University Name

Doctoral Thesis

Thesis Title

Author:
John Smith

Supervisor:
Dr. James Smith

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

January 31, 2023

Abstract

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

List of Abbreviations

For/Dedicated to/To my...

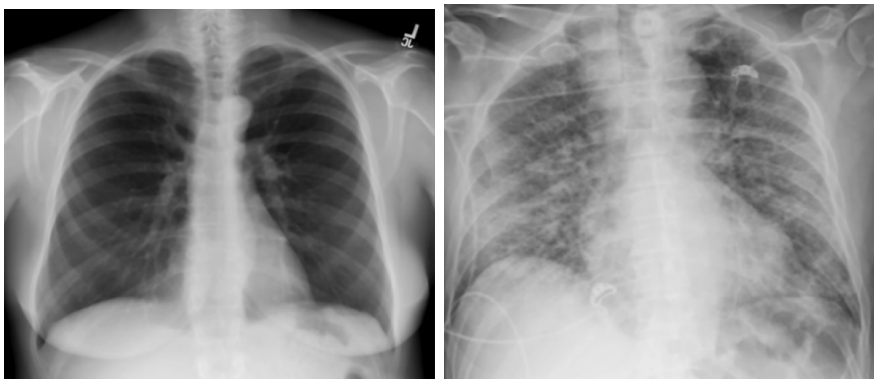
Chapter 1

Introduction

1.1 Novel Coronavirus Disease

COronaVIrus Disease 2019 (COVID-19) is a severe respiratory tract infections which can range from mild to lethal [1]. COVID-19 is contagious disease that can readily spread through direct or indirect contact with an infected person [2]. COVID-19 caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). COVID-19 initially appeared in Wuhan, China late 2019 and spread world wide [3]. World Health Organization (WHO) declared the outbreak of COVID-19 as Public Health Emergency of International Concern on January 2020, and a pandemic on March 2020 [4]. Ongoing SARS-CoV-2 infections have not only devastated human lives but also significantly damaged the financial health of both developing and developed countries. Therefore, there is an urgent need to control the pandemic by accelerating the development and mass production of efficacious vaccines against SARS-CoV-2. Healthcare practitioners, researchers, and policymakers around the globe were thrown a challenge to deliver adequate prevention and treatment modalities to combat the pandemic. From the initial stage of this pandemic, scientists were focused on either repurposing the existing drugs or developing vaccines against COVID-19 [5]. Fig. 1.1 illustrates the difference between the COVID-19 and normal lungs.

Typical diagnostic tools for COVID-19 are virus' nucleic acid by real-time reverse transcription polymerase chain reaction (rRT-PCR), transcription-mediated amplification (TMA), or by reverse transcription loop-mediated isothermal amplification (RT-LAMP) from a nasopharyngeal swab [6]. These manual traditional methods are time-consuming and complex. Chest X-rays (CXR) and chest CT offer fast screening methods for COVID-19 [7][8][9]. CXR and chest CT are preferred when RT-PCR testing is not available in time [10]. CXR



(A) Chest X-Ray image of normal lung. (B) Chest X-Ray image of COVID-19 patient

FIGURE 1.1: Samples of CXR images of Normal and COVID-19 pneumonia

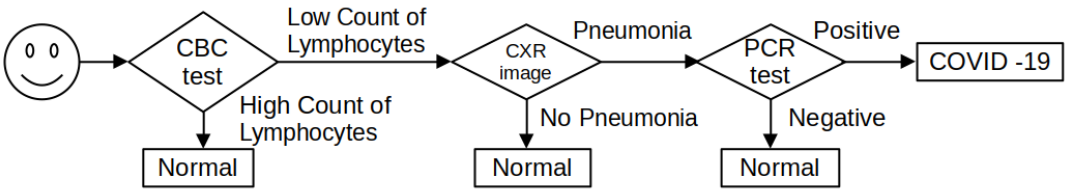


FIGURE 1.2: Typical hospitals routine of reducing COVID-19 spread

has many advantages over chest CT including the widespread of the acquisition devices, low cost, and the speed of the acquisition[11][12][13][14]. These advantages lead CXR to be a fixed routine for hindering COVID-19 spread.

1.2 Motivation

Fig. 1.2 illustrates typical Egyptian hospitals routine for reducing COVID-19 spread among healthcare workers. This process is performed for every visitor of the hospital. Phases of CBC Test and PCR test are automated and does not consume time. While CXR image diagnosing involves human factor which act a bottleneck of the process. This process can be fully automated using image classification models. As a consequence *1)* Total time required for every visitor will be reduced. *2)* Work load of radiologist is reduced.

Deep learning [15] models, namely convolutional Neural Network (CNN) [16], have shown a great performance in computer vision problems such as object detection [17][18][19][20] and object recognition [21][22]. CNN is initially introduced in [16]. CNN is based on hierarchical learning of the convolutional kernels which organized in layers [23]. The function of lower order layers is

to learn low level features such as edges and corner points [24]. Higher order layers learn a high level features such as objects [24]. Typical CNN architecture is a number of convolutional layers that is connected sequentially [21]. This sequential connection of the layers does not scale well for deep CNN [22]. CNN shows a great performance when it has large number of layers [22]. To allow CNN to take the advantage of deep architecture, residual connections are used [22]. Residual connections solve the vanishing gradient problem when gradients approach zeros. Also, residual connections allow the reuse of earlier features [25]. Layers with residual connections are easier to optimize than plain layers as it can easily approximate the identity mapping [22]. In this thesis CNN is used to automated and accelerate the diagnosing pipeline of the COVID-19.

1.3 Detection Challenges

Like many computer vision problem COVID-19 has the following challenges

- **Availability of Data.** CNN is a deep learning technique that require large number of a training Images. However, current COVID-19 dataset are small-size dataset which is a challenging to train large networks with.
- **Scale.** Like many computer vision models CNN is scale variant. CNN can not recognize images with scales different from the scales in it trained on.
- **Vanishing Gradient Problem.** Deep CNN suffer vanishing gradient which prevent CNN parameters to be updated.
- **Exploding Gradient Problem.** Deep CNN suffer Exploding gradient which make CNN diverge.

Proposed Systems try to overcome these problems as will be detailed in chapters 4 and 5.

1.4 Thesis Objective and Contributions

Objective of this thesis is to improve classification accuracy and reduce computational complexity of detecting COVID-19 cases from CXR Images. Contributions of this thesis as follows.

- *Lightweight classification model.* Lightweight classification model with very low parameter number is proposed for classification the CXR images. It reduces the computational complexity which allows deployment

on mobile devices and saving bandwidth of the network during model distribution process.

- *Scale invariant model.* Pneumonia scales in CXR are not uniform distributed which prevent CNN from recognizing infrequent scales. Scale invariant model is proposed for detection COVID-19 pneumonia at different scales.
- *High Detection accuracy.* Proposed systems provide superior performance according to various classification metrics.

1.5 Thesis Organization

This thesis is organized as follows:

- **Chapter 2**: includes required Background to understand the Thesis.
- **Chapter 3**: includes and illustrates the recent and related work in the COVID-19 detection literature.
- **Chapter 4**: presents the proposed work I which presents a lightweight classification model.
- **Chapter 5**: presents the proposed work II which includes the scale invariant model for COVID-19 classification.
- **Chapter 6**: illustrates the experimental results for both proposed work I and II and quantitative analysis of the proposed work I and II is provided.
- **Chapter 7**: concludes the thesis and provide planning for the future work as extension of the proposed approach

Chapter 2

Background

This chapter includes required background to understand the thesis proposal presented in the following chapters.

2.1 Convolutional Neural Networks (CNN)

CNN is initially introduced by LeCun [16] which is based on learning adaptive convolutional kernels. CNN consist of two parts convbase and densebase parts. For the Convbase instead of connecting all of the units in a layer to all the units in a preceding layer, convolutional networks organize each layer into feature maps [16], which can be though of as parallel planes or channels. In a convolutional layer, the weighted sums are only performed within a small local window *i.e)* receptive field, and weights are identical for all pixels, just as in regular shift-invariant image convolution and correlation. This parameter sharing reduce the required total number of parameter and allows learning shift invariant convolutional kernels. These convolutional kernels produce equivariant features maps. Fig. 2.1 represents typical CNN architecture of LeNet.

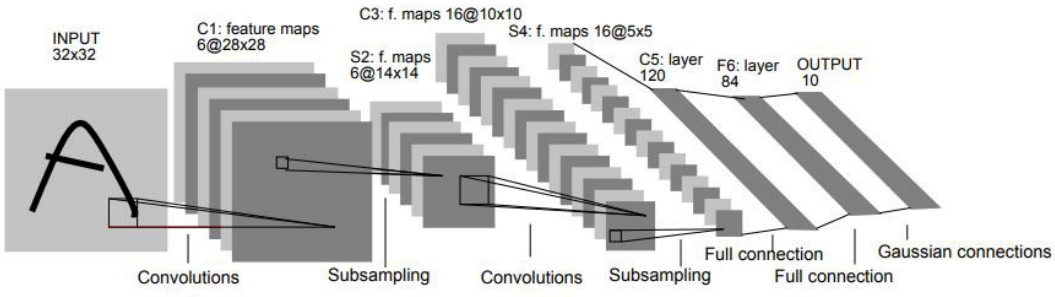


FIGURE 2.1: Typical CNN architecture

2.1.1 Convolutional Layer

The building block of the convolutional layer is the 2D convolutional kernel. Fig. 2.2 illustrates the 2D convolutional layer. Each 2D convolution kernel takes as input all of the C_{i-1} channels in the preceding layer, windowed to a small area, and produces the values in one of the C_i channels in the next layer. For each of the output channels, we have $K^2 \times C_{i-1}$ kernel weights, so the total number of learnable parameters in each convolutional layer is $K^2 \times C_{i-1} \times C_i$. In Fig. 2.2, we have $C_{i-1} = 6$ input channels and $C_i = 4$ output channels, with an $K = 3$ convolution window, for a total of $9 \times 6 \times 4$ learnable weights, shown in the middle column of the figure. Since the convolution is applied at each of the $W \times H$ pixels in a given layer, the amount of computation (multiply-adds) in each forward and backward pass over one sample in a given layer is $W \times H \times K^2 \times C_{i-1} \times C_i$. To fully determine the behavior of a convolutional layer, we still need to specify the following hyperparameter:

- **Padding.** Padding is used to preserve the spatial dimension of the input feature map after the convolution operation is performed. Typically, it is performed by inserting $\lfloor K/2 \rfloor$ columns for both sides and $\lfloor K/2 \rfloor$ rows to the top and bottom.

