An-Najah National University

Faculty of Engineering and IT

Electrical and Telecommunication Engineering Department

Graduation Project 2

COVID-19 diagnosis from chest X-ray images using transfer learning: Enhanced performance by debiasing dataloader

Submitted in partial fulfillment of the requirements for Bachelor degree in

Telecommunication Engineering

Supervisor by: Dr. Allam Mousa

Prepared by: Abeer Sawalha

Academic year 2022 / 2023

Acknowledgments

First of all, thanks to the families for all the support they have provided for us. Thanks the honorable Dr. Allam Mousa who supervised this project from the beginning. Thanks to anyone who helped to complete this project, whether with information or in any way.

Table of Contents

List of Figures

List of Tables

Abstract

Background

Chest X-ray imaging has been proved as a powerful diagnostic method to detect and diagnose COVID-19 cases due to its easy accessibility, lower cost and rapid imaging time.

Objective

This study aims to improve efficacy of screening COVID-19 infected patients using chest X-ray images with the help of a developed deep convolutional neural network model (CNN) entitled *nCoV-NET*.

Methods

To train and to evaluate the performance of the developed model, datasets were collected from different resources. Overall, 15251 COVID-19 cases and 14818 non-COVID 19 cases are involved in this report. To overcome the probable bias due to the unbalanced cases in two classes of the datasets, ResNet, DenseNet, and Inception architectures were re-trained in the fine-tuning stage of the process to distinguish COVID-19 classes using a transfer learning method. Lastly, the optimized final *nCoV-NET* model was applied to the testing dataset to verify the performance of the proposed model.

Results

Although the performance parameters of all re-trained architectures were determined close to each other, the final nCOV-NET model optimized by using DenseNet-121 architecture in the transfer learning stage exhibits the highest performance for classification of COVID-19 cases with the accuracy of 97.1 %. The Activation Mapping method was used to create activation maps that highlights the crucial areas of the radiograph to improve causality and intelligibility. Therefore, there was an ability to diagnose COVID-19 disease using efficient artificial intelligence specially CNNs methods

Conclusion

This report demonstrated that the proposed CNN model called *nCoV-NET* can be utilized for reliably detecting COVID-19 cases using chest X-ray images to accelerate the triaging and save critical time for disease control as well as assisting the radiologist to validate their initial diagnosis.

Chapter 1: Introduction

The COVID-19 outbreak has risen to the status of one of the most severe public health issues of the last several years. The virus spreads rapidly: the reproduction number or R number of COVID-19 considered as an excellent tool to measure an infectious disease's capacity to spread. The R number signifies the average number of people that one infected person will pass the virus to.But it isn't a fixed number,since it ccould be affected by a range of factors, including not just how infectious a disease is but how it develops over time, how a population behaves, and any immunity already possessed thanks to infection or vaccination. Location is also important: a densely populated city is likely to have a higher R than a sparsely peopled rural area.

However, R number varied from 2.24 to 3.58 during the initial months of the pandemic, indicating that each infected individual on average transmitted the disease to two or more others. Consequently, the number of COVID-19 infections grew up from a few hundred cases (most of them in China) in January 2020 to more than 500 million cases in May 2022 disseminated across the world.

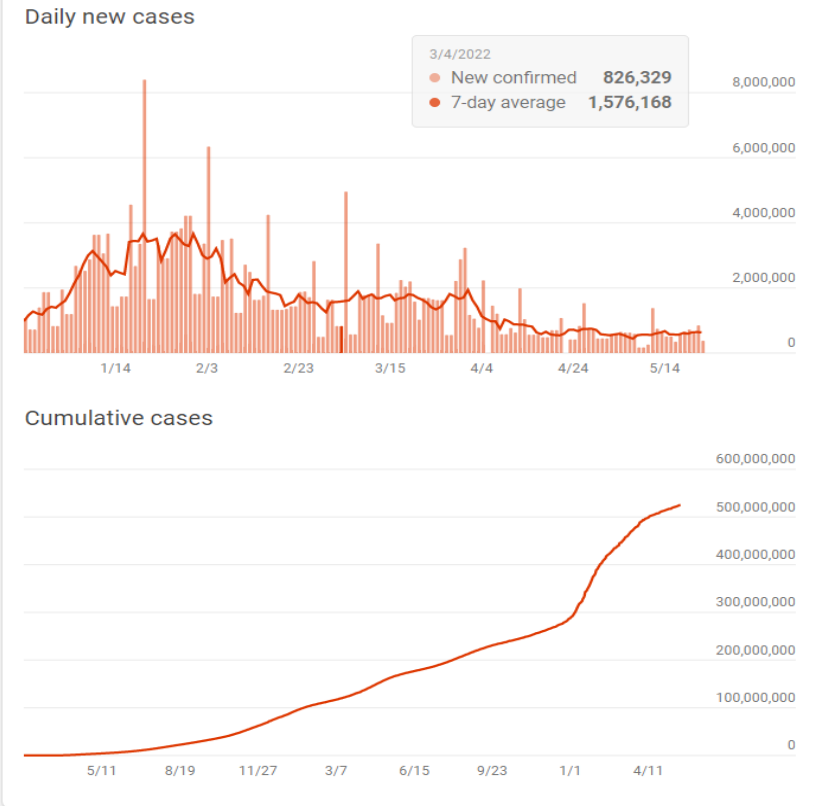


Figure 1 : Global statics for COVID -19 pandemic with confirmed cases, the x-axis shows different dates when the cases were confirmed, and y-axis shows the number of confirmed cases

It is believed that the coronavirus that causes COVID-19 is the same one that causes SARS-COV2 and MERS (MERS). COVID-19 has a wide range of symptoms that appear after an average incubation period of 5.2 days. Fever, dry cough, and tiredness are common symptoms, while others include headache, hemoptysis, diarrhea, dyspnea, and lymphopenia. In December 2019, the first human was infected with coronavirus (SARS-COV-2), and it is mostly transmitted by droplets produced when infected people talk, cough, or sneeze. Because the droplets are too heavy to travel far, they can only be propagated via direct contact. Recent research estimates that the COVID-19 can survive up to 3 h in the air, 4 h on copper, and 72 h on plastic and stainless steel, but the precise durations are yet unknown. However, the general health research community is yet to agree on answers to these issues, which are still under study. An infection with COVID-19 affects the tissues of the lungs. Some affected people may not notice any symptoms in the early stages, while fever and cough were the most common core symptoms for the majority of patients. Other side effects include muscle pains, a sore throat, and a headache. Cough medicine, painkillers, fever reducers, and antibiotics are provided to patients based on their symptoms, not the disease organism. The patient must be hospitalized and treated in an Intensive Care Unit (ICU), which may include the use of a ventilator to help the patient breathe. Because of its severity and ease of transmission, COVID-

19 has spread rapidly across the world. The greater effect on health care departments is primarily due to the number of individuals impacted day by day, as they need to give mechanical ventilators to critically ill patients admitted to ICU. As a result, the number of ICU beds must be significantly expanded. In the aforementioned scenario, early diagnosis is critical in ensuring patients receive appropriate treatment, while reducing the load on the healthcare system. COVID-19 is still a deadly disease due to the absence of early diagnostic techniques around the world, as well as having medical preconditions such as cancer, chronic liver, lung, and kidney diseases, and diabetes. However, RT-PCR diagnosis techniques are available in most parts of the world; under-developed countries still cannot afford to test all their people promptly. Moreover, they might not be able to give a precise diagnosing for COVID-19 since they have only 60% - 70% for accuracy. So, other modernized and more accurate technique should be implemented. In 2020–2021, this disease claimed the lives of millions of people across the earth. Vaccines for COVID-19 are now being developed in a number of countries. Vaccines produced by Pfizer, AstraZeneca, Moderna, Sinopharm, Sinovac, etc. that are among the vaccines that have been approved by WHO for administration. Such approved vaccines have substantially reduced the deadliness of the disease. However, artificial intelligence has shown its efficiency and excellent performance in automated image categorization issues via various machine-learning methods and is currently being used to automate the diagnosis of various diseases. Furthermore, machine learning refers to models that can learn and make decisions based on vast quantities of data samples. Artificial intelligence accomplishes activities that need human intellect, such as voice recognition, translation, visual perception, and more, by performing calculations and predictions depending on the incoming data. Deep learning is a collection of machine learning techniques that primarily concentrate on the automated extraction and categorization of image features, and has shown tremendous promise in a variety of applications, particularly in health care. Scientists from around the world are trying to develop technologies that can assist radiologists/doctors in their diagnoses. To identify the optimal network for the area of radiology and medical image processing, a variety of AI methods have been used so far .Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two prominent deep-learning-based networks that have been extensively utilized in medical research fields like speech recognition, computer vision, and Natural Language Processing (NLP), and have often produced commendable results. The classification, localization, and segmentation of images using CNN have shown impressive results in medical image processing.

This is a significant step forward in the identification of COVID-19 and other forms of lung inflammation due to artificial intelligence (AI). The WHO recommends using RT-PCR as the main diagnostic method for COVID-19 detection. Chest X-rays or chest CT scans are also widely used, but should only be preferred when the RT-PCR test is not available in a timely manner. In most cases, it is seen that the nasal swap test results vary if done at different times of the day. For instance, the possibility of getting a positive result is higher if the test is done in the morning compared to a test done in the evening. Furthermore, the detection of COVID-19 positive patients was significantly delayed due to a high number of false-negative results. The RT-PCR has a low success rate of 70% and a sensitivity of 60–70%. Moreover, the false-negative cases of RT-PCR show positive results on chest X-ray imaging. Due to its widespread availability, X-ray imaging has played a significant role in many medical and epidemiological situations. Because of its operating speed, low cost, and ease of use for radiologists, chest X-ray seems to be promising for emergency situations and therapy. A prior study, however, found significant discrepancies in chest X-ray images obtained from patients who had COVID-19.

The project proposed three models of an intelligent deep learning architecture such as a DenseNet, ResNet, and Inception with RMSprop optimizer to detect COVID-19 disease. For the study, 30069 X-ray image data were collected from the dataset. Image processing methods such as enhancement, normalization, and data augmentation were used to help the proposed model not only to avoid over fitting but also to demonstrate the highest accuracy. The performance accuracy losses of the newly proposed methods are compared among each other to conclude that ResNet was the most efficient method to apply for deep learning applications, which discussed later in this project with fine details.

The remainder of the report is organized as follows: chapter 2 will present a background, literature review and related works, chapter 3 will include the methodology that provides details on how the project will work, next chapter will mention the outcomes that wewould be used , and the last chapter will present a prototype GUI machine which would be helpful to show some conclusions and discussions for the work.

Chapter 2: Theoretical Background and Previous Work

An overview of the deep learning approaches for the analysis of COVID-19 CXR images is arranged in this section. Moreover, I have compiled this section in three aspects, specifically, relevant investigations about COVID-19 in the medical domain, related DL works in CAD of COVID-19 and related deep transfer learning works in CAD field. It has been reviewed the proposed methodology, dataset, data preprocessing, evaluation method, and the result of each paper.

2.1 Relevant investigations about COVID-19 in the medical domain

Fang et al. proposes a study to examine the sensitivity of chest CT image and viral nucleic acid detection technique using real-time polymerase chain reaction (RT-PCR) for the detection of COVID-19 cases [1]. Fifty-one subjects who had a travel or residency history in Wuhan within 14 days and had a fever or acute respiratory symptoms, with an average age of 45 years, are taken part of this study. Thirty-six patients diagnosed positive for COVID-19 in the 1st RT-PCR test. Twelve of them get positive results on the 2nd RT-PCR tests (1 to 2 days). Two patients get positive by 3rd RT-PCR tests (2-5 days) and one patient by four tests (7 days). However, 98% of the 51 patients shown evidence of viral pneumonia on the 1st day. Therefore, the finding of this study is that the sensitivity of chest CT is higher than RT-PCR (98% vs. 71%, respectively).

Bernheim et al. analyzed chest CTs of 121 symptomatic positive COVID-19 patients and finds the relationship of the outcome of chest CTs in time between symptom onset of the initial CT scan (i.e., early, 0 to 2 days (36 patients), intermediate 3 to 5 days (36 patients) and late 6 to 12 days (25 patients)) [2]. The observation of this investigation is that 28% of the early patients (10/36), 76% of intermediate patients (25/33), and 88% of late patients (22/25) show bilateral lung disease.

Tao et al. examines 1014 patients in Wuhan from 6th January to 6th February and observes that chest CT is more sensitive for the diagnosis of COVID-19 over RT-PCR [11]. Out of 1014 patients, 601 patients test positive in RT-PCR, while 888 of them tests positive in chest CT. They also analyze that the mean interval time is 5.1 ± 1.5 days from initial negative to positive and 6.9 ± 2.3 days from positive to negative in RT-PCR.

In order to find the role of chest CT and CXR in the management of COVID-19, Rubin et al. in [3] comprise a multidisciplinary panel of radiologists and pulmonologists from 10 different countries with experience of managing COVID-19 patients. Fourteen key questions and

corresponds to 11 decision points are presented on three aspects, such as varying risk factors, community condition, and resource constraints. The results are analyzed and found that there are five primary and three additional recommendations for the medical practitioners to use of CXR and CT in the management of COVID-19.

