows:

$$Y := \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{1.1}$$

where μ and σ are the mean and standard deviation of chest X-ray image (X), respectively.

After re-sizing the input chest X-ray image, the input image is normalized to have a zero mean and unit standard deviation. Then, the image can be scaled and shifted with a normalization parameter which is determined and adapted by the training dataset during the training process according to the following equation:

$$Z := w_1 Y + w_2 \tag{1.2}$$

where w_1 and w_2 are a trainable parameter.

Unlike the normalization method presented in [2], the batch normalization process presented in this paper has a z-score normalization parameter that is used in both the training and validation phases.

1.1.2 Feature Extraction and Classification

CNN models achieved outstanding success in image recognition [3]. This phase is responsible for extracting spatial features from the normalized chest X-ray image using a tailored CNN model. This phase is based on learning the CNN model by the input preprocessed chest X-ray images. The design of the tailored CNN model is described as follows:

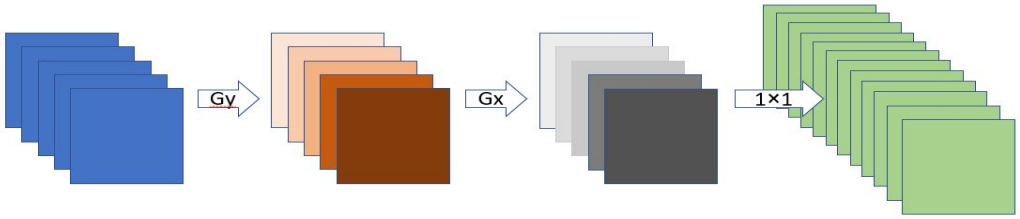

FIGURE 1.2: Separable convolution Gy and Gx have kernel size of $M \times 1$ and $1 \times M$. The combination of these kernels is approximately a $M \times M$ kernel and depth-wise convolution is applied by a 1×1 convolution. The output depth is padded with zeros to have the same spatial size of Gy, Gx. Gy, Gx are performed channel-wise.

Separable CNN kernels

Kernel separability [4] [5] is based on decomposing a 2D convolution kernel to linear combinations of two 1D vectors which leads to a large reduction in the total number of resulting parameters. For example, a 2D kernel of size 9×9 has a total number of $9^2 = 81$ trained parameters. Whereas in the case of separating this 2D kernel to linear combinations of two 1D vectors of sizes 9×1 and 1×9, this results in a total number of $9 + 9 = 18$ trained parameters. As a consequence, kernel separability reduces the number of CNN model operations (such as multiplication and addition). A 2D kernel of $k \times k$ applied for a 2D signal with spatial dimensions of $M \times N$ has a total number of $(N-4)(M-4) \times k^2$ operations but in case of applying kernel separability yields $2(N-4)(M-4)k$ operations. The flow of separated convolution operation is summarized in Fig. 1.2. Fig. 1.3 represents the structure, denoted by the Separated Convolutional Layer, used in the proposed method with kernel size of $(M \times N)$ and satisfying the convolutional kernel separability. Separated Convolutional Layer is composed of three consecutive layers. The first convolutional layer has a kernel size of $(M \times 1)$ and the number of convolutional neurons and filters are equal to the number of channels as the input feature map and the convolution operations are performed channel-wise. The second layer operates in the same way as the first layer but it has a kernel of size $(1 \times M)$. The third layer is the convolutional layer with a kernel of size (1×1) and the number of convolutional neurons is N. The collaboration of the three layers are connected to perform similarly to the convolutional layer with a kernel size of $(M \times M)$ and the number of neuron and filter is the same as N but with a large difference in the performance.

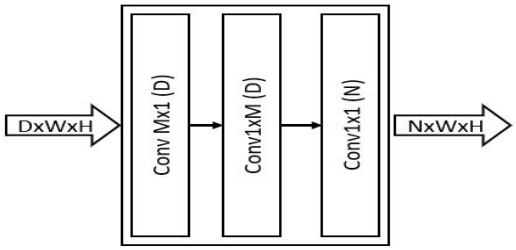

FIGURE 1.3: Separated Convolutional Layer
composed of three consecutive layers. The first Convolutional layer has a kernel size of $(M \times 1)$ and D convolutional neuron. The second layer operates in the same way as the first layer but it has a kernel of size $(1 \times M)$ and D convolutional neuron. The third layer is the convolutional layer with a kernel of size (1×1) and the number of convolutional neurons is N.

Batch Normalization and Activation function

In the proposed method linear separable convolutional kernels are followed by a batch normalization and an activation function. Rectified Linear Unit (ReLU) [6] is a nonlinear activation that allows the network to fit and approximate highly non-linear dataset distribution. The proposed method employs batch normalization which is described in [2].

Batch Normalization [2] reduces internal covariate shift produced as a result of moving between layers during the feedforward procedure [2]. Batch Normalization makes the loss landscape smoother and reduces the number of saddle points [7] which allows to use of higher learning rates. Using a higher learning rate makes the network training faster [2]. Batch normalization reduces the vanishing gradient problem and exploding gradient problem as it makes the resulted activation scale independent from the trainable parameter scale [2]. Batch normalization has the effect of regularization because of the inherited randomness when selecting the batch sample [2] which help the generalization to unseen chest X-ray image.