- **Stride.** Stride is the step taken between two centers when performing the convolution operation. Typically, Stride is equal to 1. Stride can act as down sampling operation that can be performed instead of the pooling operation.

- **Dilation.** Dilation is a technique that enlarge the kernel by inserting zeros between its consecutive elements. As a result it covers a larger area of the input without increasing the total number of parameters. Which can be discribed as

$$y[i] = \sum_{k}^{K} x[i + r \times k] w[k]$$

  where x is the input signal, w is the convolutional filter with size of K, y is the resultant signal, and r is the dilation rate.

Generally, each convolutional layer computes some activation based on its input and a nonlinear function. Activation function used in each layer have a great effect on the modeling of the problem and also the semantic assigned to the output of some units is affected by this choice. The most important activation functions will be introduced in the following section.

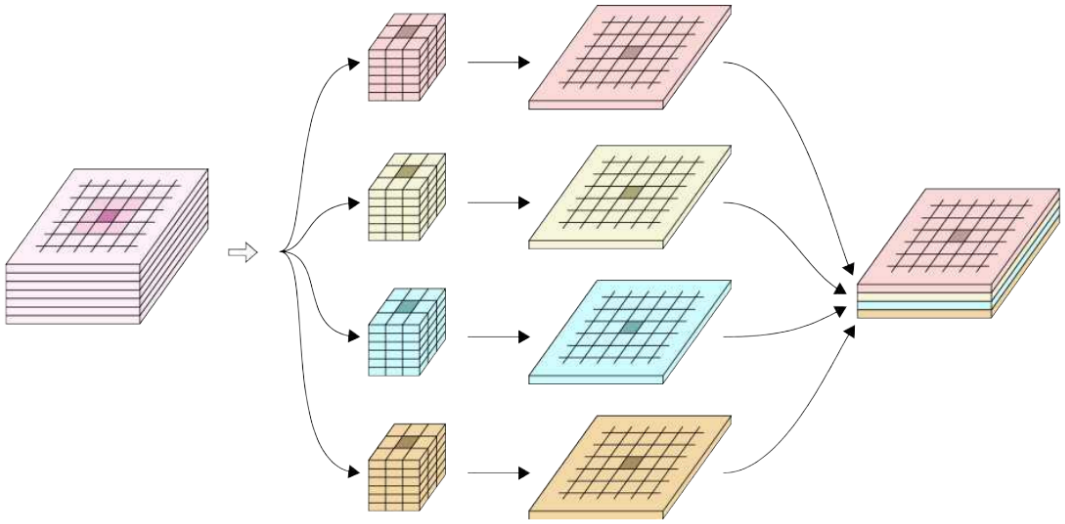


FIGURE 2.2: Convolutional layer with single input feature map and four convolutional kernels

2.2 Activation functions

The activation function is one of the most important component of an CNN. To tackle non-linearly separable problems it is imperative to map the input into a space that is linearly separable. The activation function does this by performing an *element-wise nonlinear transformation* of the pre-activation that comes from the linear combination of the convolutional layer.

The linear combination of the convolutional layer and the nonlinearity work together closely: the latter is usually fixed and does not evolve during training, but maps its input to a highly non-linear space; the former, is determined by the learned weights which is learned during the training process and uses the activation function to map the calculated activation into a new space where they are simpler and easier to separate. It is interesting to point out that N consecutive linear combination is a single linear combination. Activation function breaks this property and low introducing deeper networks.

In the following subsection common activation function is presented.

2.2.1 Sigmoid

The sigmoid, often called *logistic*, is a differentiable monotonically increasing function that takes any real-valued number and maps it to $[0, 1]$. As illustrates in its representation in Fig. 2.3, for large negative numbers it approaches 0. However, for large positive numbers it approaches to 1. It is defined as

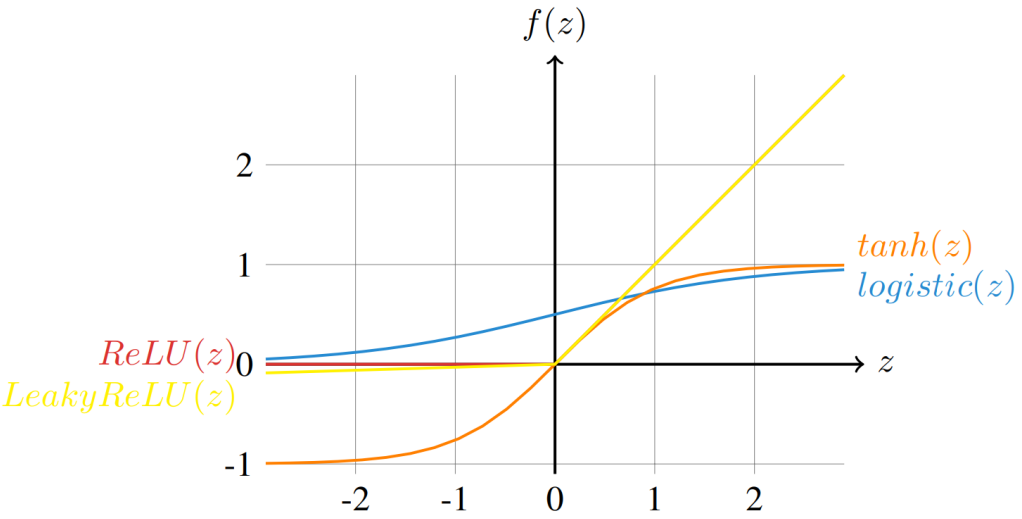


FIGURE 2.3: Some of the most common activation functions: sigmoid, tanh, ReLU and Leaky ReLU. ReLU and Leaky ReLU are overlapping for $z \geq 0$.

$$logistic(\mathbf{z}) = \frac{1}{1+\exp(-\mathbf{z})}. \tag{2.1}$$

The logistic function is the most used nonlinearity historically due to its possible interpretation as the firing probability of a neuron given its activation: when the activation is low the neuron fires less often whereas when the activation is high the frequency of the spikes increases.

Another very important property of the logistic function is that it is a very simple and fast derivative computation such as follows:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{z}} logistic(\mathbf{z}) &= \frac{\exp(-\mathbf{z})}{(1+\exp(-\mathbf{z}))^2}, \\
&= \frac{1}{1+\exp(-\mathbf{z})} \cdot \frac{\exp(-\mathbf{z})}{1+\exp(-\mathbf{z})}, \\
&= logistic(\mathbf{z}) \cdot \frac{\exp(-\mathbf{z})}{1+\exp(-\mathbf{z})}, \\
&= logistic(\mathbf{z}) \cdot \frac{1+\exp(-\mathbf{z})-1}{1+\exp(-\mathbf{z})}, \\
&= logistic(\mathbf{z}) \cdot \left(1 - \frac{1}{1+\exp(-\mathbf{z})}\right), \\
&= logistic(\mathbf{z}) \cdot (1 - logistic(\mathbf{z})).
\end{aligned} \tag{2.2}$$

It becomes out of favor due to the following major drawbacks :

- *Saturation region which weaken the gradient propagation:* backpropagation algorithm exploits the gradient of the loss function to update the parameters of the network. However, sigmoid have a derivative of zero at both of ends which saturates the training. This problem – often referred to as *vanishing gradient problem* – makes training very slow or prevents it in some cases. As a result sigmoid requires a very careful initialization of the weights of the network.

- *The output is not zero-centered:* according to [26] normalized activation (i.e., activation with zeros mean and unit variance) can accelerate the training. However, sigmoid always has non-negative activation as result non zero-center mean which slow down the training.

2.2.2 Hyperbolic tangent (tanh)

The hyperbolic tangent, *tanh*, is a differentiable monotonically increasing function that maps any real-valued number to $[-1,1]$. This nonlinearity has the same problems as sigmoid except its activation is zeros-center.

$$tanh(\mathbf{z}) = \frac{1 - exp(-2\mathbf{z})}{1 + exp(-2\mathbf{z})}. \tag{2.3}$$

2.2.3 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) is introduced in [23].It has become the nonlinearity of choice in many applications [23][22]. It is defined as

$$relu(\mathbf{z}) = max(0, \mathbf{z}). \tag{2.4}$$

Although very simple, it has some very interesting properties[27] as follows:

- *No positive saturation:* the ReLU does not response, or saturate, for non-positive inputs, but does not otherwise. This ensures a flow of gradient, update signal, whenever the input is non-negative, that was found to significantly speed up the convergence of training.

- *Cheap to compute:* unlike many other activation functions which requires expensive computation, such as exponential function, ReLU's implementation simply a threshold at zero. Another important characteristic is

that the gradient is trivial to compute:

$$\nabla(relu(\mathbf{z}^{(l)})) = \begin{cases} \mathbf{a}^{(l-1)}, & \text{if } \mathbf{z}^{(l)} > 0, \\ 0, & \text{if } \mathbf{z}^{(l)} < 0, \\ undefined, & \text{if } \mathbf{z}^{(l)} = 0. \end{cases} \tag{2.5}$$

- *Induce sparsity:* ReLU units induce sparsity, whenever the input is negative their activation is zero. Sparsity is a desired property: as opposed to dense encoding, sparsity will produce representations where only a few entries change upon small variations of the input, i.e., it will produce a representation that is more consistent and robust to perturbations. Furthermore, sparsity allows compact encoding, which is desirable in many contexts such as, e.g., data compression and efficient data transfer. Finally, it is also usually easier to linearly separate sparse representations [28].
- *ReLU units can die:* ReLU does not restrict the gradient flow from the positive part. Large gradient can update the weight in a such way it can not activate again, i.e, it always produces a negative value. This problem can be partially solved with some ReLU variant such as leaky ReLU or a parametric ReLU.