2.2 Related DL works in CAD of COVID-19

Farooq et al. recommend a deep learning-based approach, namely COVID-ResNet, to differentiate COVID-19 cases form other types of pneumonia cases in [4]. The author also builds an open-access dataset for CXR image research. A pre-trained ResNet is used, which is followed by three steps of fine-tunes work. In the three fine-tuned steps, input images are resized continuously form 128 × 128 × 3, 224 × 224 × 3 to 229 × 229 × 3. Among 6009 images they utilize in the proposed algorithm, 68 of them are COVID-19 cases. They obtained an overall accuracy of 96.23% for four-class classification.

In [5], Afshar et al. suggests a binary classification of the CXR images based on the Capsule Network framework known as COVID-CAPS to identify COVID-19 cases. 90% of the total dataset is employed for training, and 10% is for validation. The advanced framework reaches an accuracy of 95.7%, and the pre-trained framework achieves an accuracy of 98.3%.

Wang et al. introduce COVID-Net [6], which is a deep CNN based approach to detect COVID-19 cases form CXR images. The network is trained on publicly available 13,800 CXR images. Among them, 183 are COVID cases. The model achieves an overall accuracy of 92.6% in a three-class classification.

A new deep learning-based framework, namely COVIDX-Net, is suggested by Hemdan et al. [7] to diagnose COVID-19 from CXR images automatically. The network is composed of VGG19, DenseNet121, ResNetV2, InceptionV3, InceptionResNetV2, Xception and MobileNetV2. A total of 50 CXR images is considered for testing the representation of the proposed algorithm; among them, 25 are positive COVID-19 images. 80% of the images are used for training, and rests are used for testing. All images are resized to 224 × 224 pixels. VGG19 and DenseNet201 achieve an accuracy of 90%.

Shety et al. [8] suggests a deep learning-based framework for the detection of COVID-19 from CXR images. The proposed methodology utilizes different CNN models to extract features and fed into SVM for classification. Twenty-five normal and 25 COVID-19 images are used to evaluate the performance. 60%, 20%, and 20% of images are used for training,

validation, and testing the model. ResNet50 achieves the highest classification accuracy of 95.38%.

Maghdid et al. proposes an AI tools that can help radiologists to diagnose COVID-19 cases from X-rays and CT scan images quickly [9]. Modified pre-trained AlexNet is used as the backbone of the network. 170 CXR and 361 CT images are utilized in this model. 50% of the overall dataset is granted for training, and 50% is for validation. Modified CNN achieves an accuracy of 94.1%, and a pre-trained network obtains 98% accuracy in binary classification.

Apostolopoulos et al. [10], employ state of the art CNN model called MobileNet from 3905 X-Ray images, correspond to six different classes. Moreover, 455 COVID-19 CXR images are also incorporated. The images are resized to 200 × 200 pixels, and the augmentation task is performed in the preprocessing step. They achieved an accuracy of 99.18% in binary classification tasks and 87.66% for the seven class classification tasks.

Loey et al. suggests Generative Adversarial Network (GAN) with deep transfer learning for the detection of COVID-19 from CXR images in [11]. Since there is a shortage of COVID-19 CXR images; therefore, to generate more images, they employ GAN. A total number of 307 images are collected in four different classes, such as COVID-19, normal, bacterial pneumonia, and viral pneumonia. In the training and validation step with GAN, 90% of the dataset is utilized, while 10% is kept for testing. Pretrained AlexNet, GoogleNet, and Resnet18 are used as transfer learning. In the proposed model they achieved an accuracy of 80.6%, 85.3% and 100% on four class (GoogleNet), three class (AlexNet) and two class (GoogleNet) classification problem respectively.

As discussed above, promising COVID-19 identification results have been obtained using CXR with transfer learning and various deep learning models, mostly CNN, have been utilized. However, the relative performance results of various CNN transfer learning are unclear. This paper focuses on the performance comparison of CNN models (15 total) for CXR COVID-19 identification using transfer learning.

2.3 Related deep transfer learning works in CAD fields

Zhang et al. proposes a deep transfer learning-based classification approach for the classification of the cervical cell into normal and abnormal [12]. One private dataset called HEMLBC and publicly available Herlev dataset are considered to evaluate the performance of the proposed method. In the preprocessing work, they perform the patch extraction and image

augmentation operations. They achieve an accuracy of 98.3% and 98.6% on the Herlev and HEMLBC dataset individually.

Chen et al. introduces a deep transfer learning-based framework based on the Inception-V3 network for the classification of cervical immunohistochemistry images in [13]. The data augmentation task is performed in the preprocessing step. They obtain an average accuracy of 77.3%.

Song et al. carries out a transfer learning-based CNN that is pre-trained on the ImageNet dataset for the classification of breast histopathology images [14]. First, the images are represented in the Fisher Vector (FV) encoding of local features. Then, an adaptation layer is designed for fine-tuning. Finally, an overall accuracy of 87% obtained with 30% testing data.

Hall et al. [15] explore that chest X-ray images are beneficial for the diagnosis of COVID-19 viral infection. They utilize pre-trained ResNet50 and tuned on 102 COVID-19 cases and 102 other pneumonia cases and achieves an overall accuracy of 90.7%. Moreover, in another experiment, they combine pre-trained VGG16, ResNet50, and custom made CNN train and test on 33 unseen COVID-19 CXR and 208 pneumonia cases and obtained an overall accuracy of 94.4%.

In [16], Zhang et al. proposes a deep learning model to identify COVID-19 CXR images from standard CXR images. Pretrained ResNet is used as a backbone of the network. 50% of 100 COVID-19 and 1430 Normal CXR images are considered for training, and rest are regarded for testing. This algorithm can detect 96% of COVID-19 cases and 70.65% non-COVID-19 cases.

Abbas et al. suggests that their previously developed CNN, namely Decompose, Transfer and Compose network (DeTraC-Net), utilize pre-trained ResNet-50 as transfer learning to classify COVID-19 chest X-ray images from normal and severe acute respiratory syndrome cases [17]. 50 standard, 105 COVID-19, and 11 SARS CXR images are applied to evaluate the performance of their proposed method and achieve an accuracy of 95.12%.

Narin et al. [18] proposes a CNN based approach to detect coronavirus pneumonia among patients using CXR. Here, three pre-trained network models are utilized, namely, ResNet50, InceptionV3, and Inception-ResNetV2. 50 COVID-19 CXR and 50 normal CXR are used to check the performance of the proposed model. 80% of the dataset employed for training and 20% is for testing. They observed that ResNet50 gives the highest classification accuracy of 98% in binary classification, among other models.

Chapter 3: Methodology

3.1 Materials and method

COVID-19 detection should have the highest accuracy for more precise detection, since performance is the key for deep learning models .Beside; it might be an arduous task when it is necessary to have limited resources.Accuracy considered, as one of the important parameter to measure performance is 'accuracy'.

There are several methods to achieve higher accuracy with various techniques related to data, algorithm tuning and model optimization.The effects on accuracy could be noticeable after varying parameters like learning rate, batch size, number of training images, images size and number of trainable model parameters.

The following shows all performance improvement techniques.

- **Techniques for performance improvement with hyper-parameter tuning**

1. L2 regularization by penalizing complexity of the model, penalizing big weights. L2 regularization encourage weights to be small but doesn't force them to exactly 0.For explanation of regularization concept , Firstly , it would be with its definition , types , and present a comparison between the most common types of it ,specially L2 regularization. However, before regularization an early technique used in the early era of neural networks, which is labelled as early stopping technique.

So, early stopping was invented in the early days of neural networks. It had a straightforward approach; the training of the network is being stopped before it overfits to the training data. Early stopping can be regarded as straightforward in the aspect of understanding the basics; the training should be stopped when generalization decreases. Exactly when this occurs could be clear in some cases but less so in other. Therefore, a problem with this method can be deciding an appropriate stopping criterion, which takes both the generalization and the training time in aspect. The goal is then to find a criterion which secures a generalized network but stops when the increase in generalization is not of substantial character anymore; with the purpose that the training time can be seemed effective. A common stopping criterion is to stop training when the validation loss has not reached a new minimum in the last 5 epochs. Hence continuous improvements do not trigger Early stopping. These improvements considered using new modified techniques such as regularization with its various types.

Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain. One of the most important aspects when training neural networks is avoiding overfitting. Overfitting refers to the phenomenon where a neural network models the training data very well but fails when it sees new data from the same problem domain. Overfitting is caused by noise in the training data that the neural network picked up during training and learns it as an underlying concept of the data. This learned noise, however, is unique to each training set. As soon as the model sees new data from the same problem domain, but that does not contain this noise, the performance of the neural network gets much worse.The reason for this is that the complexity of this network is too high. A fit of a neural network with higher complexity is shown in the figure below :

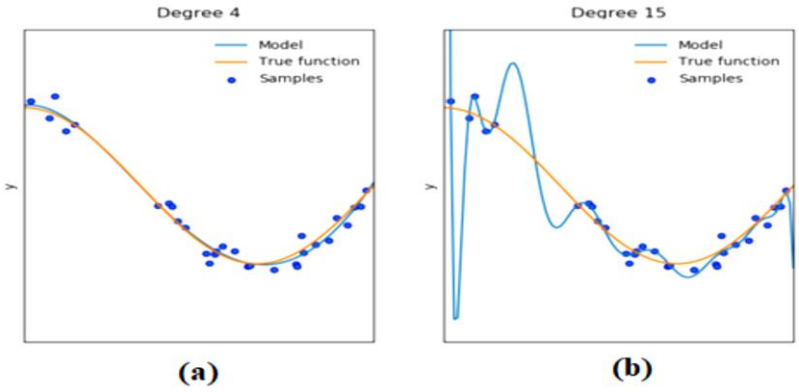


Figure 2: A fit of a neural network with higher complexity : (a) Good fit model , (b) High variance model

The model with a higher complexity is able to pick up and learn patterns (noise) in the data that are just caused by some random fluctuation or error. The network would be able to model each data sample of the distribution one-by-one, while not recognizing the true function that describes the distribution.New arbitrary samples generated with the true function would have a high distance to the fit of the model. Also, it is possible to mention that the model has a high variance.On the other hand, the lower complexity network on the left side models of the figure above ,the distribution much better by not trying too hard to model each data pattern individually.As a summary , Less complex neural networks are less susceptible to overfitting. To prevent overfitting or a high variance we must use something that is called regularization which refers to a set of different techniques that lower the complexity of a neural network model during training, and thus prevent the overfitting.

There are three very popular and efficient regularization techniques called L1, L2, and dropout.each one described carefully , as shown below :

3.1.1 L1 Regularization

It is also called regularization for sparsity. As the name suggests, it is used to handle sparse vectors which consist of mostly zeroes. Sparse vectors typically result in very high-dimensional feature vector space. Thus, the model becomes very difficult to handle.L1 regularization forces the weights of uninformative features to be zero by substracting a small amount from the weight at each iteration and thus making the weight zero, eventually.L1 regularization penalizes |weight|.Moreover,In the case of L1 regularization (also knows as Lasso regression), simply another regularization term Ω used .This term is the sum of the absolute values of the weight parameters in a weight matrix:

$$\boldsymbol{\Omega(W) = \left|\left|W\right|\right|_1 = \sum_i \sum_j |w_{ij}|}$$

Eq. 1: Regularization Term for L1 Regularization.

As in the previous case, the regularization term was multiplied by alpha and add the entire thing to the loss function.

$$\boldsymbol{\mathcal{L}`(W) = \alpha\left|\left|W\right|\right|_1 + \mathcal{L}(W)}$$

Eq. 2: Loss function during L1 Regularization.

The derivative of the new loss function leads to the following expression, which the sum of the gradient of the old loss function and sign of a weight value times alpha.

$$\boldsymbol{\nabla_w \mathcal{L}`(W) = \alpha sign(W) + \nabla_w \mathcal{L}(W)}$$

Eq. 3: Gradient of the loss function during L1 Regularization.