Deep and larger receptive field Network design

Deeper convolutional neural network design is a very important task for any image recognition task [8]. Training a deeper network is very expensive and has many challenges such as vanishing gradient problem, exploding gradient problem, and degradation problem [8]. The exploding gradient problem occurs when the gradient update becomes very large (approaching infinity) resulting in the network diversion. A vanishing gradient problem occurs when the gradient update becomes very small (approaching zero) resulting in preventing the

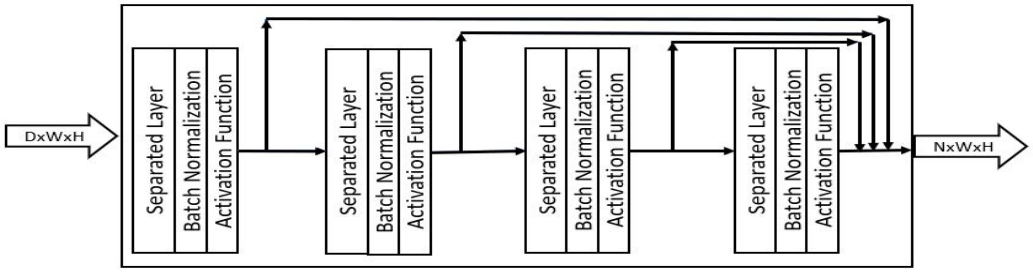

FIGURE 1.4: The stack of residual separated block (RSB) consists of four layers of separated convolutional layer each of which is followed by batch normalization and activation function.

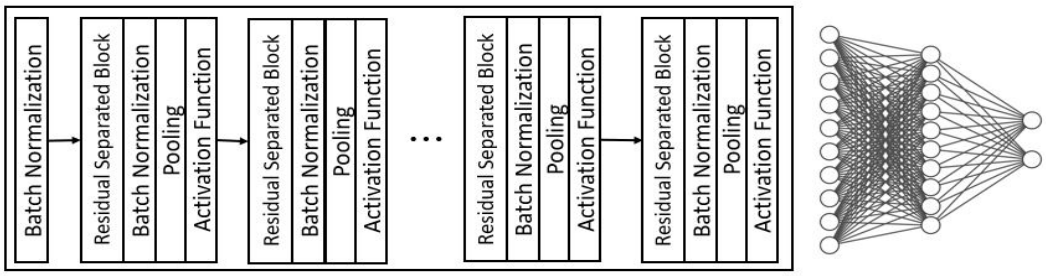

FIGURE 1.5: The complete proposed tailored CNN architecture.

parameter update for early layers [2] and preventing the network from learning new patterns. Batch normalization [2] and the use of ReLU activation function [9] alleviate these two problems.

The deep layers of CNN networks sometimes need to approximate the identity function which is not a simple task, especially with the existence of a non-linear function. Residual connection [8] overcomes this problem by using skip connection as shown in Fig. 1.4. Fig. 1.4 represents the building block layer of the feature extraction phase, denoted by a stack of Residual Separated Block (RSB). RSB consists of four layers of separated convolutional layers, each layer is followed by a batch normalization and an activation function. It has an output of depth N where each sublayer produces an output of depth $N/4$ which is concatenated at the end of the layer to produce a depth N. RSB produces a feature map that includes both low-level features and high-level features.

Unlike the traditional neural network, which is fully connected to the previous layer, the convolutional neural network is connected locally to a local region of the previous feature map. This introduces the concept of the network receptive field [10]. The receptive field should be large enough to capture large patterns in the input chest X-ray image. Therefore, any consecutive convolutional layers in the proposed method without a pooling layer in between a larger

kernel size are used in one of them. Residual Separated block, RSB, in Fig. 1.4 may have kernel sizes of 3, 5, 7, and 9, respectively.
Fig. 3.2 Represent a complete CNN architecture.

1.2 Summary

In this chapter, the proposed lightweight CNN architecture for COVID-19 detection is designed with the concept of spatial separability in mind. The spatial separability of the convolutional kernel is used to enforce the learning of linear kernels, which reduces the number of training parameters and improves computational efficiency. Essentially, the model is designed to recognize patterns in the data that are linearly separable, which enables the use of simpler and more efficient models.

The proposed architecture comprises separated kernel convolutional layers that are connected by a residual connection. The use of separated kernel convolutional layers helps to reduce the number of training parameters, while the residual connection improves network stability during the training process. The residual connection enables the model to retain important features while also discarding unimportant ones, which helps prevent overfitting.

Batch normalization is used in the proposed architecture to maintain network stability during the training process. The technique standardizes the inputs of each layer, which improves the convergence rate of the model. This is done by normalizing the layer inputs to have zero mean and unit variance, which helps prevent internal covariate shifts. Internal covariate shift refers to the change in the distribution of network activations that occurs during training, which can slow down the convergence rate.

In summary, the proposed lightweight CNN architecture for COVID-19 detection is based on the spatial separability of the convolutional kernel, with separated kernel convolutional layers connected by a residual connection. Batch normalization is used to maintain network stability during training, which improves the convergence rate of the model. By combining these techniques, the proposed architecture offers an efficient, accurate, and stable method for detecting COVID-19.

Chapter 2

Proposed Methodology II

CNN, like many computer vision models, is a scale-variant [11] model such that it cannot recognize objects at various scales unless it explicitly trained to recognize such objects. Data augmentation can accomplish some degree of invariance as it allows the network to be trained with distorted samples, but it not the case for pneumonia scales. This chapter presents a CNN architecture that learns multiscale features using scale pyramid of the CNN's internal feature maps. Scale pyramid is constructed using atrous convolution of various dilation rates. The correct scale from scale pyramid that allows minimization of the objective function loss is selected using the spatial attention mechanism.

2.1 Methodology II

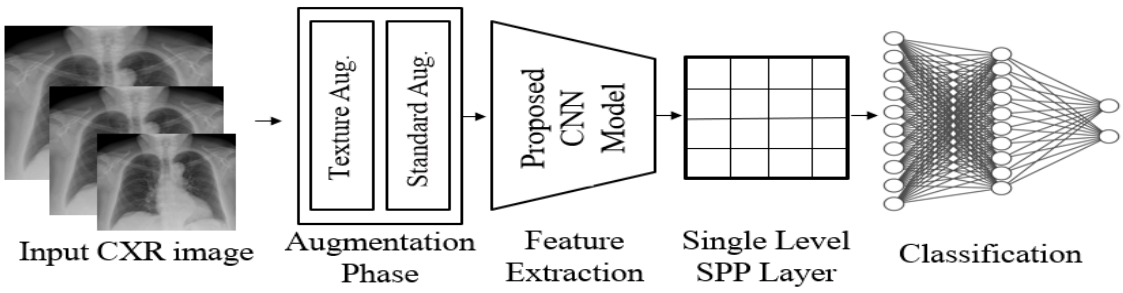

FIGURE 2.1: Proposed method for COVID-19 classification from CXR images.