2.2.4 Leaky Rectified Linear Unit (Leaky ReLU)

Leaky ReLUs have been proposed as a way to mitigate the saturated units of ReLUs caused by extreme update, by preventing the unit from have zero gradient thus allowing a gradient to flow through the unit, potentially recovering extreme values of the weights. Leaky ReLUs are widely adapted and defined as follows:

$$leaky\_relu(\mathbf{z}) = max(\beta * \mathbf{z}, \mathbf{z}), \tag{2.6}$$

where β is a small constant.

2.2.5 Softmax

Unlike previously mentioned functions softmax differs in that it does depend in all the values of the dimensions altogether to produce the categorical distribution of the over N classes. It defined as follows:

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_{k=0}^{K} \exp(z_k)}, \tag{2.7}$$

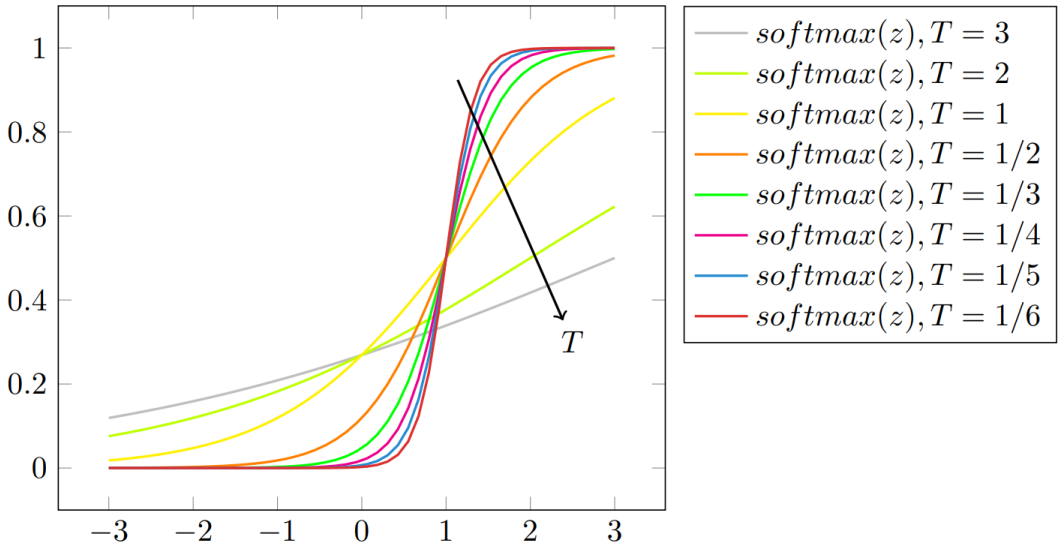


FIGURE 2.4: The steepness of softmax function as temperature T grows.

where K is the number of classes, i.e., of dimensions (or neurons).

Temperature parameter T can be used with the softmax which controls its steepness (see Fig. 2.4), i.e., to manage the randomness of predictions. High temperature, case of $T = \inf$, produces a uniform categorical distribution. While small temperature produces peaked probability distribution for the larger value.

$$softmax(z_i) = \frac{\exp(z_i/T)}{\sum_{k=0}^{K} \exp(z_k/T)}. \tag{2.8}$$

2.3 Pooling

In addition to convolutional , *pooling* operations is an important building block in CNNs. Pooling operations reduce the spatial size of feature maps by using some aggregation, i.e) max or average, function to summarize a particular region. The following properties affect the output size o_j of a pooling layer along axis j:

- i_j: input size along axis j,
- k_j: pooling window size along axis j,
- s_j: stride (distance between two consecutive positions of the pooling window) along axis j.

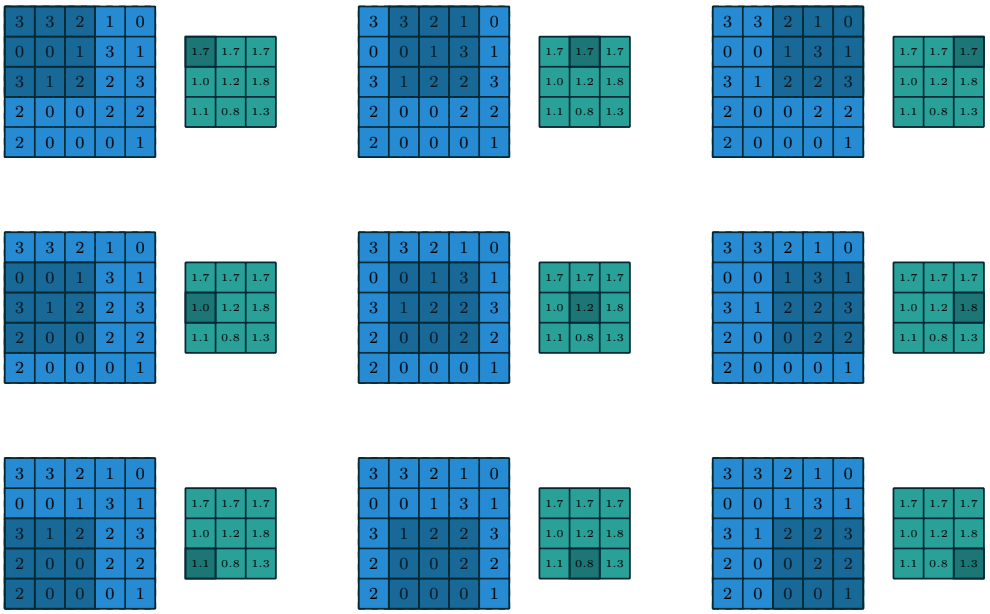


FIGURE 2.5: Computing the output values of a 3×3 average pooling operation on a 5×5 input using 1×1 strides.

2.3.1 Average Pooling

Average pooling is performed by sliding a window over the input feature map and performing and averaging the content of the window. Fig. 2.5 provides an example for average pooling.

2.3.2 Max pooling

Max pooling is performed same way as the average pooling performed, but instead of performing the averaging as an aggregation function it performs max function. Fig. 2.6 provides an example for average pooling.

CNN's are used for feature extraction from images followed by fully connected layers for classification. As convolution operation is applied in a sliding window fashion it can accept input of varied size, resulting in a varied size output. As CNN is followed by fully connected layers which can accept input of fixed size. This makes CNN incapable of accepting varied size inputs. Thus images are first reshaped into some specific dimension before feeding into CNN. This creates another issue of image warping and reduced resolution. Spatial Pyramid pooling comes as a counter to this problem.

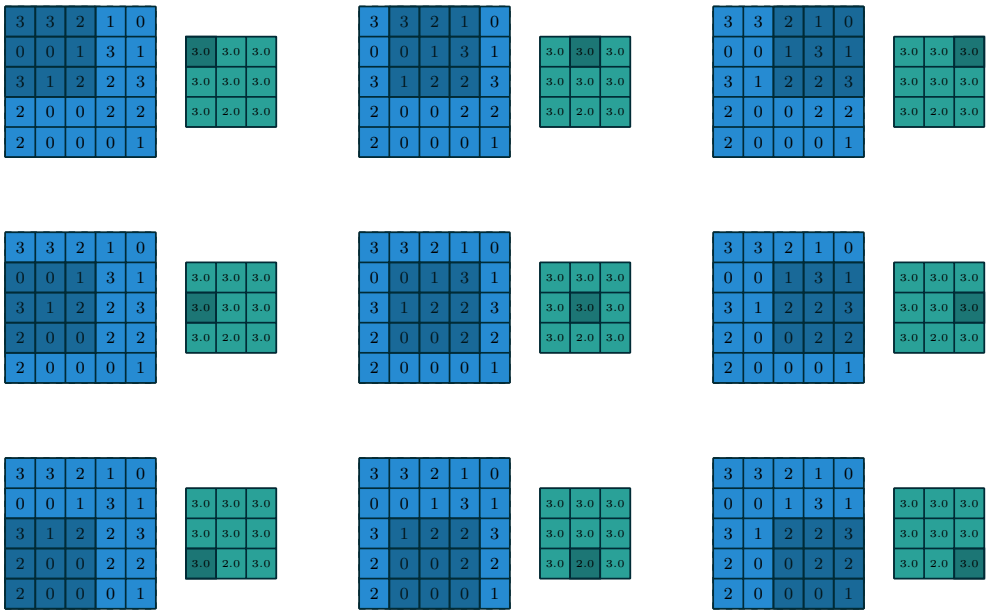


FIGURE 2.6: Computing the output values of a 3×3 max pooling operation on a 5×5 input using 1×1 strides.

2.4 Spatial Pyramid Pooling

The convolutional layers process arbitrary variable length, and also they produce outputs of variable sizes. Classifiers like SVM, decision tree or fully-connected layers require fixed-length input feature vector. Such vectors can be generated by the Bag-of-Words (BoW) approach [29] that pools the features together. Spatial pyramid pooling [30], [31] improves BoW in where it can preserve the spatial information of the input by pooling local spatial bins. These spatial bins have sizes that are proportional to the input image, so the number of bins is fixed regardless of the input dimensions. While sliding window pooling of the previous deep networks [23] the number of sliding windows depends on the input size. To adopt the deep network for images of arbitrary sizes, we replace the last pooling layer with a spatial pyramid pooling layer. Fig. 2.7 illustrates our method. In each spatial bin, we pool the responses of each filter. The outputs of the spatial pyramid pooling are kM-dimensional vectors with the number of bins denoted as M (k is the number of filters in the last convolutional layer). The fixed-dimensional vectors are the input to the fully-connected layer. With spatial pyramid pooling, the input image can be of any sizes.