3.1.2 L2 Regularization

The L2 regularization is the most common type of all regularization techniques , also is commonly known as weight decay or Ride Regression. It is also called regularization for simplicity. If the model complexity was considered as a function of weights, the complexity of a feature is proportinal to the absolute value of its weight.the following equation presented a proper illustration

$$\mathbf{y = W_1X_1 + W_2X_2 + \cdots W_nX_n}$$

$$\mathbf{L2\ Regularization\ terms\ = W_1^2 + W_2^2 + \cdots + W_n^2}$$

Eq.4: The upper equation or side represents the complexity of a model and its relationship with the absolute value of its weight, note that x considered as an inputted data .However, the lower equation or side describes L2 gularization terms , it is important to note that L2 regularization penalizes (weight) [2]

L2 regularization forced weights toward zero but it does not make them exactly zero. L2 regularization acted like a force that removed a small percentage of weights at each iteration. Therefore, weights would never be equal to zero.There is an additional parameter to tune the L2 regularization term which called regularization rate (lambda). Regularization rate is a scalar and multiplied by L2 regularization term . The following equation would give more descriptions

$$\mathbf{minimize\ [Loss(model) + Lambda * Complexity(model)]}$$

Eq.5 : L2 Regularization

Where : Lambda indicates the regularization rate , it is also called Alpha (α)

Complexity (model) indecaties the L2 Regularization term

Note: Choosing an optimal value for lambda is important. If lambda is too high, the model becomes too simple and tends to underfit. On the other hand, if lambda is too low, the effect of regulatization becomes negligible and the model is likely to overfit. If lambda is set to zero, then regularization will be completely removed (high risk of overfitting). Unfortunately , the mathematical derivation of this regularization, as well as the mathematical explanation of the reason why this method works at reducing overfitting, is quite long and complex.For more

explanation and derivation , it is important to notice that during the L2 regularization the loss function of the neural network as extended by a so-called regularization term, which called here Ω.

$$\boldsymbol{\Omega(W) = \left||W|\right|^2_2 = \sum_i \sum_j w_{ij}^2}$$

Eq.6: Regularization Term

The regularization term Ω is defined as the Euclidean Norm (or L2 norm) of the weight matrices, which is the sum over all squared weight values of a weight matrix. The regularization term is weighted by the scalar alpha divided by two and added to the regular loss function that is chosen for the current task. This leads to a new expression for the loss function:

$$\boldsymbol{\mathcal{L}`(W) = \frac{\alpha}{2}\left||W|\right|^2_2 + \mathcal{L}(W) = \frac{\alpha}{2}\sum_i \sum_j w_{ij}^2 + \mathcal{L}(W)}$$

Eq.7: Regularization loss during L2 regularization.

Alpha is sometimes called as the regularization rate and is an additional hyperparameter. It determines how much the model could be regularized .Now, in the next step the gradient of the new loss function computed, and then it is possible to put the gradient into the update rule for the weights:

$$\boldsymbol{\nabla_w \mathcal{L}`(W) = \alpha(W) + \nabla_w \mathcal{L}(W)}$$

$$\boldsymbol{W_{new} = W_{old} - \epsilon(\alpha W_{old} + \nabla_w \mathcal{L}((W)_{old}))}$$

Eq. 8: Gradient Descent during L2 Regularization.

Some reformulations of the update rule lead to the expression which is similar to the update rule for the weights during regular gradient descent:

$$\boldsymbol{W_{new} = (1 - \epsilon\alpha)W_{old} - \epsilon \mathcal{L}((W)_{old})}$$

Eq.9: Gradient Descent during L2 Regularization.

The only difference is that by adding the regularization term an additional subtraction from the current weights (first term in the equation) could be presented . In other words independent of the gradient of the loss function we are making the weights a little bit smaller each time an update is performed.The following table shows the main differences between L1 and L2 regularizations.

L1 Regularization	L2 Regularization
Sum of absolute value of weights	Sum of square of weights
Spare Solution	Non - Spare Solution
Multiple solutions	Only one Solution
Built – in feature selection	No feature selection
Robust to outliers	Not robust to outliers (due to the square term)

Table 1: Comparison of L1 and L2 regularization

It is important to note that there is another type of regression, which called Elastic net regression that combines L1 and L2 regularization.However, it is the least common type of regression and rarely to be used to avoid serious issues related to Artificial Intelligence (AI) models specifically, deep learning models such as over- fitting

3.1.3 The importance of L1 and L2 Regularizations

It is essential to consider the plots of the end functions, where represents the operation performed during L1 and the operation performed during L2 regularization.

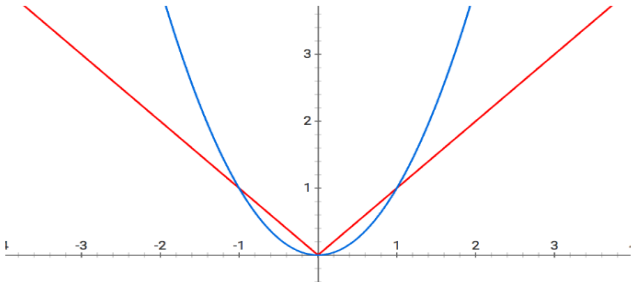


Figure 3: L1 function (red) , L2 function (blue)

In the case of L2 regularization, the weight parameters decrease, but not necessarily become zero, since the curve becomes flat near zero. On the other hand during the L1 regularization, the weight are always forced all the way towards zero.In the case of L2, It could be considered as a solution of an equation, where the sum of squared weight values is equal or less than a value (s) where (s) is the constant that exists for each possible value of the regularization term α (Alpha). For just two weight values W_1 and W_2 this equation would look as follows: $W_1{}^2 + W^{22} \leq s$ .On the other hand, the L1 regularization could be thought of as an equation where the sum of modules of weight values is less than or equal to a value (s). That would look like the following expression: $|W_1| + |W_2| \leq s$. Basically the introduced equations for L1 and L2 regularizations are constraint functions, which it is proper to visualize them as shown in the figure below.

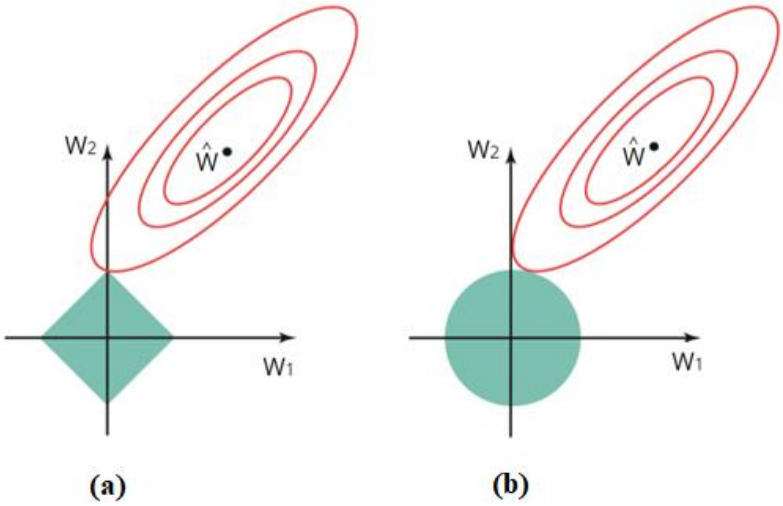


Figure 4: The introduced equations for L1 and L2 regularizations as constraint functions, (a) shows the constraint function (green area) for the L1 regularization. (b) Shows the constraint function for the L2 regularization

The red ellipses are contours of the loss function that is used during the gradient descent. In the center of the contours there is a set of optimal weights for which the loss function has a global minimum.In the case of L1 and L2 regularization, the estimates of W_1 and W_2 are given by the first point where the ellipse intersects with the green constraint area.Since L2 regularization has a circular constraint area, the intersection won't generally occur on an axis, and this the estimates for W_1 and W_2 will be exclusively non-zero.In the case of L1, the constraints area has a diamond shape with corners. And thus the contours of the loss function will often intersect the constraint region at an axis. Then this occurs, one of the estimates (W_1 or W_2) will be zero.In a high dimensional space, many of the weight parameters will equal zero simultaneously. Moreover, Performing L2 regularization encourages the weight values towards

zero (but not exactly zero).But, Performing L1 regularization encourages the weight values to be zero.

Intuitively speaking smaller weights reduce the impact of the hidden neurons. In that case, those hidden neurons become neglectable and the overall complexity of the neural network gets reduced.As previously mentioned , less complex models typically avoid modeling noise in the data, and therefore, there is no overfitting. It is important to choose the the regularization term α carefully, since the goal is to achieve the right balance between low complexity of the model and accuracy.If alpha (α) value is too high, then the model will be simple, but it would run the risk of underfitting the data.The model won't learn enough about the training data to make useful predictions.If alpha value is too low, the model will be more complex, and It is possible to run the risk of overfitting the initiated data. However, the model will learn too much about the particularities of the training data, and will not be able to generalize to new data.

3.1.4 Dropout

In addition to the L2 and L1 regularization, another famous and powerful regularization technique for acting against an overfitted network is called the dropout regularization or Dropout as short . As the name may give a hint of this method randomly drops nodes in the network's hidden layers. This is done by forcing the weight for these randomly selected nodes to zero and then scaling up the remaining weights. How much to scale up is in perfect correlation to the Dropout rate.

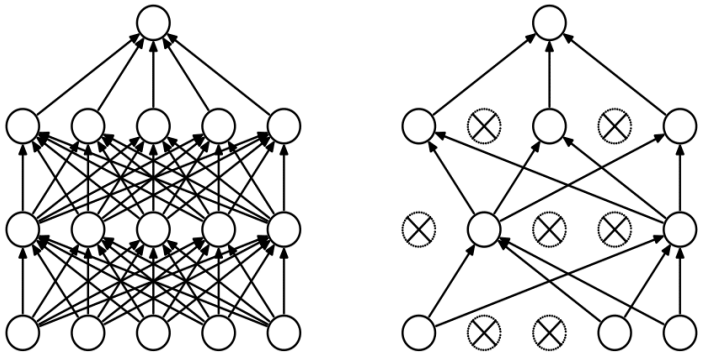


Figure 5: Dropout Neural Net Model . (a) A standard neural net with 2 hidden layers. (b) An example of a thinned net produced by applying dropout to the network .Note that crossed units have been dropped

For example, if 1/5 of the nodes are dropped the remaining weights should be multiplied with 6/5. An often-used number of dropouts is between 20-50%. This method can be seen as arbitrary, because of the randomness, but is in contrast often used for overfitted neural networks

and is regarded as effective. The idea of this technique is to make the network more uncertain when it updates the weights; to make it only pick up the more significant patterns. The network otherwise risks to pick up random noise in the training data but is now less likely to do so, because Dropout has added more noise

2. Learning Rate (LR) optimization ;the weights of a neural network cannot be calculated using an analytical method. Instead, the weights must be discovered via an empirical optimization procedure called stochastic gradient descent.The optimization problem addressed by stochastic gradient descent for neural networks is challenging and the space of solutions (sets of weights) may be comprised of many good solutions (called global optima) as well as easy to find, but low in skill solutions (called local optima).

The amount of change to the model during each step of this search process, or the step size, is called the "learning rate" and provides perhaps the most important hyperparameter to tune for the neural network in order to achieve good performance on any problem. The Learning Rate is deep learning neural networks are trained using the stochastic gradient descent algorithm.

Stochastic gradient descent is an optimization algorithm that estimates the error gradient for the current state of the model using examples from the training dataset, then updates the weights of the model using the back-propagation of errors algorithm, referred to as simply backpropagation.The amount that the weights are updated during training is referred to as the step size or the "learning rate."

Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. The learning rate is often represented using the notation of the lowercase Greek letter eta (η).

During training, the backpropagation of error estimates the amount of error for which the weights of a node in the network are responsible. Instead of updating the weight with the full amount, it is scaled by the learning rate.

This means that a learning rate of 0.1, a traditionally common default value, would mean that weights in the network are updated 0.1 * (estimated weight error) or 10% of the estimated weight error each time the weights are updated. A neural network learns or approximates a function to best map inputs to outputs from examples in the training dataset.

The learning rate hyperparameter controls the rate or speed at which the model learns. Specifically, it controls the amount of apportioned error that the weights of the model are updated with each time they are updated, such as at the end of each batch of training examples.

Given a perfectly configured learning rate, the model will learn to best approximate the function given available resources (the number of layers and the number of nodes per layer) in a given number of training epochs (passes through the training data).

Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights ,but might take significantly longer to train.However , to ensure the optimum accuracy for the model it is possible to start with a base LR and subsequently decreasing it for next epoch.

At extremes, a learning rate that is too large will result in weight updates that will be too large and the performance of the model (such as its loss on the training dataset) would oscillate over training epochs. Oscillating performance is said to be caused by weights that diverge (are divergent). A learning rate that is too small may never converge or may get stuck on a suboptimal solution.

3. Batch Size by trying with max batch size that GPU can handle, depends on the memory of the GPU. However, the higher the batch size the higher the network accuracy, meaning that the batch size has a huge impact on the CNN performance.
4. Increase model capacity by Increasing model depth (more number of layers) and width (number of filters in each convolution layer) would increase the performance of the model (specially CNN models) and the accuracy of its results.

- **Techniques for performance improvement with data redesigning**

    1. Increase image resolution (progressive resizing) by increasing the size of that image From 128 x 128 x 3 to 256 x 256 x 3 or to even higher size.

    2. Random image rotations by changing the orientation of the image.

    3. Random image shifts which would be useful when object is not at the center of the image.Otherwise, it might lead to deterioration

4. Vertical and horizontal shift by randomly flip the image vertically or horizontally. (Algorithm should identify an image whether its face up or face down)

- **Techniques for performance improvement with model optimization**

1. Fine tuning the model with subset data by dropping few data samples for some of the overly sampled data classes.
2. Class weights that used to train highly imbalanced (biased) database, class weights will give equal importance to all the classes during training.
3. Fine tuning the model with train data by using the model to predict on training data, retrain the model for the wrongly predicted images.

In this report, I have used the same three models that used previously , which were Inceptionv3 , Resnet50(Residual Networks 50) , Densenet121 (Densely Connected Networks121) .in fact , these models considered as an efficient models to be used for severe diseases detection such as , COVID-19 .