The proposed system presented in this chapter proposes a novel CNN micro-architecture model for learning scale-invariant features from row input CXR images and then classifies these features into normal or COVID-19 cases. Fig. 2.1

illustrates trainable end-to-end pipeline of the proposed system. The proposed system depends on a novel Spatially weighted Atrous Spatial Pyramid Pooling (SWASPP) to extract multi-scale features of input CXR images. A novel attention module is then used to fuse the extracted these multi-scale features and select relevant features' scale that the next layer should consider.

2.1.1 Data augmentation

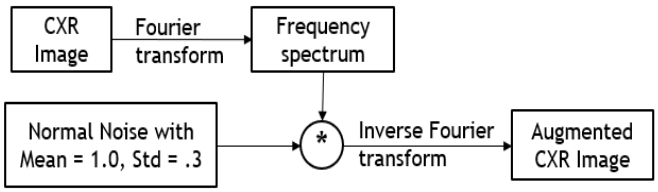

FIGURE 2.2: Texture Augmentation module

The first phase of the proposed CXR classification system is data augmentation phase. Data augmentation is used to reduce the overfitting by artificially enlarge the training dataset [9] using label preserving transformation. Data augmentation phase introduces a degree invariance to a distortion transformation such as the flipping and rotation. The input CXR images are augmented using texture augmentation. Texture augmentation is performed by introducing a multiplicative normally distributed noises to the frequency spectrum of the input image. CXR image is transformed to the frequency spectrum using the fourier transform. Noise is modeled using $\mathcal{N}(\mu = 1, \sigma = 0.3)$. Fig. 2.2 illustrates texture augmentation process for frequency distortion of the CXR image. Fig. 2.3 shows the original CXR image and the corresponding frequency distorted CXR image. A standard augmentation techniques such as random rotation, horizontal flipping, and vertical flipping are included in the augmentation process.

2.1.2 Spatially Weighted Atrous Spatial Pyramid Pooling

Atrous convolution is a powerful technique for adjusting the resolution of convolutional kernels. This allows to effectively enlarge the field-of-view of the kernel

FIGURE 2.3: Texture Augmentation
The resulting CXR image from Texture augmentation **left**: is the original image. **Right** is the augmented CXR Image

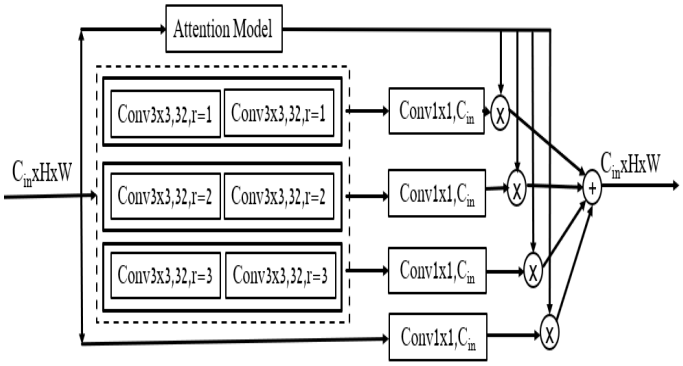

FIGURE 2.4: Spatially weighted atrous spatial Pyramid Pooling (SWASPP) interal layers within dashed square are parameter shared.

without increasing neither the number of kernel parameters nor the computational complexity of the convolution operation. Atrous convolution is equivalent to performing downsampling and then performing convolution with original kernel without dilation. As a result different dilation rates of the kernel corresponding to different downsampling degrees. A novel spatially weighted atrous spatial pyramid pooling (SWASPP) micro-architecture is presented that exploit the scale space of the CNN's feature maps. Fig. 2.4 shows the architecture of the SWASPP. In Fig. 2.4, internal pipelines, bounded by dashed-line square, are parameter-shared and each pipeline of these has a different dilation rates. These pipelines are responsible for extracting multi-scale, scale invariant, features. Sharing of the parameters enforce these pipelines to learn features that exists at multiple levels of scale-pyramid and hence scale-invariance. For a given input CXR image, three scales feature maps are produced.

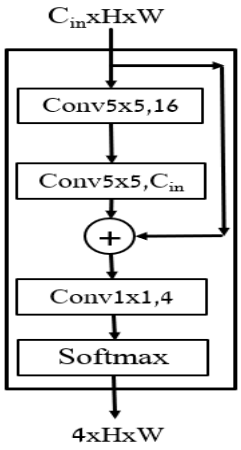

FIGURE 2.5: Attantion module structure used by SWASPP micro-architecture

To fuse these feature maps produced by different pipeline of the SWASPP from the input feature map, an attention module is emerged. Attention module can be thought as a pixel level classification of which scale does this pixel it belongs to. Fig. 2.5 illustrates the proposed attention module structure. Proposed attention module generates four heatmaps. The first three heatmaps correspond to the three scale feature maps while the remaining heatmap corresponds to the input feature map itself. These heatmaps are summed up to one (*i.e.,* for a spatial position (x, y), $\sum_{i=1}^{4} H(i, x, y) = 1$ where $H(i, x, y)$ is the i heatmap produced by the attention module). To make sure this property holds, softmax function is used.