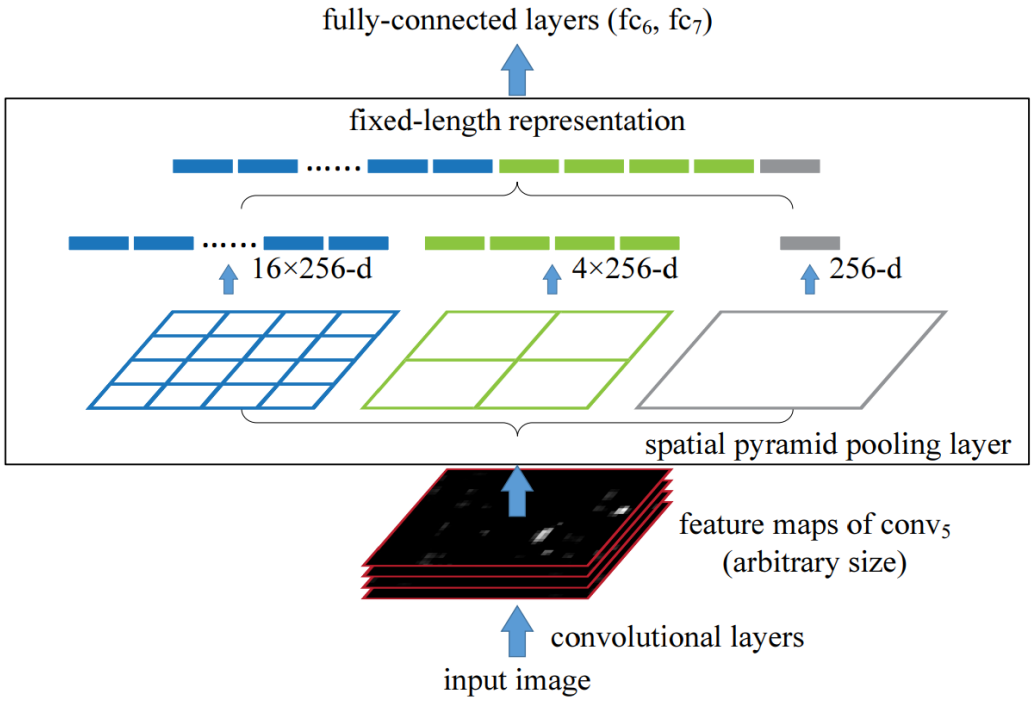


FIGURE 2.7: spatial pyramid pooling layer. Input feature map is divided to pins for each pin an aggregation function is performed

2.5 Neural Network Attention

In a vision system, an attention mechanism can be defined as a dynamic selection process that is realized by weighting features according to the importance of the input in an adaptive manner. Attention attempts to selectively concentrate on relevant information or features while ignoring other irrelevant ones. Currently, attention based models have shown a great success in natural language processing [32] and computer vision tasks [33]. Attention modules guide the network to focus on the most relevant features only [33]. Attention has many categories such as channel attention and spatial attention. Stated as follows

2.5.1 Channel Attention

Internal CNN feature maps have different channels in different usually each channel represents different objects [34]. Channel attention recalibrates the weight of each channel, and can be viewed as an object selection process, thus determining what to pay attention to. [35] is a pioneering channel attention

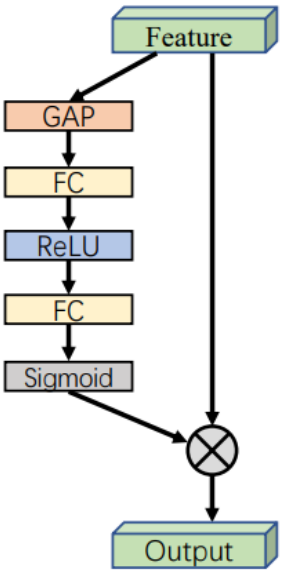


FIGURE 2.8: Channel attention block of SE network

mechanism which is formulated as follows

$$\begin{aligned} Att(X) &= \sigma(W_2 ReLU(W_1 GAP(X))) \\ Y &= Att(X)X \end{aligned} \tag{2.9}$$

where σ is sigmoid activation function, GAP is global average pooling. Fig. 2.8 illustrates the channel attention mechanism of [35]

2.5.2 Spatial Attention

Spatial attention can be considered as an adaptive spatial region selection mechanism: where to pay attention.

In [36] generates a spatial attention map by utilizing the spatial relationship of features. Unlike channel attention, the spatial attention focuses on where is an informative part, which is complementary to the channel attention. they compute the spatial attention by first applying average-pooling and max-pooling operations over each channel and concatenate them to generate the feature descriptor. After pooled-feature concatenation convolution layer is applied to generate a spatial attention map $\mathbf{M}_s(F) \in \mathcal{R}^{HW}$ which encodes the spatial importance. We aggregate channel information of a feature map by using two pooling operations, generating two 2D maps: $\mathbf{F}^s_{avg} \in \mathbb{R}^{1\times H\times W}$ and $\mathbf{F}^s_{max} \in \mathbb{R}^{1\times H\times W}$. Each denotes average-pooled features and max-pooled features across the channel. Those are then concatenated and convolved by a standard convolution layer, producing the 2D spatial attention map. In short,

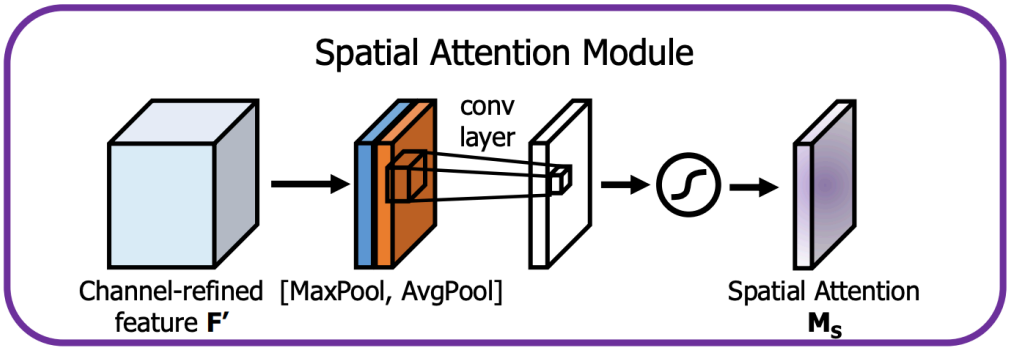


FIGURE 2.9: Spatial attention mechanism

the spatial attention is computed as:

$$\begin{aligned} &\mathbf{M}_s(F) = \sigma\left(CONV_{7\times7}\left(\left[\text{AvgPool}(F); \text{MaxPool}(F)\right]\right)\right) \\ &OR \\ &\mathbf{M}_s(F) = \sigma\left(CONV_{7\times7}\left(\left[\mathbf{F}^s_{avg}; \mathbf{F}^s_{max}\right]\right)\right) \end{aligned} \tag{2.10}$$

where σ denotes the sigmoid function, $CONV_{7\times7}$ represents a convolution operation with the filter size of 7×7, and $[\cdot]$ is a concatenation operator.

2.6 Normalization

Normalization is crucial for training a deep network. Normalization allows the use of larger learning rate and results in smoother loss landscape of the objective function. Different types are summarized as follows:

2.6.1 Batch Normalization

As proposed in [26] batch normalization is reduces *Internal Covariate Shift* (ICS). ICS is defined as the change in the distribution of network activations due to the change in network parameters during training. Batch Normalization (BN) mitigate the ICS by normalizing the internal feature maps, minibatch, to have zero mean and unit variance. BN also allows the CNN to undo the normalization by scaling and shifting the normalized features using γ, β.

BN is applied to each layer for a minibatch $\mathcal{B}$ as follows:

$$\begin{aligned}
\mu_{\mathcal{B}} &= \frac{1}{m}\sum_{i=1}^{m} x_i \\
\sigma_{\mathcal{B}}^2 &= \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \\
\hat{x}_i &= \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\
y_i &= \gamma \hat{x}_i + \beta = \mathrm{BN}_{\gamma,\beta}(x_i)
\end{aligned} \tag{2.11}$$

BN has many benefits which is summarized as follows:

- **Accelerate network training.** BN allows larger learning rate.
- **Regularization effect.** BN has regularization effect due to batch construction is done stochastically.
- **Mitigate the effect of saturating activation function.** Normalization performed by BN relocate the layer activation to linear regime of saturating activation function such as sigmoid.

2.6.2 Layer Normalization

Unlike batch normalization, Layer Normalization [37] directly estimates the normalization statistics from the summed inputs to the neurons within a hidden layer so the normalization does not introduce any new dependencies between training samples.

We compute the layer normalization statistics over all the hidden units in the same layer as follows:

$$\begin{aligned}
\mu^l &= \frac{1}{H}\sum_{i=1}^{H} a_i^l \\
\sigma^l &= \sqrt{\frac{1}{H}\sum_{i=1}^{H} \left(a_i^l - \mu^l\right)^2}
\end{aligned} \tag{2.12}$$

where H denotes the number of hidden units in a layer. Under layer normalization, all the hidden units in a layer share the same normalization terms μ and σ, but different training samples have different normalization terms. Unlike batch normalization, layer normalization does not impose any constraint on the size

FIGURE 2.10: Data augmentation using crop augmentation with different preserving degrees and the corresponding accuracy of recognizing certain classes using ResNet

of the mini-batch and it can be used in the pure online regime with batch size of 1 sample.