3.2 Proposed methods

3.2.1 ResNet

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet is to make it possible to train extremely deep neural networks with 150+ layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.However, increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient extremely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.One of the problems ResNets solve is the famous known vanishing gradient. This is because when the network is too deep, the gradients from where the loss function is calculated easily shrink to zero after several applications of the chain rule. This result on the weights never updating its values and therefore, no learning is being performed.

With ResNets, the gradients can flow directly through the skip connections backwards from later layers to initial filters.

3.2.1.1 Architecture

The ResNet's architecture use residual mapping (H(x) = F(x) + x) instead of learning a direct mapping (H(x) = F(x)) and these bolcks are called residual bocks. The complete ResNet architecture is consist of many residual bocks with 3 × 3 convolution layers. The following figure illustrates the difference between the direct mapping and the residual mapping.

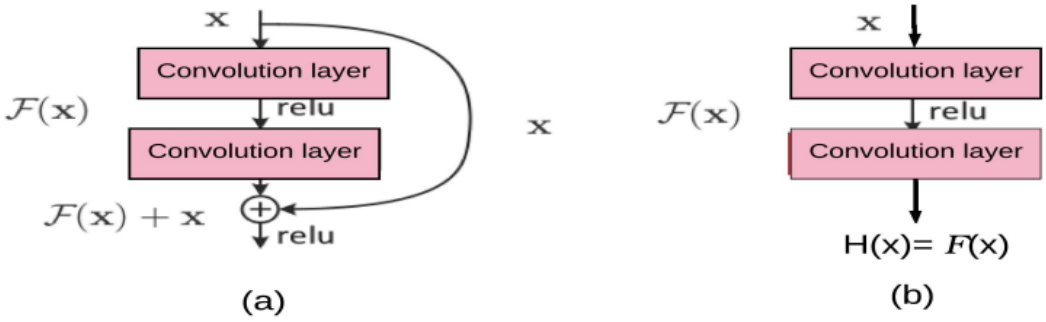


Figure 6:(a) Mapping inside Residual block, (b) Simple direct mappings

It is significant to be aware of the following:

• The authors propose several version of ResNet with different depth, and they also used 'bottleneck' layer for dimensionality reduction in each ResNet architecture that has depth more than 50.

• Although the ResNet (with 152 Layer) is 8 times deeper than VGGNets (22 layers), it has complexity lower than VGGNets (16/19).However, The VGG model, or VGGNet, that supports 16 layers is also referred to as VGG16.The VGG16 model achieves almost 92.7% top-5 test accuracy in ImageNet. ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes. Moreover, it was one of the most popular models submitted to ImageNet Large Scale Visual Recognition Challenge (ILSVRC)-2014. It replaces the large kernel-sized filters with several 3×3 kernel-sized filters one after the other, thereby making significant improvements over AlexNet.there is another type of VGG model , such as VGG19 model.

The concept of the VGG19 model (also VGGNet-19) is the same as the VGG16 except that it supports 19 layers. The "16" and "19" stand for the number of weight layers in the model

(convolutional layers). This means that VGG19 has three more convolutional layers than VGG16.

VGG Architecture VGGNets are based on the most essential features of convolutional neural networks (CNN). The following graphic shows the basic concept of how a CNN works:

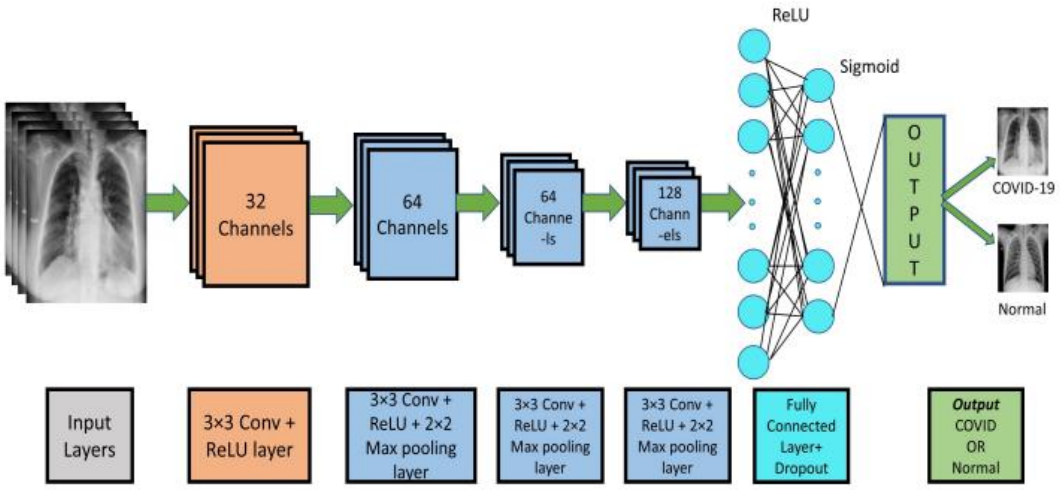


Figure 7:The architecture of a Convolutional Neural Network: Image data is the input of the CNN; the model output provides prediction categories for input images

The VGG network is constructed with very small convolutional filters. The VGG-16 consists of 13 convolutional layers and three fully connected layers. The following demonstrated the architecture of VGG:

Input: The VGGNet takes in an image input size of 224×224. For the ImageNet competition, the creators of the model cropped out the center 224×224 patch in each image to keep the input size of the image consistent. Convolutional Layers: VGG's convolutional layers leverage a minimal receptive field, i.e., 3×3, the smallest possible size that still captures up/down and left/right. Moreover, there are also 1×1 convolution filters acting as a linear transformation of the input. This is followed by a ReLU unit, which is a huge innovation from AlexNet that reduces training time to six days . ReLU stands for rectified linear unit activation function which is a piecewise linear function that will output the input if positive.Otherwise, the output is zero. The convolution stride is fixed at 1 pixel to keep the spatial resolution preserved after convolution (stride is the number of pixel shifts over the input matrix).

Hidden Layers: All the hidden layers in the VGG network use ReLU. VGG does not usually leverage Local Response Normalization (LRN) as it increases memory consumption and training time. Moreover, it makes no improvements to overall accuracy.

Fully-Connected Layers: The VGGNet has three fully connected layers. Out of the three layers, the first two have 4096 channels each, and the third has 1000 channels, only one for each class.

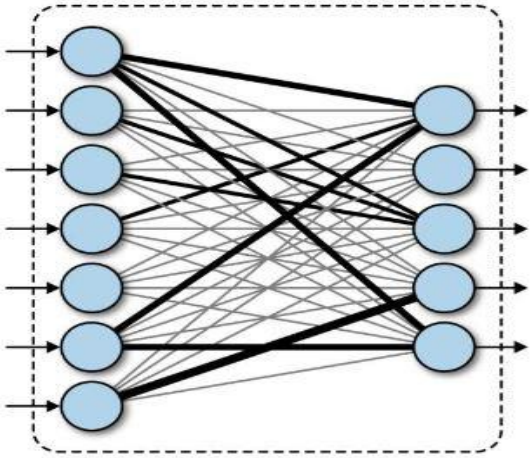


Figure 8: Fully Connected Layers

3.2.1.2 VGG16 Architecture

The number 16 in the name VGG refers to the fact that it is 16 layers deep neural network (VGGnet). This means that VGG16 is a pretty extensive network and has a total of around 138 million parameters. Even according to modern standards, it is a huge network. However, VGGNet16 architecture's simplicity is what makes the network more appealing. Just by had a look at its architecture, it could be said that it is quite uniform. There are a few convolution layers followed by a pooling layer that reduces the height and the width.Currently, around 64 filters are available and could be used, so it is possible to double the number of filters to about 128 and then to 256 filters.In the last layers, it is proper to use 512 filters.

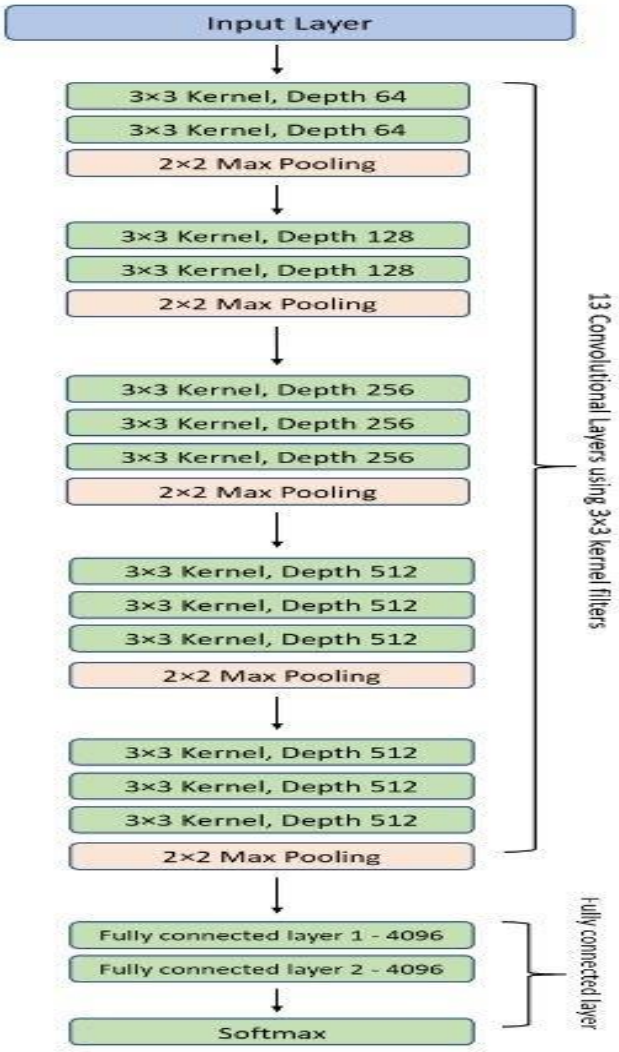


Figure 9:VGG-16 Architecture of a VGG16 model

3.2.1.3 Complexity and challenges among models

The number of filters that could be used doubles on every step or through every stack of the convolution layer. This is a major principle used to design the architecture of the VGG16 network. One of the crucial downsides of the VGG16 network is that it is a huge network, which means that it takes more time to train its parameters. Because of its depth and number of fully connected layers, the VGG16 model is more than 533MB. This makes implementing a VGG network a time-consuming task. The VGG16 model is used in several deep learning image classification problems, but smaller network architectures such as GoogLeNet and SqueezeNet are often preferable. In any case, the VGGNet is a great building block for learning purposes as it is straightforward to implement. Performance of VGG Models VGG16 highly surpasses the previous versions of models in the ILSVRC-2012 and ILSVRC-2013 competitions. Moreover, the VGG16 result is competing for the classification task winner

(GoogLeNet with 6.7% error) and considerably outperforms the ILSVRC-2013 winning submission Clarifai. It obtained 11.2% with external training data and around 11.7% without it. In terms of the single-net performance, the VGGNet-16 model achieves the best result with about 7.0% test error, thereby surpassing a single GoogLeNet by around 0.9%.

3.2.1.4 VGGNet vs. ResNet

VGG stands for Visual Geometry Group and consists of blocks, where each block is composed of 2D Convolution and Max Pooling layers. It comes in two models which are VGG16 and VGG19 with 16 and 19 layers. As the number of layers increases in CNN, the ability of the model to fit more complex functions also increases. Hence, more layers promise better performance. This should not be confused with an Artificial Neural Network (ANN), where an increase in the number of layers doesn't necessarily result in a better performance. Unfortunately, it is impossible to use VGGNet with more layers, such as VGG20, or VGG50, or VGG100, because this is where the problem arises. The weights of a neural network are updated through the backpropagation algorithm or backward propagation of errors, is an algorithm that is designed to test for errors working back from output nodes to input nodes. It is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning. In fact, it makes a minor change to each weight so that the loss of the model decreases. It updates each weight so that it takes a step in the direction along which the loss decreases. This is nothing but the gradient of this weight which can be found using the chain rule. However, as the gradient keeps flowing backward to the initial layers, the value keeps increasing by each local gradient. This results in the gradient becoming smaller and smaller, thereby making changes to the initial layers very small. This, in turn, increases the training time significantly. The problem could be solved if the local gradient becomes 1. This is where ResNet comes into the picture since it achieves this through the identity function. So, as the gradient is back-propagated, it does not decrease in value because the local gradient is 1. Deep residual networks (ResNets), such as the popular ResNet-50 model, are another type of convolutional neural network architecture (CNN) that is 50 layers deep, which is the model that I have trained and tested it through the project. A residual neural network uses the insertion of shortcut connections in turning a plain network into its residual network counterpart. Compared to VGGNets, ResNets are less complex since they have fewer filters. ResNet, also referred to as Residual Network that does not allow the vanishing gradient problem to occur. The skip connections act as gradient superhighways, which allow the gradient to flow undisturbed.

This is also one of the most important reasons why ResNet comes in versions like ResNet50, ResNet101, and ResNet152.The ResNet used SGD learning algorithm with the min-batch size of 128, weight decay of 0.0001 and momentum factor of 0.9.

The ResNet was the winner of ILSVRC-2015 with a big leap in performance, it reduces the top-5 error rate to 3.6% (the previous year's winner GoogleNet has the top-5 error rate to 6.7%).