The proposed mirco-architecture uses a pixel level weights produced by corresponding attention module rather than a single weight value for each scale. A single input CXR image may have multiple COVID-19 pneumonia scales which effectively lead to simply averaging the scale space when using single weight for each scale on scale space. In SWASPP, every convolution operation is followed by a BN and leakyReLU [9] non-linearity except the re-projection layers that used to project back to the input space is not followed by nonlinearity. BN allows the use of larger learning rate [2] and makes network stable during training [2]. BN makes the loss landscape of the optimization problem significantly smoother [7]. leakyReLU is used to reduce the vanishing gradient problem [9]. A bottleneck is introduced within both the attention module and multi-scale feature extractor pipeline. A bottleneck in SWASPP is used to project the input feature map of dimension $C_{in} \times H \times W$ to $32 \times H \times W$ then re-project back to $C_{in} \times H \times W$. Multi-scale feature extraction is preformed on the projected dimension. Same logic is applied to the attention module where the input feature map is projected to a dimension of $16 \times H \times W$. This bottleneck allows the efficient use of model capacity and reduce the network computational complexity [12]. It only allows the flow of important information and discarding irrelevant information.

2.1.3 Proposed CNN Architecture

SWASPP is densely stacked [12] together as Fig. 2.6 illustrates. This kind of connectivity allows implicit deep supervisions as each layer is effectively connected to the last layer using shorter path also facilitate feature reuse [12] and gradient flow. Residual layers are easier to optimize if the required mapping is the identity mapping or simply near to it [8]. Densely stacked SWASPP is denoted by (DSWASPP). Convolutional part of proposed model consists of stacking six DSWASPP layers such that the first four layers are interconnected using maxpooling to reduce the spatial size and enlarge the Network receptive field. A single level Spatial Pyramid Pooling (SPP) [13] is added after to produce a fixed size feature vector for a variable size input. SPP layer divides the input feature map into $10 \times 10 = 100$ bins then performs a max for each bin as an aggregation function.

The fixed length feature vector produced by SPP is used as an input to dropout [14] layer. Dropout layer randomly sets the activation of to 0 with a probability of 0.5. Dropout prevents the overfitting and reduce complex

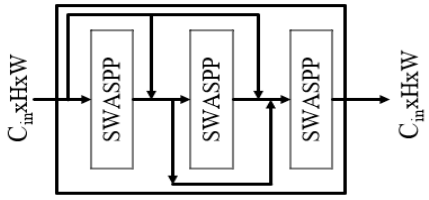

FIGURE 2.6: Densely connected SWASPP (DSWASPP): is a stack of densely connected SWASPP, such that the output of any SWASPP is Concatenated to the input of all next layers. All the three layers produce an output of dimension of $C_{in} \times H \times W$.

co-adaptation between the neurons allowing them to learn better representation [14]. It allow implicit ensempling of exponential number of sampled thin network from the original network which enhance the network performance [14]. The result of dropout layer is used as input to the classification network. Classification network consists of a fully connected layers with a 3 Dense layers such that the output layer is 2-neuron for binary classification *i.e)* COVID19 or not. Table 2.1 shows the details of the proposed architecture.

2.2 Summary

Convolutional Neural Networks (CNNs) have been widely used in computer vision tasks, including image classification, object detection, and segmentation. However, CNNs are known to be scale variant models, meaning that they can miss important features at different scales. To overcome this issue, various approaches have been proposed, such as shared networks, feature pyramid networks, and atrous convolution.

Atrous convolution, also known as dilated convolution, increases the receptive field of the convolutional kernel without increasing the number of parameters or computational complexity. In the proposed work II, atrous convolution is used to construct the scale space of the input feature, allowing the CNN to extract multiscale features.

Moreover, to select the correct scale and fuse multiple scales of the input feature map, a spatial attention module is used in the proposed work II. This attention mechanism guides the network to focus on the most relevant parts of the feature maps, which helps to improve the accuracy of the network.

To further enhance the performance of the network, a novel CNN architecture is proposed in which multiscale feature maps are internally produced and then fused using an attention-based mechanism. This approach leads to a compact representation of the input data via a bottleneck dimension introduced in

TABLE 2.1: Proposed CNN architecture of methodology II

Layer Name	**Proposed CNN Architecture of Methodology II**		
	Input Shape	***Output Shape***	***Param. Count***
Input layer	-	$1 \times 320 \times 320$	0
BatchNorm-1	$1 \times 320 \times 320$	$1 \times 320 \times 320$	2
DSWASPP-1	$1 \times 320 \times 320$	$32 \times 320 \times 320$	121,035
Maxpooling-1	$32 \times 320 \times 320$	$32 \times 160 \times 160$	0
DSWASPP-2	$32 \times 160 \times 160$	$64 \times 160 \times 160$	298,236
Maxpooling-2	$64 \times 160 \times 160$	$64 \times 80 \times 80$	0
DSWASPP-3	$64 \times 80 \times 80$	$128 \times 80 \times 80$	604,956
Maxpooling-3	$128 \times 80 \times 80$	$128 \times 40 \times 40$	0
DSWASPP-4	$128 \times 80 \times 80$	$128 \times 80 \times 80$	784,092
DSWASPP-5	$128 \times 80 \times 80$	$128 \times 80 \times 80$	784,092
DSWASPP-6	$128 \times 80 \times 80$	$128 \times 80 \times 80$	784,092
SPP-1	$128 \times 80 \times 80$	12800	0
Dropout-1	12800	12800	0
FC-1	12800	128	1,638,528
FC-2	128	128	16,512
FC-3	128	64	8,256
FC-4	64	2	130
Softmax	2	2	0
Total Number of Parameter			5,040,571

Any linear combination is followed by BN and leakyReLU nonlinearity excluding re-projection layer of the SWASPP modules

both the multiscale feature extractor module and the attention module. Overall, these techniques and architectures help to improve the performance and accuracy of CNNs for computer vision tasks.

Chapter 3

Experimental Results

In this chapter proposed methodologies are evaluated and compared with the related work.

All models are Trained using QaTa-Cov-19 [15] dataset using NVIDIA Tesla P-100 GPU and programmed using PyTorch.