2.7 Augmentation

Data augmentation [38] is a technique that is used for enlarging the training set, based on different modifications, using label preserving transformation. Data augmentation not only helps to grow the dataset but it also increases the diversity of the dataset and introduce robustness to these transformations. When training machine learning models, data augmentation acts as a regularizer and helps to avoid overfitting.

Data augmentation techniques have been found useful in domains like NLP and computer vision. In computer vision, transformations like cropping, flipping, and rotation are used. Fig. 2.10 represents pre-class performance to the degree of augmentation.

2.8 Dropout

Dropout [39] is a regularization technique for neural networks that drops a unit (along with connections) at training time with a specified probability p (a common value is $p = 0.5$). At test time, all units are present, but with weights scaled by p (i.e. w becomes pw). Dropout has the following benefits.

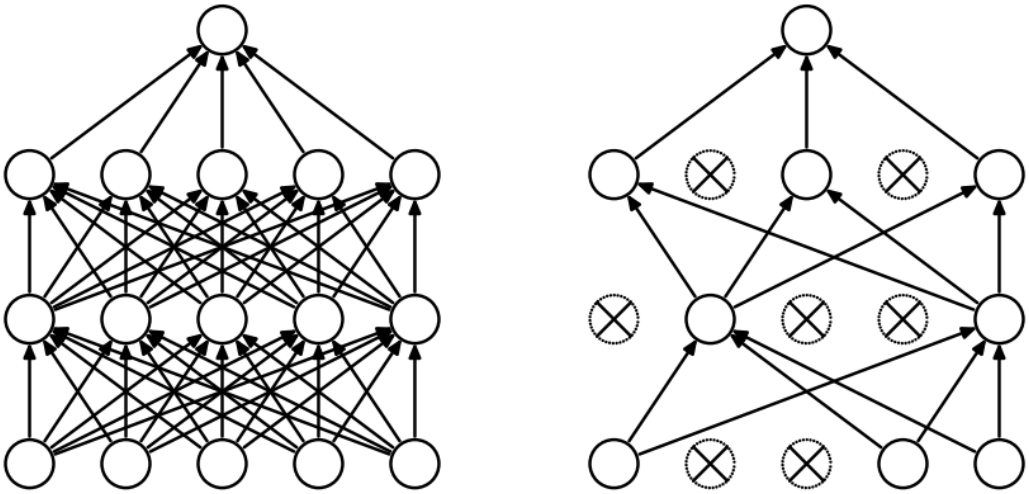


FIGURE 2.11: Dropout Neural network Model. **Left**: A standard network with two hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- **Reduces co-adaptation between the neurons.** Complex co-adaptation between neuron occurs when the neurons of the one layer depend on the neurons on next layers to correct their errors. Dropout fixes this issue by making neuron existance stochastic.
- **Implicit ensemble.** Dropout approximately combining exponential number different thinned neural network architectures efficiently

Fig. 2.11 illustrates the operation of the Dropout.

2.9 Initialization

2.10 Optimization

2.11 Summary

Chapter 3

Literature Review

Chapter 4

Proposed Methodology I

This chapter presents a lightweight architecture for COVID-19 detection which is based on spatial kernel separability and residual connection.

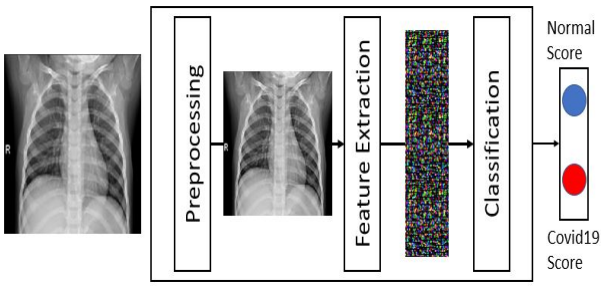


FIGURE 4.1: The phases of the proposed method I.

4.1 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 CNN model to classify the input chest X-Ray image to 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.4.1.

4.1.1 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 [16]. In addition, this phase leads the learning model of 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:

$$Y := \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{4.1}$$

where μ and σ is 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:

$$Z := w_1 Y + w_2 \tag{4.2}$$

where w_1 and w_2 are a trainable parameter.

Unlike the normalization method presented in [26], the batch normalization process presented in this paper has z-score normalization parameter that is used in both training and validation phases.

4.1.2 Feature Extraction and Classification

CNN models achieved an outstanding success in image recognition [15]. 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:

Separable CNN kernels

Kernel separability[40] [41] 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 the multiplication and the addition). A 2D kernel of $k \times k$ applied for 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

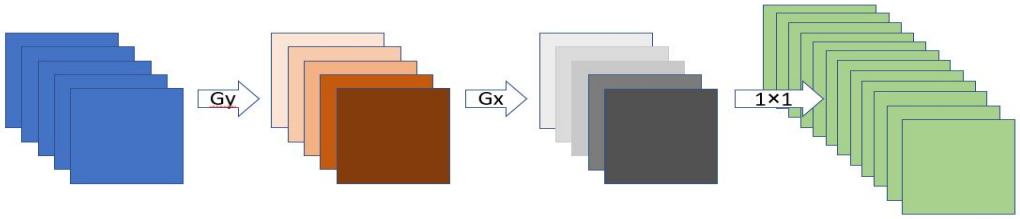


FIGURE 4.2: Separable convolution Gy and Gx have kernel size of $M \times 1$ and $1 \times M$. The combination of these kernels is approximately a $M \times M$ kernel and depth wise convolution are applied by a 1×1 convolution. The output depth is padded with zeros to have the same spatial size of Gy, Gx. Gy, Gx are performed channel wise.

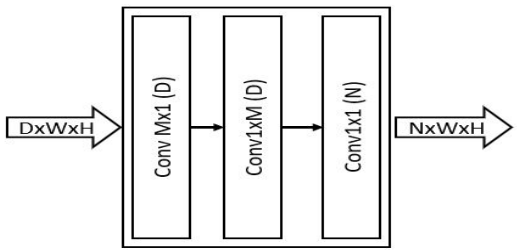


FIGURE 4.3: Separated Convolutional Layer composed of three consecutive layers. The first Convolutional layer has a kernel size of $(M \times 1)$ and D convolutional neuron. The second layer operates in the same way as the first layer but it has a kernel of size $(1 \times M)$ and D convolutional neuron. The third layer is the convolutional layer with kernel of size (1×1) and number of convolutional neuron is N.

$2(N-4)(M-4)k$ operations. The flow of separated convolution operations are summarized in Fig. 4.2. Fig. 4.3 represents the structure, denoted by 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 neuron and filters are equal to the number of channels as the input feature map and the convolution operations are performed in a 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 kernel of size (1×1) and number of convolutional neuron is N. The collaboration of the three layers are connected to preform similarly to the convolutional layer with kernel size of $(M \times M)$ and number of neuron and filter are the same as N but with large difference in the performance.

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) [27] is a nonlinear activation that allows the network to fit and approximate highly non-linear datasets distribution. The proposed method employs the batch normalization which is described in [26].

Batch Normalization [26] reduces internal covariate shift produced as a result of moving between layers during the feedforward procedure [26]. Batch Normalization makes the loss landscape smoother and reduces the number of saddle points [42] which allows to use higher learning rates. Using a higher learning rate makes the network training faster[26]. Batch normalization reduces the vanishing gradient problem and exploding gradient problem as it makes the resulted activation scale independent from the trainable parameter scale[26]. Batch normalization has the effect of regularization because of the inherited randomness when selecting the batch sample[26] 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 [22]. Training a deeper network is very expensive and has many challenges such as vanishing gradient problem, exploding gradient problem, and degradation problem [22]. Exploding gradient problem occurs when the gradient update becomes very large (approaching infinity) resulting in the network diversion. Vanishing gradient problem occurs when the gradient update becomes very small (approaching zero) resulting in preventing the parameter update for early layers[26] and preventing the network to learn new patterns. Batch normalization [26] and the use of ReLU activation function [23] 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 nonlinear functions. Residual connection[22] overcomes this problem by using skip connection as shown in Fig. 4.4. Fig. 4.4 represents the building block layer of the feature extraction phase, denoted by 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

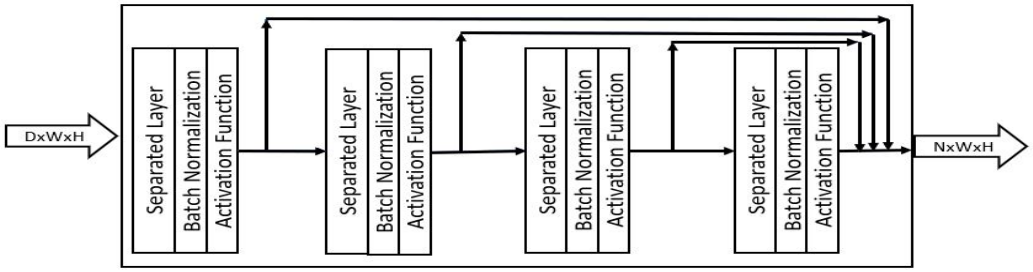


FIGURE 4.4: The stack of residual separated block (RSB) consists of four layer of separated convolutional layer each of which is followed by batch normalization and activation function.

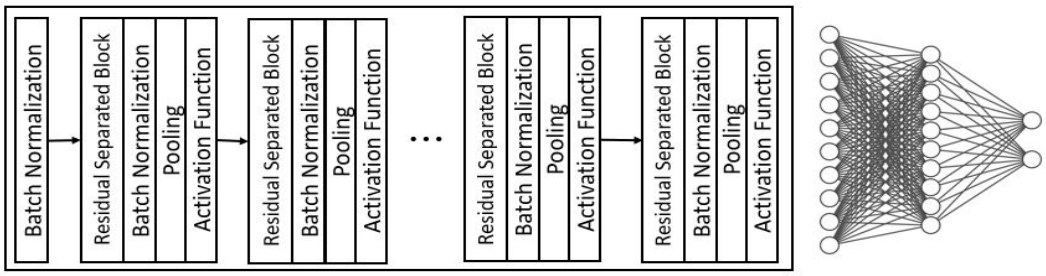


FIGURE 4.5: The complete proposed tailored CNN architecture.