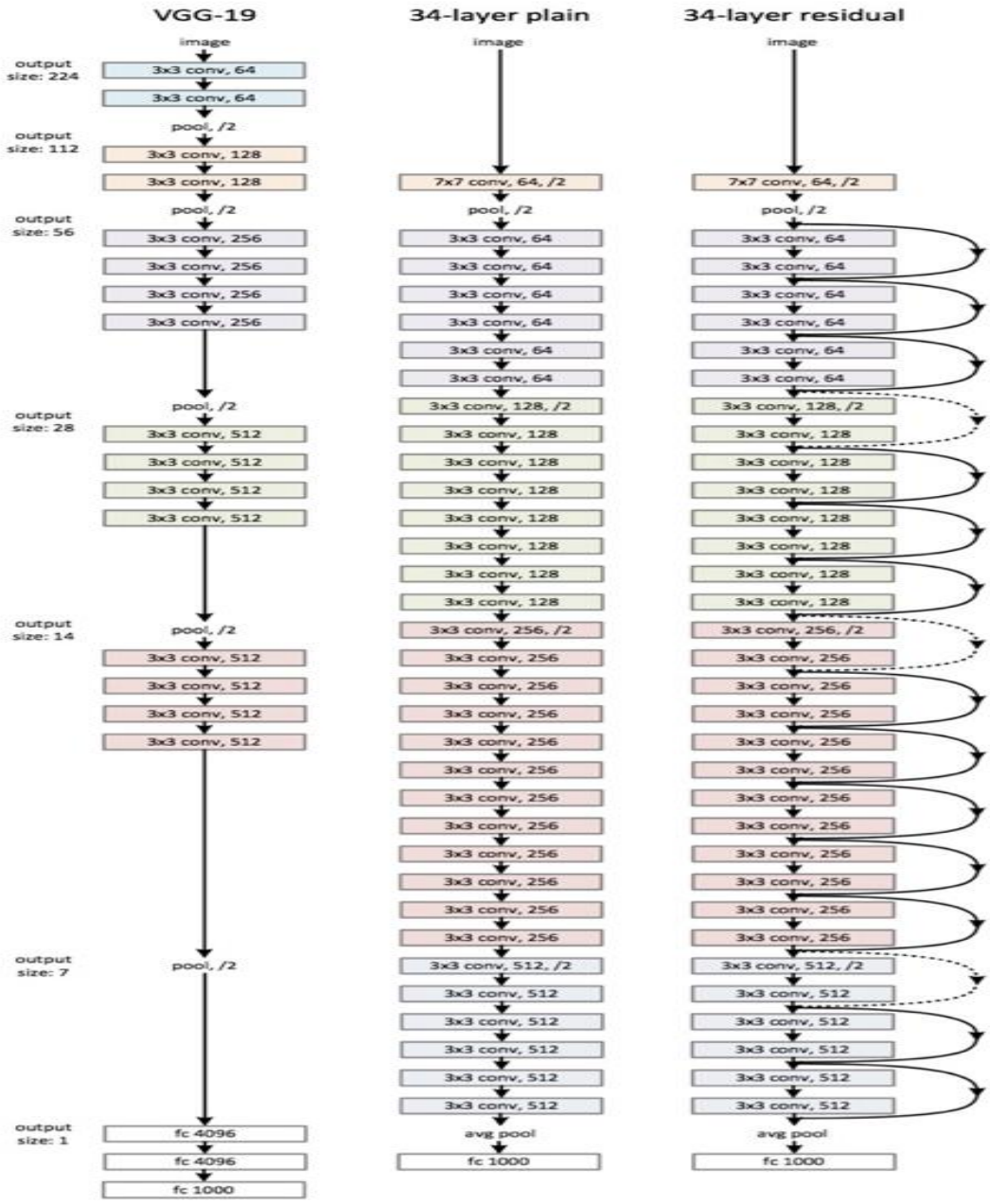


Figure 10:Example network architectures for ImageNet .Left : the VGG – 19 model (19.6 billion FLOPs) as a reference. Middle: a plain network with 34 parameter layers (3.6 billion FLOPs). Right: a residual network with 34-parameter layers (3.6 billion FLOPs).The dotted shortcuts increase dimensions

In more descriptive way, ResNets can have variable sizes, depending on how big each of the layers of the model are, and how many layers it had. Thus, in the figure above it is possible to see that the ResNet (the one on the right) consists on one convolution and pooling step (on orange) followed by 4 layers of similar behavior.

Each of the layers follow the same pattern. They perform 3x3 convolution with a fixed feature map dimension (F) [64, 128, 256, 512] respectively, bypassing the input every 2 convolutions. Furthermore, the width (W) and height (H) dimensions remain constant during the entire layer.

The dotted line occurred, precisely because there has been a change in the dimension of the input volume (for sure the reduction because of the convolution). Note that this reduction between layers is achieved by an increase on the stride, from 1 to 2, at the first convolution of each layer; instead of by a pooling operation.The following table, there is a summary of the output size at every layer and the dimension of the convolutional kernels at every point in the structure.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 11:Sizes of outputs and convolutional kernels for ResNet 34

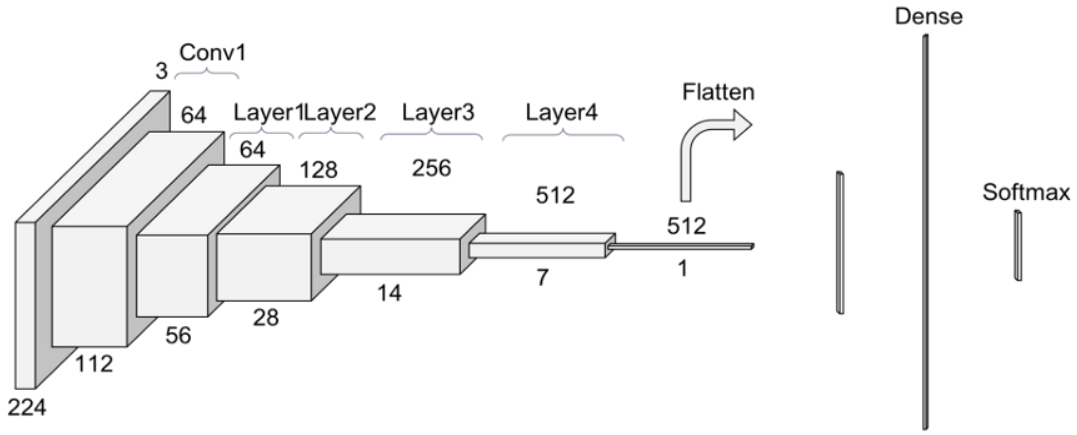


Figure 12:ResNet 34 , note that Resnet 50 has similar block structure but the major differences between both of them are in width and depth

3.2.1.5 Convolution 1

The first step on the ResNet before entering the common layer behavior is a block called Conv1 consisting on a convolution + batch normalization + max pooling operation.

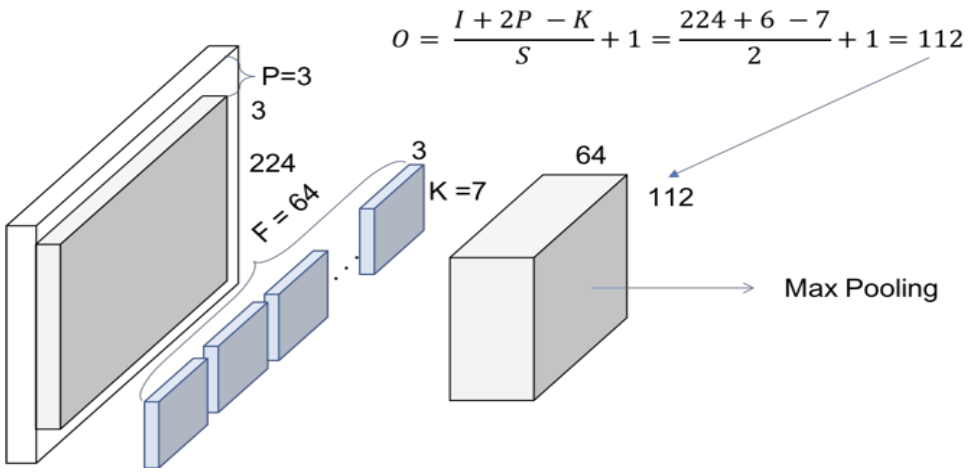


Figure 13:Conv1 (Convolution)

The next step is the batch normalization, which is an element-wise operation and therefore, it did not change the size of the volume. Finally, I have the (3x3) Max Pooling operation with a stride of 2. It is possible to infer that they first pad the input volume, so the final volume has the desired dimensions.

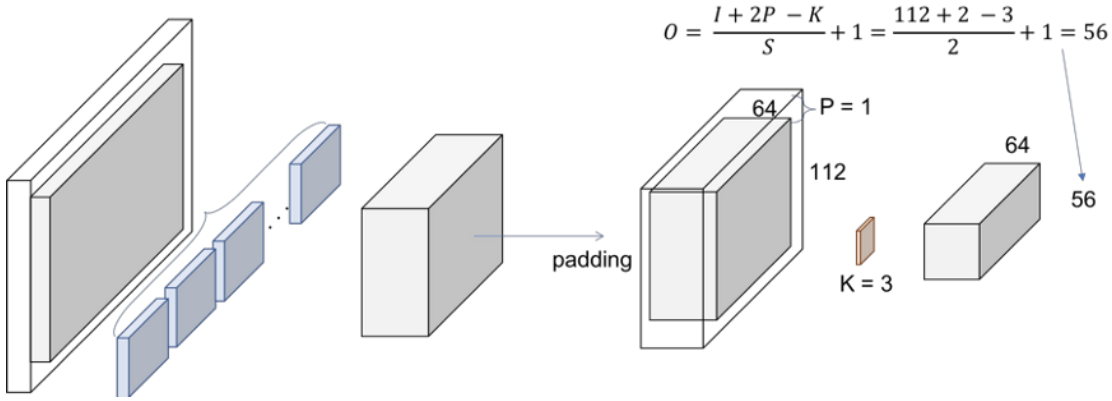


Figure 14:Conv1 (Max Pooling)

3.2.1.6 ResNet Layers

Every layer of a ResNet is composed of several blocks. That is because when ResNets went deeper, it was normally done by increasing the number of operations within a block, but the number of total layers remains the same . An operation refers to a convolution a batch normalization and a ReLU activation to an input, except the last operation of a block, that does not have the ReLU.

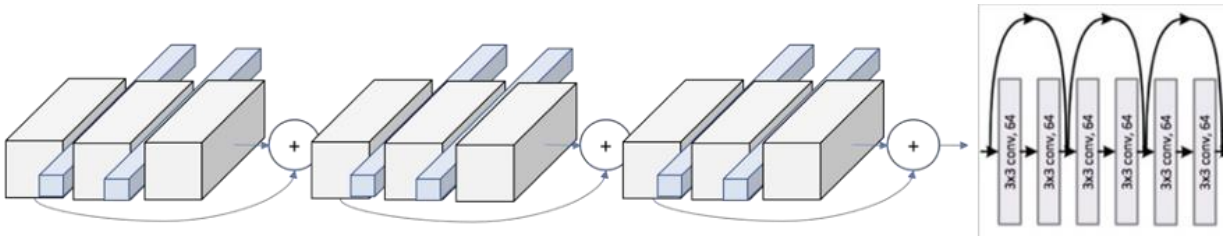


Figure 15:ResNet different layers

3.2.1.7 Patterns

The next step is to escalate from the entire block to the entire layer. In the **Figure 9** it is possible to see how the layers are differentiable by colors. However, the first operation of each layer, was strided which used at that first one is 2, instead of 1 like for the rest of them.note that down sampling of the volume though the network is achieved by increasing the stride instead of a pooling operation like normally CNNs do.

Number of Layers	Number of Parameters
ResNet 18	11.174 Million
ResNet 34	21.282 Million
ResNet 50	23.521 Million
ResNet 101	42.513 Million
ResNet 152	58.157 Million

Table 2:ResNets architectures for ImageNet

3.2.2 DenseNet

Convolutional neural networks (CNNs) have become the dominant machine learning approach for visual object recognition. Although they were originally introduced over 20 years ago , improvements in computer hardware and network structure have enabled the training of truly deep CNNs only recently. The original LeNet5 consisted of 5 layers, VGG featured 19 , and only last year Highway Networks and Residual Networks (ResNets) have surpassed the 100-layer barrier.