3.1 QaTa-COV19 Dataset

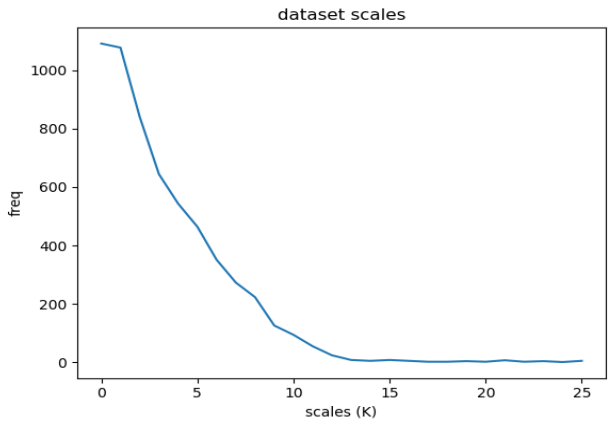

FIGURE 3.1: Pneumonia Scales of QaTa-COV19-v1, Y-axis represents the frequency, number of occurrences, of pneumonia with a particular area

QaTa-COV19 is a benchmark dataset for COVID-19 detection and Segmentation from CXR images. All models used for comparison are trained using QaTa-COV19-v1. Qata-COV19-v1 consists of 4603 COVID-19 CXRs and $120,013$ control group CXRs. A balanced number of samples for the two classes is used, namely 4603 CXR images for each class to train the models. Pneumonia Scales

of QaTa-COV19-v1 do not exhibit a uniform distribution. The scale of the Pneumonia can be defined as the number, area, of 8-neighbor-connected pixels labeled as COVID-19 pneumonia. QaTa-COV19-v1 provides a binary mask of 2951 COVID-19 CXR images which can be used for approximating the distribution of scales across the dataset. Fig. 3.1 illustrates the statistical distribution of QaTa-COV19 scales. The non-uniform distribution of the scales allows the CNN models to only recognize the small scales and not the large scales.

3.2 Evaluation of the Methodology I

Experiments are conducted on a Lenovo Z50-70 with Intel CORE i7-4510U CPU 2.00 GHz, 8GB RAM, NVIDIA GeForce 840M GPU; and with Python and PyTorch library.

3.2.1 Details of the Proposed Architecture

The Proposed architecture is composed of Convbase and Densebase. Convbase is composed of a 6 feature extraction modules *(FX)* preceded by batch normalization layer. Each FX module can be considered a sub-sequential model consisting of an RSB layer followed by Batch Normalization, Max-pooling, and LeakyReLU activation function. The Densebase is two fully connected layers that classify the Convbase output.

3.2.2 Hyperparameter Specification

All input chest X-ray images are resized to be 200×200 . After resizing the input images, these images are fed the Convbase model part which consists of 6 layers of residual separated block. Each residual separated block is followed with batch normalization and LeakyReLU [6] as an activation function. The output depth of each residual separated block is 4×16, 4×32, 4×64, 4×64, 4×64, and 4×16, respectively. The output of the Convbase model part is a 1D feature vector of 576 in length. The densebase model part consists of two hidden layers. Each layer has a size of 64 and the output layer of size 2. Each layer of Densebase layers is fully connected to its previous layer. The activation function used in the densebase model part is LeakyReLU. Table 3.1 summarizes the architecture hyperparameters.

TABLE 3.1: The proposed architecture hyperparameters

Layer Number	**Layer Size**	**Activation Function**
RSBLayer1	4 × 16	LeakyReLU
RSBLayer2	4 × 23	LeakyReLU
RSBLayer3	4 × 64	LeakyReLU
RSBLayer4	4 × 64	LeakyReLU
RSBLayer5	4 × 64	LeakyReLU
RSBLayer6	4 × 16	LeakyReLU
Flatten The Feature maps to 1D 576 feature vector		
LinearLayer1	64	LeakyReLU
LinearLayer2	64	LeakyReLU
LinearLayer3	2	Softmax

3.2.3 Network Training

The proposed CNN model is trained for 22 epochs. Adaptive Moment Estimation (Adam) optimizer [16] is a popular optimization technique for training deep networks. Adam optimizer is used during the training phase of the proposed CNN model. Both batch size and Adam optimizer learning rate are changed during the training phase if the training loss stops decreasing. Table 3.2 summarizes the parameter values used in the training phase of the proposed CNN model. Fig. 3.2(a) shows the progress for training and validation loss across each epoch. The difference between the training loss and validation loss through epochs shows that we did not memorize the dataset.

TABLE 3.2: The change of batch size and learning rate through the Training process

Epoch Number	**Batch Size**	**Learning Rate**
From 0 to 6	128	1e-3
From 7 to 12	256	1e-3
From 13 to 21	256	1e-4

3.2.4 Model Evaluation

To assess the efficiency of the proposed method, the proposed method is compared to recent state-of-the-art methods for detecting Covid-19 cases. Experiments are conducted with the same dataset and the corresponding hyperparameter of each work. All the methods depend on CNN. The comparison is performed using precision, sensitivity, F1-score, and accuracy [17]. In addition, the number of parameters used in the training phase is a very important comparison factor. Table 3.3 depicts the comparison between state-of-the-art methods and the proposed method. As shown in the comparison, the proposed method outperforms other methods achieving the maximum accuracy and the lowest parameter count.

TABLE 3.3: A performance comparison between the proposed method and state-of-the-art models.

Method	**PC**	**P(%)**	**S(%)**	**F1(%)**	**A(%)**
Proposed Method	0.15M	100.00	100.00	100.00	100.00
ResNet-34 [18]	21.8M	96.77	100.00	98.36	98.33
ACoS Phase I [19]	-	98.266	96.512	98.551	98.062
ResNet-50 [18]	25.6M	95.24	100.00	97.56	97.50
GoogleNet [18]	5M	96.67	96.67	96.67	96.67
VGG-16 [18]	138M	95.08	96.67	95.87	95.83
AlexNet [18]	60M	96.72	98.33	97.52	97.50
MobileNet-V2 [18]	3.4M	98.24	93.33	95.73	95.83
Inception-V3 [18]	24M	96.36	88.33	92.17	92.50
SqueezeNet [18]	1.25M	98.27	95.00	96.61	96.67

PC is Parameter count, P is precision, S is sensitivity

F1 is F1-score, and A is accuracy

3.3 Evaluation of the Methodology II

Methodology II is evaluated and compared against strong baselines and related works.