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, convolutional neural network is connected locally to a local region of the previous feature map. This introduces the concept of the network receptive field [43]. 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 is used in one of them. Residual Separated block, RSB, in Fig. 4.4 may have kernel sizes of 3, 5, 7, and 9, respectively.
Fig. 6.2 Represent a complete CNN architecture.

4.2 Summary

In this chapter a lightweight CNN architecture is proposed for COVID19 detection. Proposed architecture is based on spatial separability of the convolutional kernel to enforce the learning of linear kernels. The proposed architecture consists of separated kernels convolutional layers that is connected by a residual

connection. The proposed architecture uses batch normalization to maintain the network stability during the training process.

Chapter 5

Proposed Methodology II

CNN, like many computer vision models, is a scale-variant [44] model such that it cannot recognize objects at various scales unless it explicitly trained to recognize such objects. This chapter presents a CNN architecture that learn multiscale features using scale pyramid. Scale pyramid is constructed using atrous convolution. The correct scale from scale pyramid is selected using the spatial attention mechanism.

5.1 Methodology II

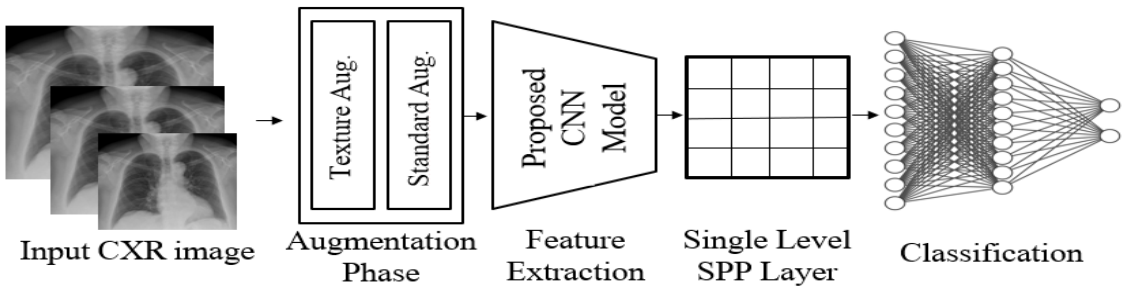


FIGURE 5.1: Proposed method for COVID-19 classification from CXR images.

The proposed system presented in this chapter proposes a novel CNN micro-architecture model for learning scale-invariant features of CXR images and then classifies these features into normal or COVID-19 cases. Fig. 5.1 illustrates the proposed 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 model

is then used to fuse the extracted multi-scale features and select the relevant scale features that the next CNN network should consider.

5.1.1 Data augmentation

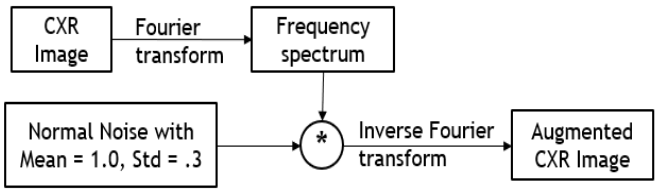


FIGURE 5.2: Texture Augmentation module

The first phase of the proposed CXR classification system is data augmentation. Data augmentation is used to reduce the overfitting and artificially enlarge the training dataset [23]. The input CXR images are augmented using texture augmentation. Texture augmentation is performed by adding a multiplicative normally distributed noises to the frequency spectrum of the input image. Noise is modeled using $\mathcal{N}(\mu = 1, \sigma = 0.3)$. Fig. 5.2 illustrates texture augmentation process. Fig. 5.3 shows the resultant CXR image. A standard augmentation such as random rotation, horizontal flipping, and vertical flipping are included in the augmentation process.

FIGURE 5.3: Texture Augmentation
The resulting CXR image from Texture augmentation **left**: is the original image. **Right** is the augmented CXR Image

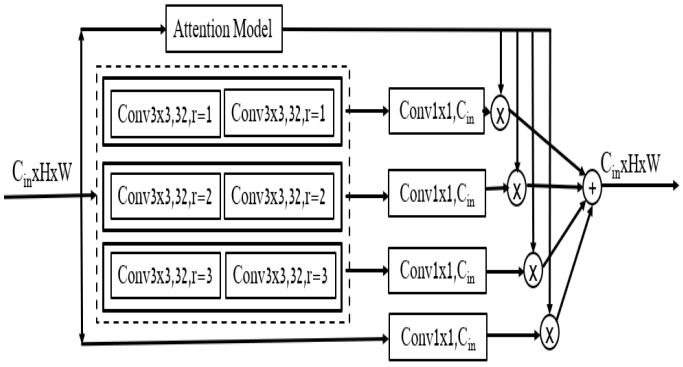


FIGURE 5.4: Spatially weighted atrous spatial Pyramid Pooling (SWASPP) interal layers within dashed square are parameter shared.

5.1.2 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 without increasing neither the number of kernel parameters nor the computational complexity of the convolution performance. A novel spatially weighted atrous spatial pyramid pooling (SWASPP) micro-architecture is presented. Fig. 5.4 shows the architecture structure. In Fig. 5.4, internal layers, bounded by dashed square, are parameter-shared and have different atrous rates. These layers are responsible for extracting multi-scale features. Sharing of the parameters enforce these layers to learn scale-invariant features. For a given input CXR image, three scales feature maps are produced. Each feature map corresponds to a particular scale.

To fuse the produced feature maps representing different scales of the input image, an attention module is added. Attention module can be thought as a pixel level classification of which scale does this spatial position belong. Fig. 5.5 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.

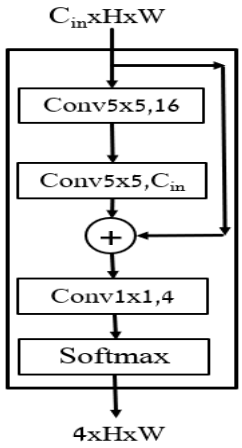


FIGURE 5.5: Attantion module structure used by SWASPP micro-architecture

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 [23] non-linearity except the re-projection layers that used to project back to the input space. BN allows the use of larger learning rate[26] and makes network stable during training[26]. BN makes the loss landscape of the optimization problem significantly smoother[42]. leakyReLU is used to reduce the vanishing gradient problem [23]. A bottleneck is introduced within both the attention module and multi-scale feature extractor layer. 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 [25].

5.1.3 Proposed CNN Architecture

SWASPP is densely stacked [25] together as Fig. 5.6 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 [25]. Residual layers are easier to optimize if the required mapping is the identity mapping or simply near to it [22]. 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 max-pooling to reduce the spatial size and enlarge the Network receptive field. A single level Spatial Pyramid Pooling (SPP) [45] 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.

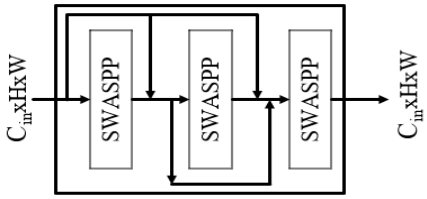


FIGURE 5.6: Densely connected SWASPP (DSWASPP): is a stack of densely connected SWASPP, such that the output of any SWASPP is Concatenated to the input of all next layers. All the three layers produce an output of dimension of $C_{in} \times H \times W$.

The fixed length feature vector produced by SPP is used as an input to dropout [39] layer. Dropout layer randomly sets the activation of to 0 with a probability of 0.5. Dropout prevents the overfitting and reduce complex co-adaptation between the neurons allowing them to learn better representation [39]. It allow implicit ensempling of exponential number of sampled thin network from the original network which enhance the network performance [39]. 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 5.1 shows the details of the proposed architecture.

5.2 Summary

CNN is a scale variant model. Many approaches are introduced to overcome this problem such as shared networks, feature pyramid network and atrous convolution. Atous convolution increases the receptive field of the convolutional kernel without increasing the parameter number. Atrous convolution is used in the proposed work II to construct the scale space of the input feature. To select the correct scale of the input a spatial attention module is used. Attention is a technique to guide the network to select most relevant part of the network

TABLE 5.1: 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

to process. A novel CNN architecture is proposed that internally produces multiscale feature maps which is further fused using attention based mechanism. Compact representation is learned via a bottleneck dimension which is introduced in both the multiscale feature extractor module and the attention module.

Chapter 6

Experimental Results

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

All models are Trained using QaTa-Cov-19 [46] dataset using NVIDIA Tesla P-100 GPU and programmed using PyTorch.

6.1 QaTa-COV19 Dataset

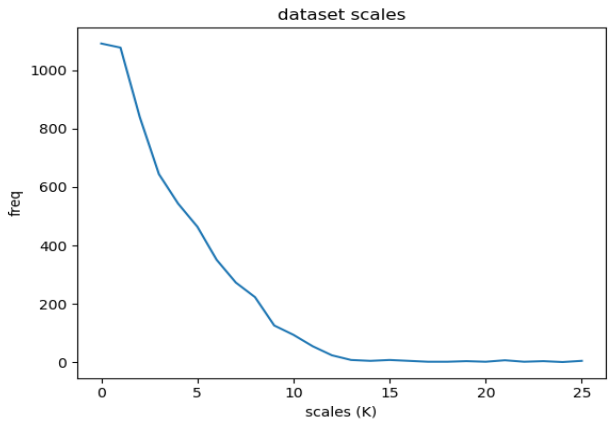


FIGURE 6.1: Pneumonia Scales of QaTa-COV19-v1, Y-axis represents the frequency, number of occurrence, of a pneumonia with a particular area

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

Scales of QaTa-COV19-v1 does not exhibit a uniform distribution. Scale of the Pneumonia can be defined as number, area, of 8-neighbor connected pixels labeled as COVID-19 pneumonia. QaTa-COV19-v1 provides a binary masks of 2951 COVID-19 CXR image which can be used for approximating the distribution of scales across the data set. Fig. 6.1 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 large scales.