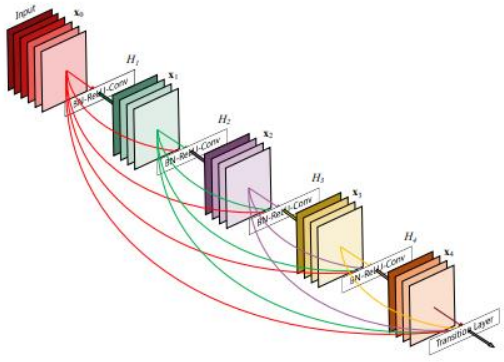


Figure 16:A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input

As CNNs become increasingly deep, a new research problem emerges that for an information about the input or gradient passes through many layers, it can vanish and “wash out” by the time it reaches the end (or beginning) of the network. Many recent publications address this or related problems. ResNets and Highway Networks bypass signal from one layer to the next via

identity connections. Stochastic depth shortens ResNets by randomly dropping layers during training to allow better information and gradient flow. However, FractalNets repeatedly combine several parallel layer sequences with different number of convolutional blocks to obtain a large nominal depth, while maintaining many short paths in the network. Although these different approaches vary in network topology and training procedure, they all share a key characteristic that they create short paths from early layers to later layers. In this report, I proposed an architecture that distills this insight into a simple connectivity pattern , so to ensure maximum information flow between layers in the network, It is vital to connect all layers (with matching feature-map sizes) directly with each other. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. **Figure 15** illustrated this layout schematically. Crucially, in contrast to ResNets, importantly , never combine features through summation before they are passed into a layer; instead, I combine features by concatenating them. Hence, the $\mathcal{L}^{th}$ layer has $\mathcal{L}$ inputs, consisting of the feature-maps of all preceding convolutional blocks. Its own feature-maps are passed on to all $L-\mathcal{L}$ subsequent layers. This introduced $\frac{L(L+1)}{2}$ connections in an L-layer network, instead of just L, as in traditional architectures. Because of its dense connectivity pattern, I would like to refer to the approach as Dense Convolutional Network (DenseNet). A possibly counter-intuitive effect of this dense connectivity pattern is that it requires fewer parameters than traditional convolutional networks, as there is no need to relearn redundant feature-maps. Traditional feed-forward architectures can be viewed as algorithms with a state, which is passed on from layer to layer. Each layer reads the state from its preceding layer and writes to the subsequent layer. It changes the state but also passes on information that needs to be preserved. ResNets make this information preservation explicit through additive identity transformations. Recent variations of ResNets show that many layers contribute very little and can in fact be randomly dropped during training. This makes the state of ResNets similar to (unrolled) recurrent neural networks , but the number of parameters of ResNets is substantially larger because each layer has its own weights. The proposed DenseNet architecture explicitly differentiates between information that added to the network and information that is preserved. DenseNet layers are very narrow (e.g., 12 filters per layer), adding only a small set of feature-maps to the "collective knowledge" of the network and keep the remaining featuremaps unchanged and the final classifier makes a decision based on all feature-maps in the network. Besides better parameter efficiency, one big advantage of DenseNets is their improved flow of information and gradients throughout the network, which made them easy to train. Each layer

has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision . This helped training of deeper network architectures. Further, It is observed that dense connections have a regularizing effect, which reduced overfitting on tasks with smaller training set sizes

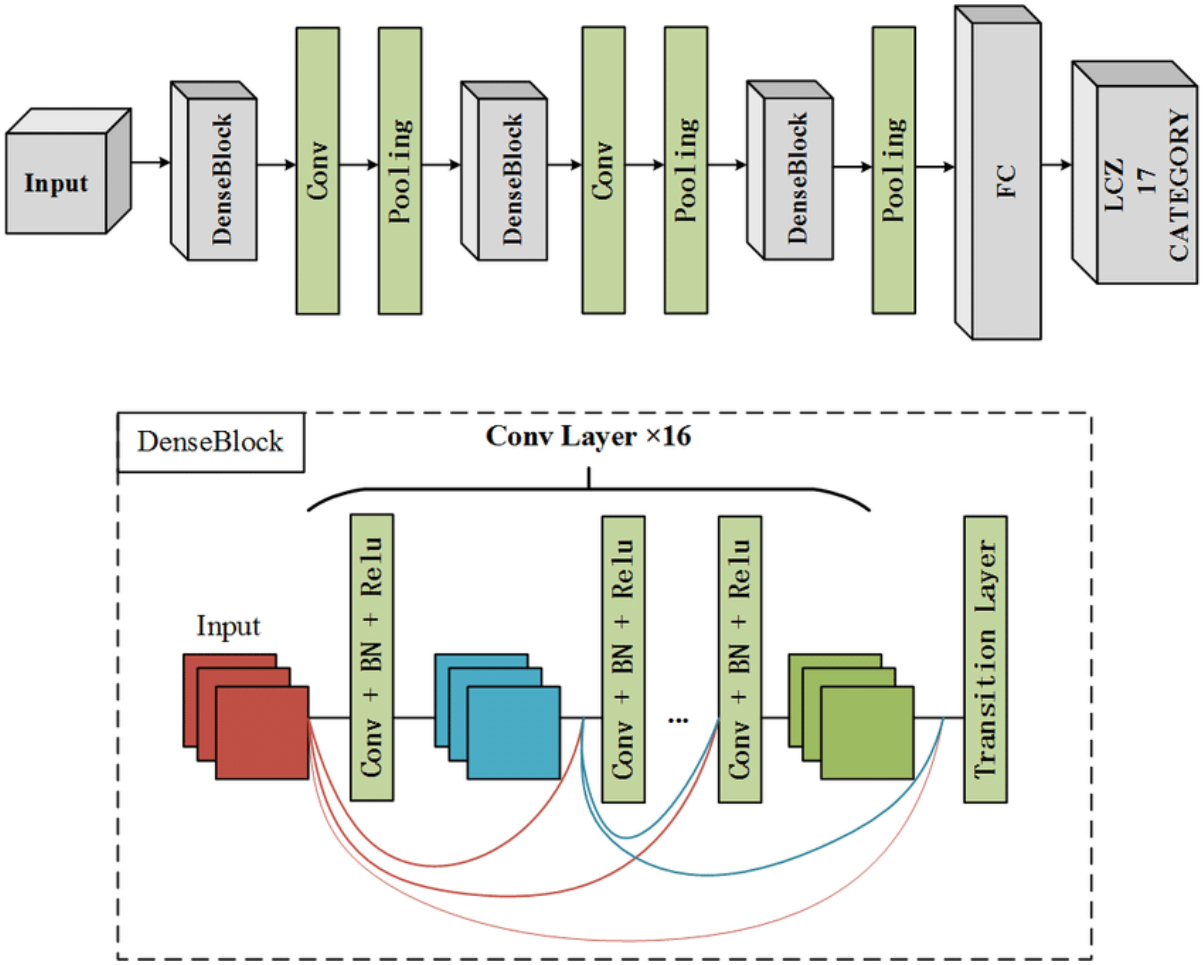


Figure 17:Densenet121 Structure

3.2.2.1 DenseNet Structure

Raditional feed-forward neural networks connect the output of the layer to the next layer after applying a composite of operations.Normally, that composite includes a convolution operation or pooling layers, a batch normalization and an activation function.

The equation (**Eq.10**)for this would be:

$$\mathbf{x_l = H_l(x_{l-1})}$$

ResNets extended the latter behavior including the skip connection, reformulating this equation (**Eq.11**) into:

$$\mathbf{x_l = H_l(x_{l-1}) + x_{l-1}}$$

DenseNets make the first difference with ResNets right here. DenseNets do not sum the output feature maps of the layer with the incoming feature maps but concatenate them. Consequently, the equation reshapes again into (**Eq.12**):

$$\mathbf{x_l = H_l([x_0, x_1, \dots, x_{l-1}])}$$

Unfortunately, this grouping of feature maps could not be done when the sizes of them are different. Regardless if the grouping is an addition or a concatenation. Therefore, and similarly to the implementation in Resnet , DenseNets are divided into DenseBlocks, where the dimensions of the feature maps remains constant within a block, but the number of filters changes between them. These layers between them are called Transition Layers .It is important to be aware of the downsampling applying a batch normalization, a 1x1 convolution and a 2x2 pooling layers. Since I am concatenating feature maps, this channel dimension is increasing at every layer. If I made H_l to produce k feature maps every time, then I could generalize for the l^{th} layer as the represented in the following equation (**Eq.13**)

$$\mathbf{k_l = k + k * (l - 1)}$$

This hyperparameter k is the growth rate. The growth rate regulates how much information is added to the network each layer. In fact, it is possible to see the feature maps as the information of the network. Every layer has access to its preceding feature maps, then, to the collective

knowledge. Each layer is then adding a new information to this collective knowledge, in concrete k feature maps of information

3.2.2.2 DenseNets-B

DenseNets-B are just regular DenseNets that take advantage of 1×1 convolution to reduce the feature maps size before the 3×3 convolution and improve computing efficiency. The B comes after the name Bottleneck layer .

3.2.2.3 DenseNets-BC

DenseNets-C are another little incremental step to DenseNets-B, for the cases where it is proper to reduce the number of output feature maps. The compression factor (theta (θ)) determines this reduction. Instead of having m feature maps at a certain layer, It would have theta*m. Of course, is in the range [0–1]. So DenseNets would remain the same when theta=1, and would be DenseNets-B otherwise.although there are different datasets might worked perfectly with Densenet model , such as Canadian Institute For advanced Research (CIFAR -10) ,the Street View House Numbers (SVHN) , and ImageNet datasets .

CIFAR. The CIFAR datasets consist of colored natural images with 32×32 pixels. CIFAR-10 (C10) consists of images drawn from 10 and CIFAR-100 (C100) from 100 classes. However, SVHN. The Street View House Numbers (SVHN) dataset contains 32×32 colored digit images.finally, the ImageNet dataset that used in the ILSVRC since 2010. It considered as a benchmark in image classification and object detection. Moreover, it consisted of the highest number of images for about 1.2 million images whether they used to train, test, or validate the model .Although, the structure is simpler for CIFAR-10 or SVHN since the input volumes are smaller.But, imageNet dataset must be considered for any medical detection in general.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 18:DenseNet architectures for ImageNet. The growth rate for all the networks is k = 32. Note that each "conv" layer shown in the figure corresponds the sequence BN-ReLU-Conv

3.2.2.4 Dense and Transition Blocks

However, because of the highly dense number of connections on the DenseNets, the visualization gets a little bit more complex that it was for VGG and ResNets. Figure 18 shows a very simple scheme on the architecture of the DenseNet-121,there are other types of densenet such as densenet169 , and densenet201 , which differentiate within the depth of the model . Otherwise, they are adequately similar. Currently, it is important to concentrate on densenet model since it is the simplest DenseNet among those designed over the ImageNet dataset.By comparing the figures on DenseNet-121. The measures under each volume represent the sizes of the width and depth, whereas the numbers on top represents the feature maps dimension.

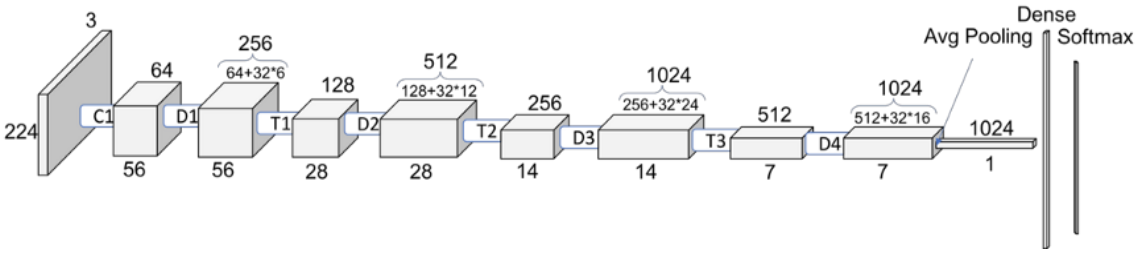


Figure 19:Another look at Dense-121. Dx: Dense Block x. Tx: Transition Block x

To demonstrate the bypass connections it is essential to note about how the first number of the addition to calculate the feature maps of every new volume matches the feature maps dimension of the previous volume. Therefore, it is a good explanation for the bypass connection because it

precisely means that they were concatenating (concatenate means add dimension, but not add values) new information to the previous volume, which was being reused.

Note that 32 is precisely the growth rate .However, the volume after every Dense Block increase by the growth rate times the number of Dense Layers within that Dense Block.

The below figure (**Figure 20**) showed deeply to explain what is actually happening inside every block.

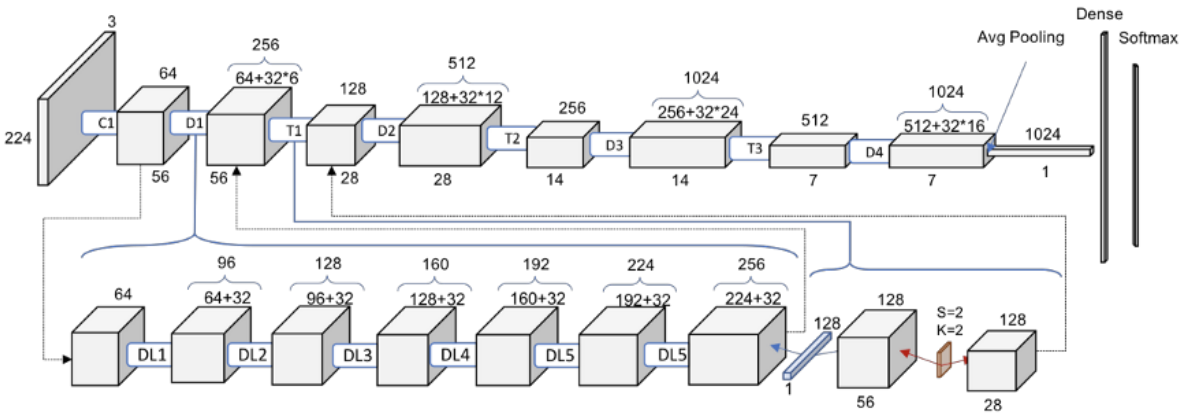


Figure 20:One level deeper look at DenseNet-121. Dense Block and Transition Block. DLx: Dense Layer x

3.2.2.5 Dense Layers

Every layer is adding to the previous volume these 32 new feature maps. That is why feature maps might go from 64 to 256 after 6 layers. In addition, Transition Block performs as 1×1 convolution with 128 filters. followed by a 2×2 pooling with a stride of 2, resulting on dividing the size of the volume and the number of feature maps on half. As a summary, the volume within a Dense Block remains constant, and the volume and the feature maps are halved after every Transition Block.