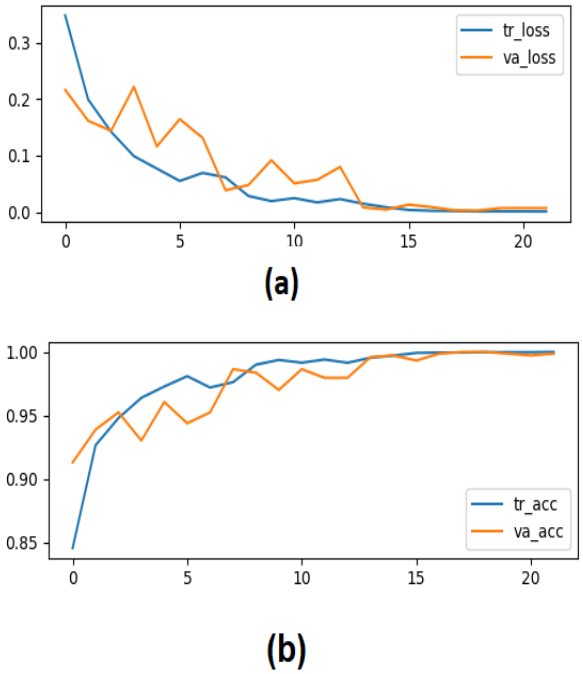

FIGURE 3.2: (a) The training loss and the validation loss of each epoch and (b) The training accuracy and the validation accuracy of each epoch.

3.3.1 Baseline Networks

Different architectures are trained to validate the effectiveness of the proposed method.

Spatial Pyramid Pooling (SPP-net) Based model

Four variants of SPP-net[13] is trained. All 4 variants have the same architecture but different SPP-layer. These variants of SPP-layer are as follows:

- full pyramid SPP of 8-levels using average-pooling as an aggregation function
- full SPP pyramid of 8-levels using max-pooling as an aggregation function
- single level SPP with 10-bins using average-pooling as an aggregation function
- single level SPP with 10-bins using max-pooling as an aggregation function

A Fixed Architecture is used for all SPP variant models with the same design principles of the proposed architecture. These architectures are the same

as the proposed architecture but DSWASPP is replaced by DC6 and SPP-1 layer is replaced with the corresponding SPP layer. DC6 is defined as six convolutional layers Densely connected. For SPP-net variants training a multiscale augmentation is added to the proposed augmentation process. Multi-scale augmentation is done by randomly sampling different 5 scales typically $\{320, 320 \pm 25, 320 \pm 50\}$.

Switchable Atrous Spatial Pyramid Pooling (SASPP-net) Based models

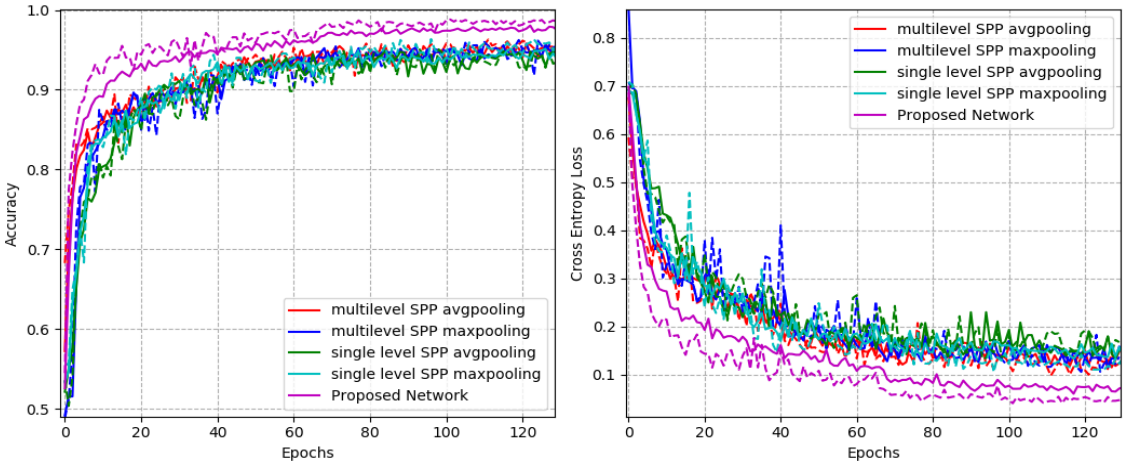

FIGURE 3.3: Training profiles of both SPP-net variants and the proposed network. For the same color solid line represents training statistics while the dashed line represents the validation statistics for the corresponding model. **left:** is the training accuracy. **Right:** is the training loss.

Another Baseline is introduced for comparison which is exactly as same as the proposed network but with a different Attention module structure and does not include a bottleneck within ASPP. This architecture is referred to as Switchable Atrous Spatial Pyramid Pooling (SASPP-net). The attention module structure is $Softmax(FC(GAP(X)))$ where: *X: is the input feature map, GAP: is a global average pooling, FC: is fully Connected layer performs a non-linear projection to $\mathbb{R}^4$ 4 values for the three scales and the input feature map.*

Table 3.4 summarizes the baseline models and the Corresponding parameter count.

TABLE 3.4: Baselines and their total number of parameters

Model Type	**Baseline CNN Architectures**	
	Variant	***Param. Count***
SPP	ML Average pooling	$14,916,420$
	ML max pooling	$14,916,420$
	SL Average pooling	$14,490,436$
	SL max pooling	$14,490,436$
SASPP		$13,031,841$

ML: Multilevel, **SL**: Single level

3.3.2 Models Training

The proposed architecture and baseline architecture are trained with the same hyperparameters. The dataset is split into 0.6, 0.2, and 0.2 for training, validation, and testing, respectively. For training a Cross Entropy Loss is used. All models trained with ADAM [16] optimizer with learning rate start by 10^{-3} and reduced every time validation loss plateau by multiplying by 10^{-1}. A Max Norm Constraint is used to clip the gradient value to the norm of 1 [9]. A batch size of 128 is used to calculate the gradient.