6.2 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.

6.2.1 Details of the Proposed Architecture

The Proposed architecture composed Convbase and Densebase. Convbase is composed of a 6 feature extraction modules *(FX)* preceded by batch normalization layer as shown in Fig.5. Each FX module can be considered sub-sequential model consists of RSB layer followed by Batch Normalization, Max-pooling and LeakyReLU activation function. The Densebase is a two fully connected layers that classify the Convbase output.

6.2.2 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 [27] as activation function as shown in. 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 Convbase model part is 1D feature vector of 576 length. Densebase model part consists of two hidden layers. Each layer has the 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 6.1 summarizes the architecture hyperparameters.

TABLE 6.1: 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

6.2.3 Network Training

The proposed CNN model is trained for 22 epoch. Adaptive Moment Estimation (Adam) optimizer [47] 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 is changed during the training phase if the training loss stopped decreasing. Table 6.2 summarizes the parameters values used in the training phase of the proposed CNN model. Fig. 6.2(a) show the progress for training and validation loss across each epoch. The difference between the training loss and validation loss through epochs show that our did not memorize the dataset.

TABLE 6.2: 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

6.2.4 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 [48]. In addition, the number of the parameters used in the training phase is very important comparison factor. Table 6.3 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 6.3: 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 [49]	21.8M	96.77	100.00	98.36	98.33
ACoS Phase I [50]	-	98.266	96.512	98.551	98.062
ResNet-50 [49]	25.6M	95.24	100.00	97.56	97.50
GoogleNet [49]	5M	96.67	96.67	96.67	96.67
VGG-16 [49]	138M	95.08	96.67	95.87	95.83
AlexNet [49]	60M	96.72	98.33	97.52	97.50
MobileNet-V2 [49]	3.4M	98.24	93.33	95.73	95.83
Inception-V3 [49]	24M	96.36	88.33	92.17	92.50
SqueezeNet [49]	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

6.3 Evaluation of the Methodology II

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

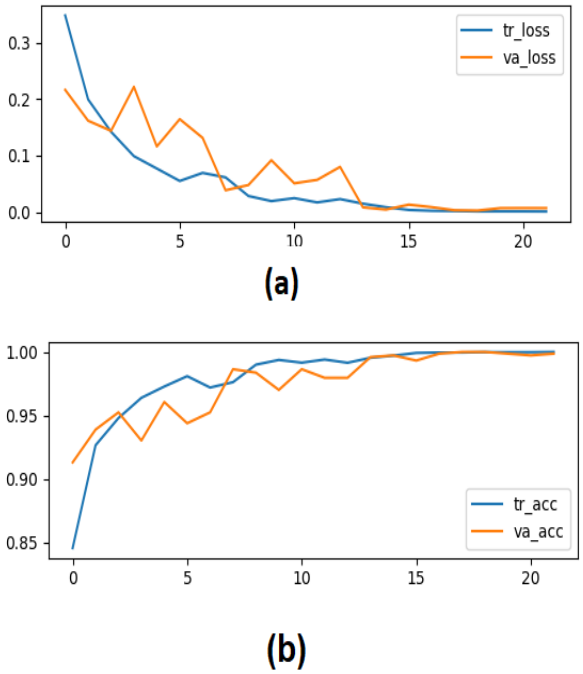


FIGURE 6.2: (a) The training loss and the validation loss of each epoch and (b) The training accuracy and the validation accuracy of each epoch.

6.3.1 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[45] 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 aggregation function
- full SPP pyramid of 8-levels using max-pooling as aggregation function
- single level SPP with 10-bins using average-pooling as aggregation function
- single level SPP with 10-bins using max-pooling as 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 together. For SPP-net variants training a multiscale augmentation is added to the proposed augmentation process. Multiscale 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

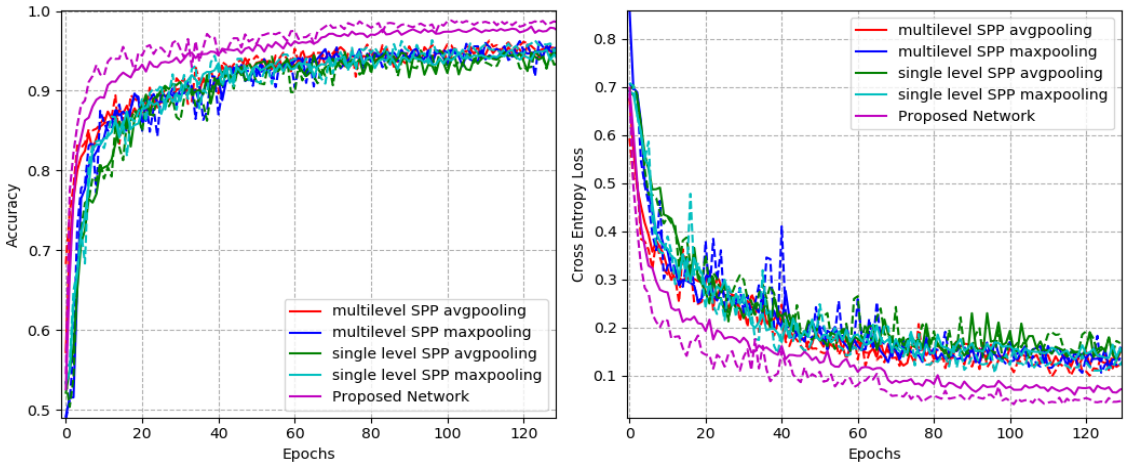


FIGURE 6.3: Training profiles of both SPP-net variants and the proposed network. For the same color solid line represents training statistics while dashed line represents the validation statistics for the corresponding model. **left:** is the training accuracy. **Right:** is the training loss.

Another Base-line is introduced for comparison which is exactly as same as the proposed network but with different Attention module structure and does not include a bottleneck within ASPP. This architecture is referred Switchable Atrous Spatial Pyramid Pooling (SASPP-net). 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* $\mathbb{R}^4$ *a 4 values for the three scales and the input feature map.*

Table 6.4 summarizes the base-line models and the Corresponding parameter count.

TABLE 6.4: 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

6.3.2 Models Training

Proposed architecture and baseline architecture are trained with the same hyperparameters. Dataset is split to 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 [47] 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 norm of 1 [23]. A batch size of 128 is used to calculate the gradient.

6.3.3 Reducing the overfitting

Overfitting is a critical problem for training large networks [23]. Proposed work has reduced the overfitting by using:

- Using Dropout with retrain probability of 0.5 [39].
- Using BatchNorm adds noise due to randomization introduced when constructing the minibatch [26].
- Using max norm constraint [23].
- Deep and thin architectures by design has an implicit regularization effect [22].
- Augmentation process i.e.) Texture augmentation [23].
- The use of small kernel size [21].

- Bottleneck in SWASPP module and the attention module.

During training no overfitting effects is observed.

6.3.4 Comparison with baselines

Proposed network is compared with the vanilla SPP-based Architectures and ASPP architecture.

Comparing with SPP-nets

Fig. 6.3 illustrates both training loss and training and validation accuracies and losses. Table 6.5 illustrates the testing accuracy for comparison between the SPP-nets baseline and the proposed architecture.

TABLE 6.5: Comparison between Proposed network and baseline SPP architectures

Model Name	Accurracy
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. 6.4 illustrates the training and validation loss of training a SASPP baseline architecture. As shown in Fig. 6.4 SASPP unable to generalize and start overfitting the training set. This comparison empirically shows the importance of the bottleneck introduced in the proposed architecture.

6.3.5 Comparing with the related works

To fairly compare with the related works proposed work is further trained. Fig. 6.5 shows the training and validation loss of the proposed network. Fig. 6.6 shows the of the training and validation accuracy.

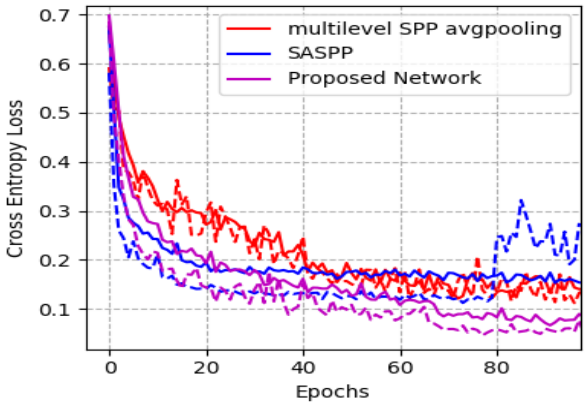


FIGURE 6.4: SASPP baseline architecture loss during both training, solid line, and validation, bashed line, compared with Proposed network and best performing SPP architecture.

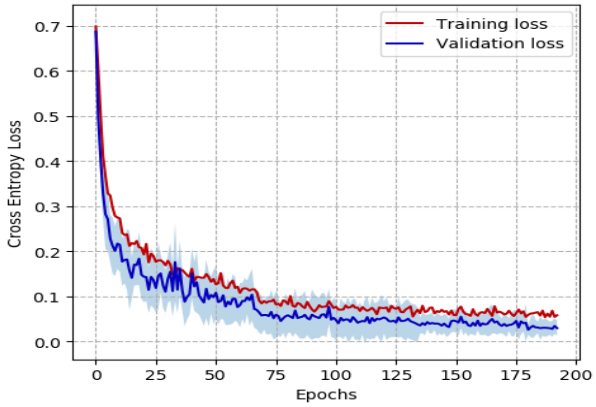


FIGURE 6.5: Cross entropy loss of the proposed architecture.