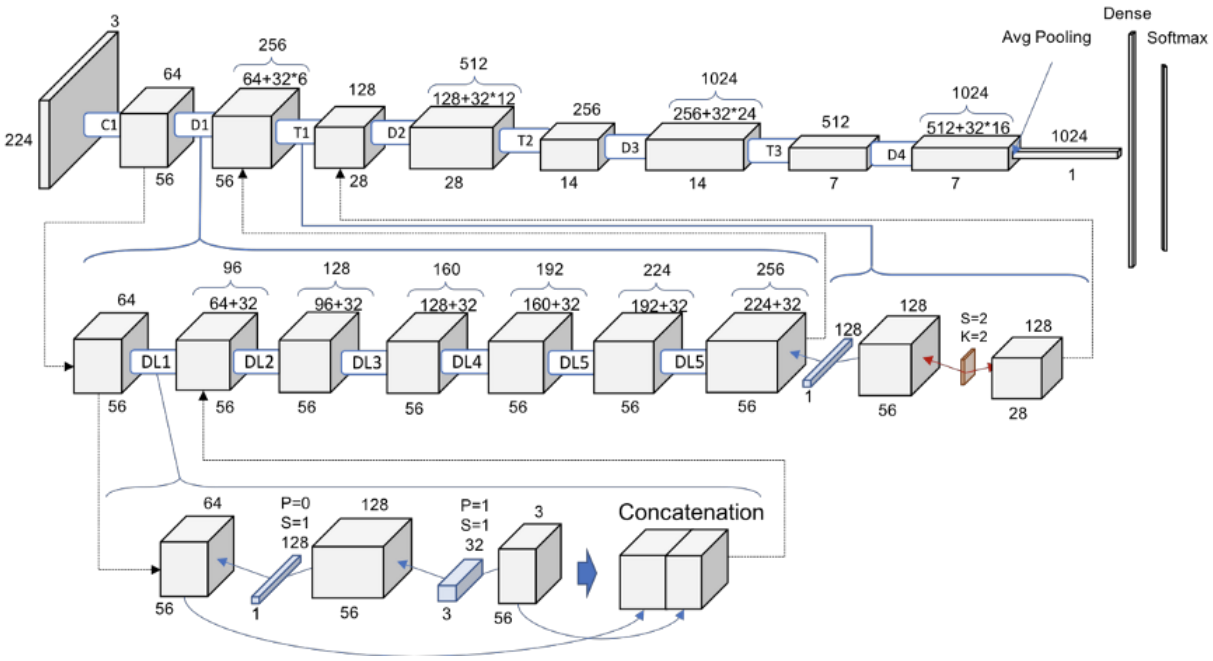


Figure 21:level deep. Full schematic representation of Densenet-121

In the new deeper level representing the first Dense Layer within the first Dense Block as shown in the figure above . It is obvious how actually this behavior of adding 32 times the number of layers is achieved.

As a suggestion, a 1×1 convolution with 128 filters to reduce the feature maps size and the perform a more expensive 3×3 convolution (remember to include the padding to ensure the dimensions remain constant) with this chosen 32 number of feature maps of growth rate.Then, the input volume and the result of the two operations (which are the same for every Dense Layer within every Dense Block) are concatenated, in the action of adding new information to the common knowledge of the network.

3.2.3 Inception

Inception v3 is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for Googlenet. It is the third edition of Google's Inception Convolutional Neural Network, originally introduced during the ImageNet Recognition Challenge. The design of Inceptionv3 was intended to allow deeper networks while also keeping the number of parameters from growing too large: it has "under 25 million parameters", compared against 60 million for AlexNet.

3.2.3.1 Inceptionv3 Architecture

The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:

1. Factorized Convolutions: this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network.Also, it keeps a check on the network efficiency.

2. Smaller convolutions: replacing bigger convolutions with smaller convolutions definitely leads to faster training. As an example a 5 × 5 filter has 25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3×3 + 3×3) parameters instead.

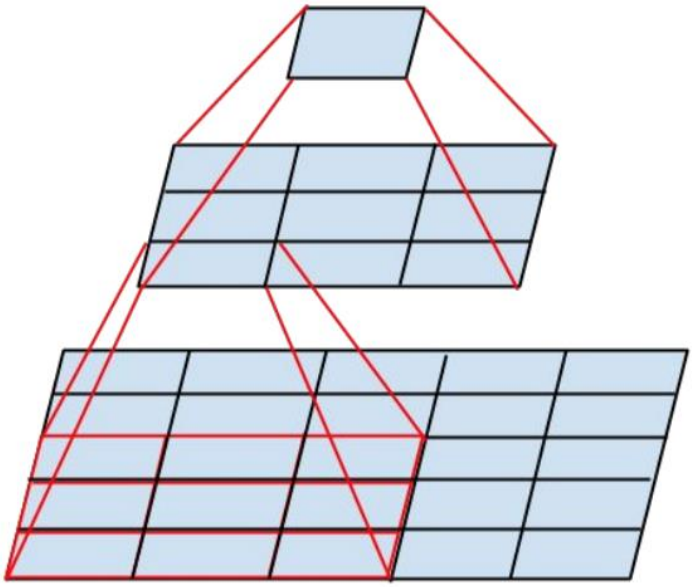


Figure 22:Inceptions' convolutions

The middle of the figure above presented a 3×3 convolution, and below it illustrated a fully-connected layer. Since both 3×3 convolutions could share weights among themselves, the number of computations could be reduced.

3. Asymmetric convolutions: a 3 × 3 convolution could be replaced by a 1 × 3 convolution followed by a 3 × 1 convolution. If a 3 × 3 convolution is replaced by a 2 × 2 convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed.

Figure 23:Inceptions' Asymmetric convolutions

4. Auxiliary classifier: an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In GoogLeNet auxiliary classifiers were used for a deeper network, whereas in Inception v3 an auxiliary classifier acts as a regularizer.

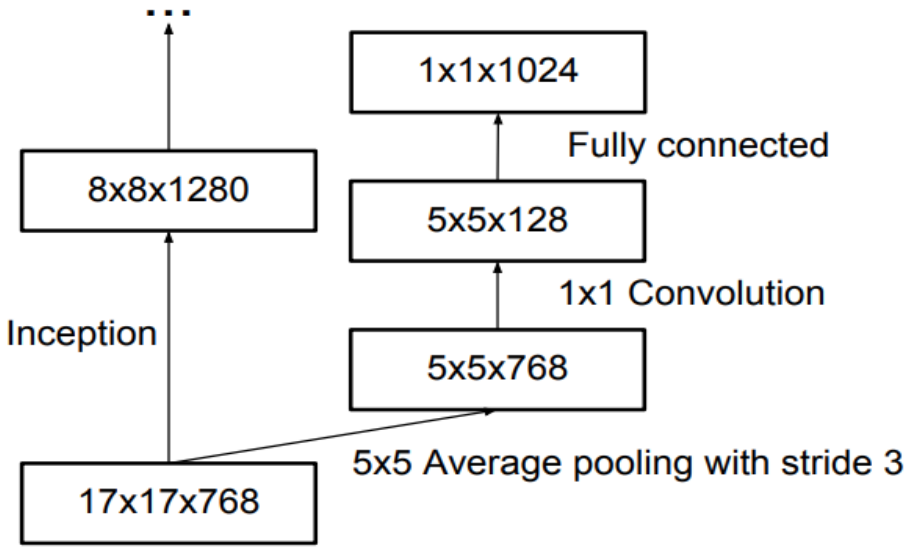


Figure 24:Inception's auxiliary classifier

5. Grid size reduction: Grid size reduction is usually done by pooling operations. However, to combat the bottlenecks of computational cost, a more efficient technique is proposed:

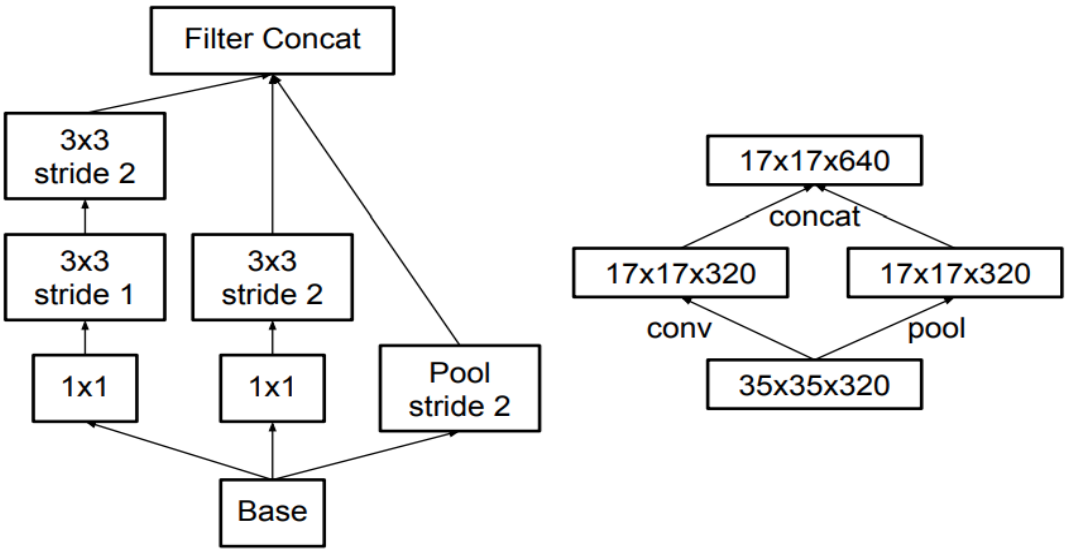


Figure 25:Inception's grid size reduction

All the above concepts are consolidated into the final architecture.

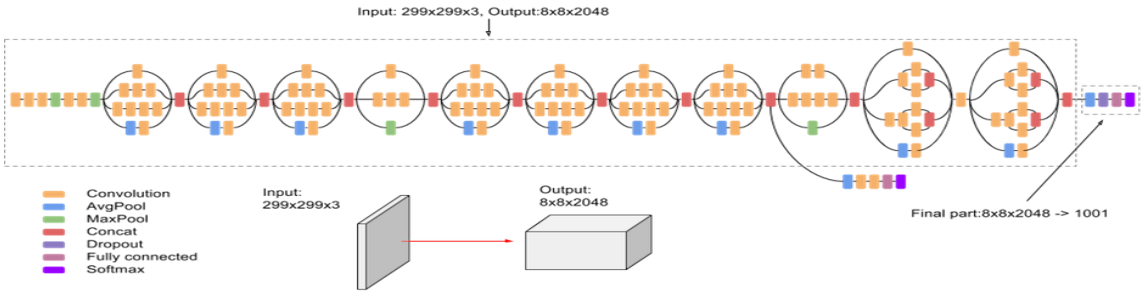


Figure 26:Inceptionv3 architecture

3.2.3.2 InceptionV3 Training and Results

Inception v3 was trained on ImageNet and compared with other contemporary models, as shown below.

Network	Top-1 Error	Top-5 Error	Cost
GoogLeNet	29%	9.2%	1.5
BN- GoogLeNet	26.%8	-	1.5
BN-Inception	25.2%	7.8%	2.0
Inception-v3-basic	23.4%	-	3.8
Inception-v3-rmsprop RMSProp	23.1%	6.3%	3.8
Inception-v3-smooth Label Smoothing	22.8%	6.1%	3.8
Inception-v3-fact Factorized 7*7	21.6%	5.8%	4.8
Inception-v3 BN-auxiliary	21.2%	5.6%	4.8

Table 3: InceptionV3 Training and Results in addition to comparison to other competitors models

As shown in the table above, when augmented with an auxiliary classifier, factorization of convolutions, RMSProp, and Label Smoothing, Inception v3 can achieve the lowest error rates compared to its contemporaries.