3.3.3 Reducing the overfitting

Overfitting is a critical problem for training large networks [9]. The proposed work has reduced the overfitting by using:

- Using Dropout with retrain probability of 0.5 [14].
- Using BatchNorm adds noise due to randomization introduced when constructing the minibatch [2].
- Using max norm constraint [9].
- Deep and thin architectures by design have an implicit regularization effect [8].
- Augmentation process i.e.) Texture augmentation [9].
- The use of small kernel size [20].

- Bottleneck in SWASPP module and the attention module.

During training no overfitting effects are observed.

3.3.4 Comparison with baselines

The proposed network is compared with the vanilla SPP-based Architecture and ASPP architecture.

Comparing with SPP-nets

Fig. 3.3 illustrates both training loss and training and validation accuracies and losses. Table 3.5 illustrates the testing accuracy for comparison between the SPP-nets baseline and the proposed architecture.

TABLE 3.5: Comparison between Proposed network and baseline SPP architectures

Model Name	Accuracy
SPP ML Average pooling	0.958
SPP ML max pooling	0.950
SPP SL Average pooling	0.927
SPP SL max pooling	0.957
Proposed Network	0.987

ML: Multilevel, **SL**: Single level

Comparing with SASPP

Fig. 3.4 illustrates the training and validation loss of training a SASPP baseline architecture. As shown in Fig. 3.4 SASPP is unable to generalize and start overfitting the training set. This comparison empirically shows the importance of the bottleneck introduced in the proposed architecture.

3.3.5 Comparing with the related works

To fairly compare with the related works proposed work is further trained. Fig. 3.5 shows the training and validation loss of the proposed network. Fig. 3.6 shows the of the training and validation accuracy.

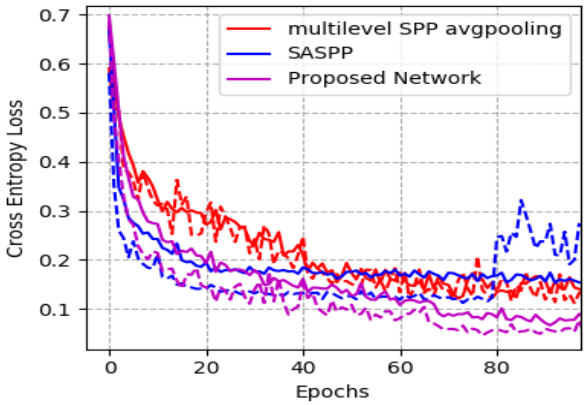

FIGURE 3.4: SASPP baseline architecture loss during both training, solid line, and validation, bashed line, compared with Proposed network and best performing SPP architecture.

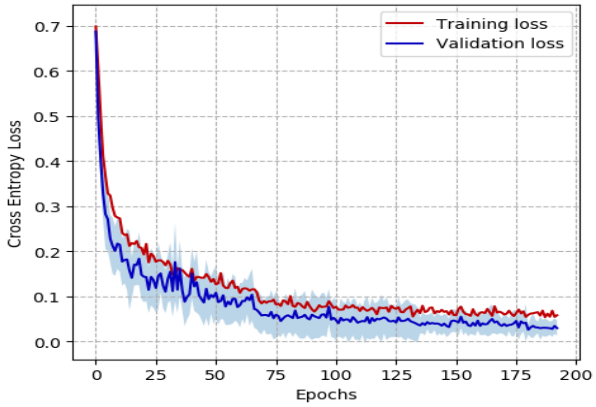

FIGURE 3.5: Cross entropy loss of the proposed architecture.

The proposed Network has a sensitivity, recall, and precision of 0.994 and 0.991 respectively on the validation set. Precision can be improved by investigating the precision-recall trade-off. Fig. 3.7 shows the trade-off between precision and recall for different thresholds. A threshold of 0.618 is used to improve the precision resulting in a sensitivity, recall, of 0.9903 and precision of 0.9956. Comparison metrics are defined as follows:

- *Accuracy*: is the ratio of correctly classified samples to the total number of samples
- *Sensitivity*: is the ratio of correctly classified Covid-19 samples to the total number of actual Covid-19 samples

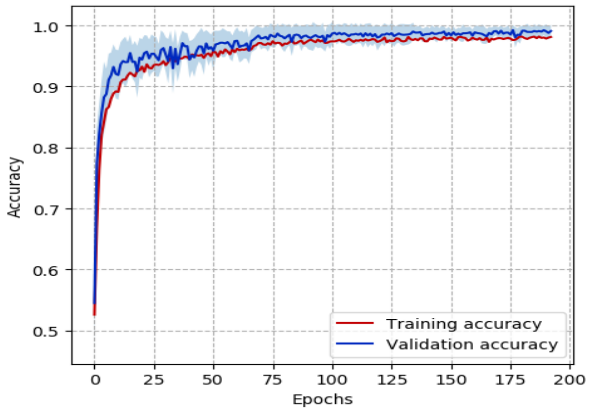

FIGURE 3.6: Training and validation accuracy of the proposed architecture.