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

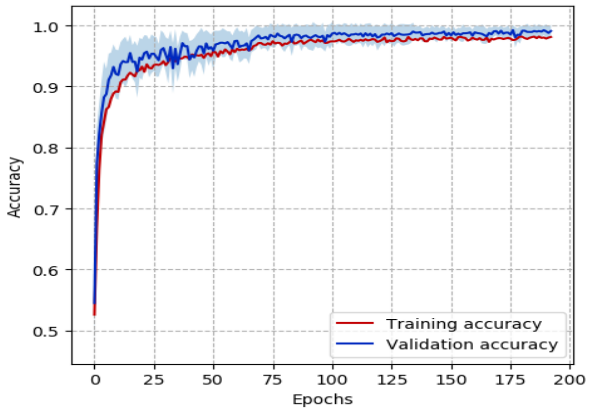


FIGURE 6.6: Training and validation accuracy of the proposed architecture.

TABLE 6.6: 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[51]	0.985	0.886	-	0.993	-	-
SRC-Hom[51]	0.977	0.921	-	0.982	-	-
CRC-light[51]	0.973	0.955	-	0.974	-	-
DenseNet121*[51]	0.992	0.9714	-	0.9949	-	6,955,906
Inception-v3[51]	0.993	0.954	-	0.998	-	21,772,450
Modified MobileNetV2 [52]	0.98	0.98	0.97	-	0.97	-
ReCovNet-v2[53]	0.99726	0.98571	0.94262	0.9977	0.96369	-
ReCovNet-v1[53]	0.99824	0.9781	0.97438	0.99901	0.97624	-
DenseNet-121[53]	**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.

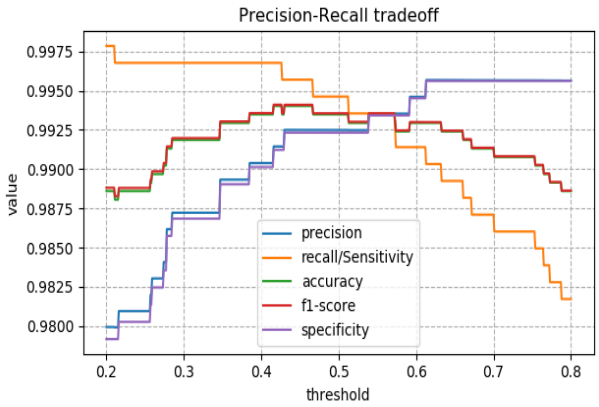


FIGURE 6.7: precision-recall trade-off of the proposed network.

Table 6.6 summarizes the comparison between the recent related works and the proposed architecture. Proposed architecture outperform these works in many metrics. As their training and testing does not depend on a balanced number of samples, accuracy and specificity are not good metrics for evaluation.

6.4 Summary

This chapter illustrates the superior performance of the proposed work I and II. Experimental results of proposed work I show the effectiveness of spatial separable kernels and residual connection for detecting COVID-19. The proposed architecture I use batch normalization to maintain the network stability during the training process. During the training process, the hyperparameters (such as batch size and learning rate) are determined dynamically. Proposed architecture I outperformed previous works for binary classification of chest X-Ray images to normal or COVID-19 cases. The proposed architecture has a very low parameter count (150K trainable parameter) compared to previous work. The proposed architecture I achieved a performance of 100% for accuracy, sensitivity, precision and F1-score. Proposed work I does not take care of the fact that CNN is scale variant model while proposed work II does. Better quantitative results for CXR COVID-19 classification can be obtained with a multiscale training approaches. Proposed work II internally produces multiscale feature maps using Atrous Spatial pyramid pooling. These multiscales feature maps are fused using an attention module. To learn a compact representation a bottleneck dimension is introduced in both the multiscale feature extractor module and the attention module. Proposed work II outperformed current sate-of-the-art architecture with lower parameter number. Proposed method has recorded a 0.9929 for $F1 - score$.

Chapter 7

Conclusion

Appendix A

السيرة الذاتية لمقدم الرسالة

- العنوان : سملا مركز

Appendix B

موجز الرسالة

sfdsfdfsdf

Bibliography

[1] Thamina Acter et al. “Evolution of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) as coronavirus disease 2019 (COVID-19) pandemic: A global health emergency”. In: *Science of the Total Environment* 730 (2020), p. 138996.

[2] Tanu Singhal. “A review of coronavirus disease-2019 (COVID-19)”. In: *The indian journal of pediatrics* 87.4 (2020), pp. 281–286.

[3] David S Hui et al. “The continuing 2019-nCoV epidemic threat of novel coronaviruses to global health—The latest 2019 novel coronavirus outbreak in Wuhan, China”. In: *International journal of infectious diseases* 91 (2020), pp. 264–266.

[4] Sara Platto, Tongtong Xue, and Ernesto Carafoli. “COVID19: an announced pandemic”. In: *Cell Death & Disease* 11.9 (2020), pp. 1–13.

[5] Tung Thanh Le et al. “Evolution of the COVID-19 vaccine development landscape”. In: *Nat Rev Drug Discov* 19.10 (2020), pp. 667–668.

[6] Alireza Tahamtan and Abdollah Ardebili. “Real-time RT-PCR in COVID-19 detection: issues affecting the results”. In: *Expert review of molecular diagnostics* 20.5 (2020), pp. 453–454.

[7] Sana Salehi et al. “Coronavirus disease 2019 (COVID-19): a systematic review of imaging findings in 919 patients”. In: *Ajr Am J Roentgenol* 215.1 (2020), pp. 87–93.

[8] Fan Wu et al. “A new coronavirus associated with human respiratory disease in China”. In: *Nature* 579.7798 (2020), pp. 265–269.

[9] Zi Yue Zu et al. “Coronavirus disease 2019 (COVID-19): a perspective from China”. In: *Radiology* 296.2 (2020), E15–E25.

[10] KONRAD A Erickson, KR Mackenzie, and AJ Marshall. “Advanced but expensive technology. Balancing affordability with access in rural areas.” In: *Canadian family physician Medecin de famille canadien* 39 (1993), pp. 28–30.

[11] Ali Narin, Ceren Kaya, and Ziynet Pamuk. "Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks". In: *Pattern Analysis and Applications* 24.3 (2021), pp. 1207–1220.

[12] David J Brenner and Eric J Hall. "Computed tomography—an increasing source of radiation exposure". In: *New England journal of medicine* 357.22 (2007), pp. 2277–2284.

[13] Geoffrey D Rubin et al. "The role of chest imaging in patient management during the COVID-19 pandemic: a multinational consensus statement from the Fleischner Society". In: *Radiology* 296.1 (2020), pp. 172–180.

[14] Feng Shi et al. "Review of artificial intelligence techniques in imaging data acquisition, segmentation, and diagnosis for COVID-19". In: *IEEE reviews in biomedical engineering* 14 (2020), pp. 4–15.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[16] Yann LeCun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems* 2 (1989).

[17] Dumitru Erhan et al. "Scalable object detection using deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2147–2154.

[18] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[19] Pierre Sermanet et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229* (2013).

[20] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[21] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[22] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[24] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[25] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[26] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[27] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[28] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*. Apr. 2011.

[29] Josef Sivic and Andrew Zisserman. "Video Google: A text retrieval approach to object matching in videos". In: *Computer Vision, IEEE International Conference on*. Vol. 3. IEEE Computer Society. 2003, pp. 1470–1470.

[30] Kristen Grauman and Trevor Darrell. "The pyramid match kernel: Discriminative classification with sets of image features". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE. 2005, pp. 1458–1465.

[31] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[32] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[33] Meng-Hao Guo et al. "Attention mechanisms in computer vision: A survey". In: *Computational Visual Media* (2022), pp. 1–38.

[34] Long Chen et al. “Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5659–5667.

[35] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[36] Sanghyun Woo et al. “Cbam: Convolutional block attention module”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[37] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[38] Randall Balestriero, Leon Bottou, and Yann LeCun. “The effects of regularization and data augmentation are class dependent”. In: *arXiv preprint arXiv:2204.03632* (2022).

[39] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[40] Roberto Rigamonti et al. “Learning separable filters”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2754–2761.

[41] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[42] Shibani Santurkar et al. “How does batch normalization help optimization?” In: *Advances in neural information processing systems* 31 (2018).

[43] Wenjie Luo et al. “Understanding the effective receptive field in deep convolutional neural networks”. In: *Advances in neural information processing systems* 29 (2016).

[44] Nanne Van Noord and Eric Postma. “Learning scale-variant and scale-invariant features for deep image classification”. In: *Pattern Recognition* 61 (2017), pp. 583–592.

[45] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[46] Mete Ahishali et al. “Advance warning methodologies for covid-19 using chest x-ray images”. In: *IEEE Access* 9 (2021), pp. 41052–41065.

[47] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[48] Mohammad Hossin and Md Nasir Sulaiman. “A review on evaluation metrics for data classification evaluations”. In: *International journal of data mining & knowledge management process* 5.2 (2015), p. 1.

[49] Soumya Ranjan Nayak et al. “Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study”. In: *Biomedical Signal Processing and Control* 64 (2021), p. 102365.

[50] Tej Bahadur Chandra et al. “Coronavirus disease (COVID-19) detection in chest X-ray images using majority voting based classifier ensemble”. In: *Expert systems with applications* 165 (2021), p. 113909.

[51] Mete Ahishali et al. “Advance warning methodologies for covid-19 using chest x-ray images”. In: *IEEE Access* 9 (2021), pp. 41052–41065.

[52] Shamima Akter et al. “COVID-19 detection using deep learning algorithm on chest X-ray images”. In: *Biology* 10.11 (2021), p. 1174.

[53] Aysen Degerli et al. “Reliable COVID-19 Detection Using Chest X-ray Images”. In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 185–189.