TABLE 3.6: Comparison between Proposed network and Related works

Model Name	Accuracy	Sensitivity	Precision	Specificity	F1-score	Param. Count
Proposed	0.99294	**0.9903**	**0.9956**	0.9956	**0.9929**	**5,040,571**
SRC-Dalm[21]	0.985	0.886	-	0.993	-	-
SRC-Hom[21]	0.977	0.921	-	0.982	-	-
CRC-light[21]	0.973	0.955	-	0.974	-	-
DenseNet121*[21]	0.992	0.9714	-	0.9949	-	6,955,906
Inception-v3[21]	0.993	0.954	-	0.998	-	21,772,450
Modified MobileNetV2 [22]	0.98	0.98	0.97	-	0.97	-
ReCovNet-v2[23]	0.99726	0.98571	0.94262	0.9977	0.96369	-
ReCovNet-v1[23]	0.99824	0.9781	0.97438	0.99901	0.97624	-
DenseNet-121[23]	**0.9988**	0.97429	0.9932	**0.99974**	0.98365	6,955,906

- *Precision*: is the ratio of correctly, according to the ground-truth labels, classified Covid-19 samples to the total number of samples classified as Covid-19.
- *Specificity*: is the ratio of correctly, according to the ground-truth labels, classified non-COVID-19 to the total number of non-COVID-19.
- *F1-score*: is the harmonic mean of both Sensitivity and Precision.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

- *Param. Count*: is the total number of the trainable parameters.

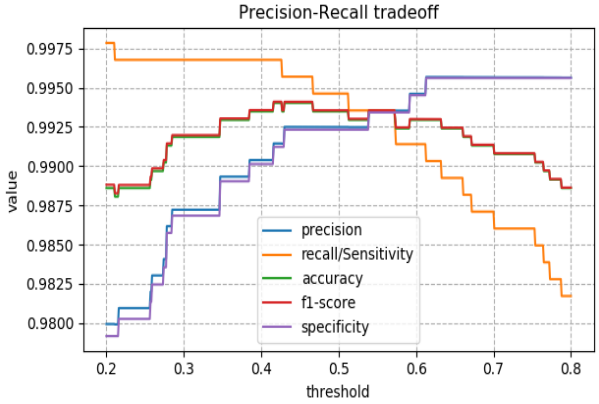

FIGURE 3.7: precision-recall trade-off of the proposed network.

Table 3.6 summarizes the comparison between the recent related works and the proposed architecture. The proposed architecture outperforms these works in many metrics. As their training and testing do not depend on a balanced number of samples, accuracy, and specificity are not good metrics for evaluation.

3.4 Summary

This chapter illustrates the superior performance of the proposed work I and II. In addition to the superior performance of the proposed works I and II, the chapter also provides a detailed analysis of the experimental results. The evaluation of proposed work I demonstrates that the use of spatially separable kernels and residual connection significantly improves the performance of the COVID-19 detection system. The batch normalization technique is found to be effective in maintaining network stability during the training process. The dynamic selection of hyperparameters such as batch size and learning rate further improves the performance of the proposed architecture I.

Furthermore, the proposed architecture I outperforms existing works for binary classification of chest X-ray images for normal and COVID-19 cases. This is particularly noteworthy as the proposed architecture has a low parameter count of only 150K trainable parameters compared to previous works. The achievement of a performance of 100% for accuracy, sensitivity, precision, and F1-score indicates the high accuracy and reliability of the proposed architecture.

Although the proposed architecture I performs exceptionally well, it does not address the scale variant nature of the CNN model. This issue is addressed in the proposed work II, which incorporates multiscale training approaches to obtain better quantitative results for CXR COVID-19 classification. The use of Atrous Spatial pyramid pooling enables the internal production of multiscale

feature maps, which are then fused using an attention module. To achieve a compact representation, a bottleneck dimension is introduced in both the multiscale feature extractor module and the attention module.

The proposed work II outperforms the current state-of-the-art architecture with lower parameter numbers. The recorded 0.9929 for the F1-score indicates the high performance of the proposed architecture. The detailed analysis and experimental results presented in this chapter provide valuable insights into the effectiveness of the proposed architectures and their potential for improving the accuracy and reliability of COVID-19 detection systems.

Chapter 4

Conclusion And Future Work

The COVID-19 pandemic has caused severe respiratory tract infections that have rapidly spread through contact with infected individuals, resulting in devastating loss of life and economic damage worldwide. The high rate of transmission has put tremendous pressure on healthcare systems to develop fast and accurate methods for diagnosing the disease. Convolutional Neural Networks (CNNs) have shown success in various computer vision tasks, but they are scale-variant and computationally expensive. In this thesis, we proposed novel architectures for multiscale feature extraction and classification, as well as a lightweight architecture for COVID-19 diagnosis.

The proposed lightweight CNN model, referred to as CNN-I, exploits spatial kernel separability to significantly reduce the number of training parameters and regularizes the model to only learn linear kernels. To maintain network stability and reduce overfitting, residual connections, and batch normalization are extensively used. We trained this lightweight architecture on the QaTa-Cov19 benchmark dataset, achieving 100% accuracy, sensitivity, precision, and F1-score with a parameter count of only 150K, which is significantly lower than other methods in the literature. As future work, attention and context attention can be explored to further enhance performance, and evaluating atrous convolution in the context of spatial separability may be beneficial.

Our second proposed architecture, CNN-II, learns multiscale features using a pyramid of shared convolution kernels with different atrous rates, making it scale-invariant. An attention-based mechanism is used to guide and select the correct scale for each input. CNN-II is an end-to-end trainable network that exploits a novel augmentation technique, Texture Augmentation, to reduce overfitting. This architecture achieved an F1-score of 0.9929 when tested on the QaTa-Cov19 benchmark dataset, with a total of 5,040,571 trainable parameters. We suggest that the SWASPP (Spatial Pyramid Atrous Spatial Pyramid

Pooling) can show great performance for segmentation, especially atrous convolution originating in the segmentation literature. Additionally, this work can be extended to classify various types of pneumonia.

In conclusion, this thesis proposes novel architectures for COVID-19 diagnosis that address the limitations of traditional CNN models. These architectures achieved high accuracy while reducing computational cost and parameter count. Further research can explore attention mechanisms and evaluate the use of atrous convolution in the context of spatial separability to improve performance. This work has the potential to improve COVID-19 diagnosis and aid in the development of fast and effective methods to combat future pandemics.