3.3 Preparing of datasets

Previously, in the first project I have trained 3 CNN models, which are Inceptionv3, Resnet50, and Densenet121 for a COVID-19 dataset that classified to 2 classes, which are COVID -19, and non-COVID -19 cases. The dataset contains 12194 AP x-ray images. However, AP is an X-ray picture in which the beams pass from front-to-back (anteroposterior). As opposed to a PA (posteroanterior) film in which the rays pass through the body from back-to-front.the contents of that latter dataset have been classified into 3 classifications (i.e., training, testing , and validation), each one has categorized to both COVID -ve (class1) and COVID +ve (class0), the real number of images that used in each of them shown in the following table :

Dataset	COVID -ve (class1)	COVID +ve (class0)	Total
Training	5124	3456	8588
Testing	1170	633	1803
Validation	1170	633	1803

Table 4: Dataset generation and classification

As shown above, first, the consolidated images normalized and resized from their original shape 1048 × 1488 into 256 × 256, note that, the shaped images for normalization 255 × 255 image's size could use. Then the images are shuffled and split into training, test and validation data. The training data have 8588 .Thus,images where 5124 images are for class 1 (Negative) and 3456 images are for class 0 (Positive). The test data have 1803 images where 1170 images are for class 1 (Negative) and 663 images for class 0 (Positive). Similarly, the validation data have 1803 images where 1170 images are for class 1 and 663 images for class 0.The accuracy of the previously mentioned pre-trained models considered the latter dataset was roughly 90%. To ensure higher accuracy, I have modified the dataset that has been used, to be collected from different sources, Kaggle considered as the most important one of them. The accuracy of these labels is reported to be over 90% .The bit-depth of the radiographs in 16-bit Digital Imaging and Communications in Medicine (DICOM) format was reduced to 8-bit Portable Network Graphics (PNG) format without compression (without Joint Photographic Experts Group (JPEG) artifacts). At this stage, 256 times data loss was experienced.Furthermore, that dataset there were 10625 radiographs for COVID-19 case and 10192 radiographs for non-COVID-19 However, for testing and validating the models there were 2313 radiographs for COVID-19 case and 2313 radiographs for non-COVID-19 and 2313 for viral pneumonia or Pneumonia as short for training the models with Posteroanterior (PA) and Anteroposterior (AP) views. However, this database, which also includes contralateral and lateral view chest radiographs, is created to transform the data obtained from the articles and resources into a database for access to COVID-19 radiological data. Hence, it contains lossy-images taken from articles beside resources. In addition, it has been noticed that some of the properties such as brightness, contrast, sharpness are modified by the authors in most of the images. It can be said that the database may be biased due to such effects.

Dataset	COVID -19 (class0)	Normal COVID-ve (class1)	Total
Training	10625	10192	20817
Testing	2313	2313	4626
Validation	2313	2313	4626

Table 5:Modified dataset generation and classification

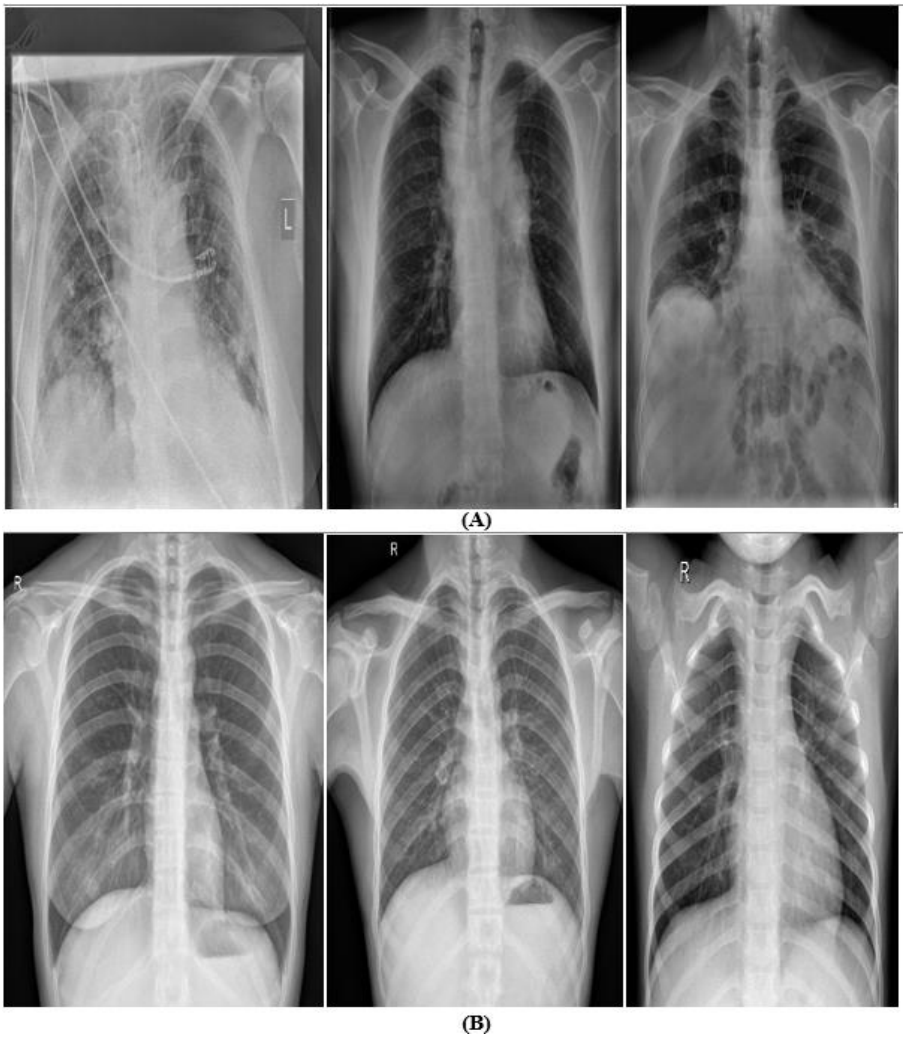
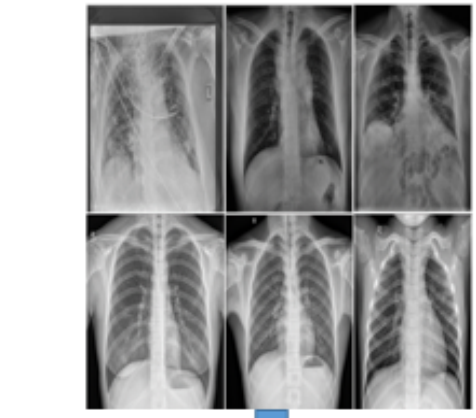


Figure 27:Example of the (A) COVID19, (B) Normal experimental X-ray images



Preprocessing:

- Scale the size of all images 244 x 244
- Data augmentation(flipping , resizing , and rotation)

Training Model:

- Split the dataset of 70% of the total images for training stage and 30% for testing
- Select the model of deep learning classifier

Tuning Process:

Using CNN models such as; ResNet50, DenseNet201, and Inceptionv3

Classification:

Identify the diseases from X-ray images

COVID +ve (COVID-19)

COVID -ve (Normal)

Figure 28:Methodology of pre-trained models

Since the previously trained Convolutional Neural Networks (CNN) networks to be used work with RGB (Red, Green, Blue) inputs, grayscale radiographs need to be colored with colormap (**Figure 29**). Although all the data used are 8-bit grayscale images, the main reason for applying colormap to the graphics is to create a custom dataloader ready to work with 16-bit DICOM formats. By this way, when DICOM files are used on the model, 16-bit data can be transferred to 24-bit RGB color space without loss. It is also expected that the coloring process may provide a solution to the probable bias problem. For this purpose, all radiographs are colored with the colormap named COLORMAP_JET in the OpenCV library and transferred to 24-bit space. Due to the few numbers of data and unbalanced classes in the dataset, all the radiographs were augmented randomly by horizontal and vertical flipping, rotating, zooming, changing brightness values, and warping (**Figure 30**). Since these augmentations also simulate different types of images created by different x-ray devices, it is a very crucial step to generalize the suggested model.

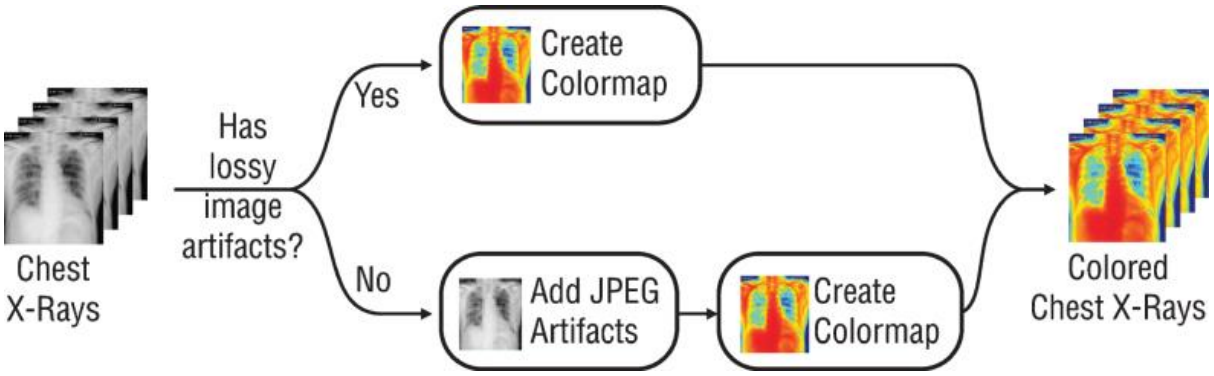


Figure 29:Schematic presentation of pre-processing step

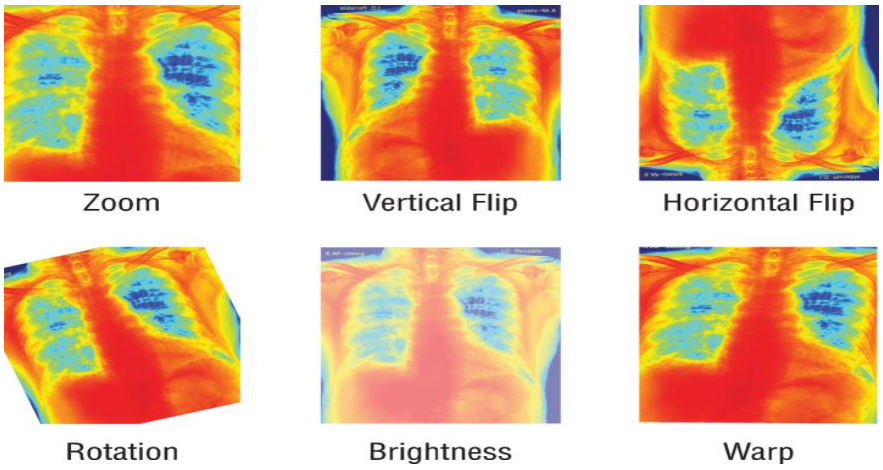


Figure 30:Schematic representation of the augmentation process

3.4 Standards

Transfer learning technique and CNN models were used in this report since there are few numbers of the available dataset. The transfer learning and optimization of the model was conducted on a personal computer (with a processor of 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz , the type is 64-bit operating system, x64-based processor , Installed RAM 16.0 GB). Fastai and OpenCV [19] libraries were used to develop the model.

3.4.1 Transfer learning and optimization

Residual Neural Network (ResNet) [20] , Dense Convolutional Network (DenseNet) [21] , and Inceptionv3 [22] architectures, which have already been trained by ImageNet database, and therefore already able to classify in 1,000 different classes, were re-trained to distinguish COVID-19 pneumonia and non-COVID-19 pneumonia by transfer learning. It is an advantage to use pre-trained architectures to enhance training speed and accuracy of the new proposed model, since the main image features that have already been learned can be transferred to the new task.During transfer learning studies, the convolution layers of ResNet18, Resnet34, ResNet50 (for special), ResNet101, ResNet152, DenseNet121(for special), DenseNet161, DenseNet169, DenseNet201, and Inceptionv3 architectures, which have previously trained by ImageNet to achieve the accurate results with few datasets, were frozen and only fully connected layers were re-trained. The numbers in the name of architectures such as 34, 50, 152, and 121, etc. represents the number of weight layers within the respective neural network.

As mentioned descriptively before, the main base element of ResNet is the residual blocks. In ResNet, all these residual blocks are stacked together very deeply.due to that , it prevented the distortion that occurs as the network gets deeper and more complex. In addition, bottleneck blocks are used to make training faster in the ResNet model. Another thing with this very deep architecture is that it is enabling up to 150 layers deep of this. The number of filters can be doubled and downsample spatially using stride two in ResNet.DenseNet is densely connected CNN where each layer is connected to all previous layers. Therefore, it forms very dense connectivity between the layers. The DenseNet consists of several dense blocks, and the layer between two adjacent blocks is called transition layers. The former define how the inputs and outputs are concatenated, while the latter control the number of channels so that it is not too large. DenseNet uses the modified “batch normalization, activation, and convolution” structure of ResNet .The inception especially inceptionv3 model considered as an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset .

However, the model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax.

3.4.2 Hyperparameter optimization

The learning process and the structural structure are controlled by several hyperparameters, which may be classified as either structural or algorithmic hyperparameters . The structure and topology are characterized by structural hyperparameters, which include the number of layers of the network, the number of neurons in each layer, the degree of connection, the neuron transfer function, and others. They alter the network's structure, which affects the effectiveness and computational complexity. The learning process is driven by algorithmic parameters, including the size of the training set, the training method, the learning rate, and other factors. Although these variables do not belong to the neural network model and do not affect how well it performs, they do affect how quickly and effectively the training step goes.

A machine learning model's hyperparameter settings are a predetermined set of choices that directly affect the learning process and the output of the prediction, which shows how well the model works. Model training is the process of instructing a model to find patterns in training data and predict the outcome of incoming data based on these patterns. In addition to the hyperparameter selections, model architecture, which reflects the model's complexity, has a direct bearing on how long it takes to train and test a model. The setting has become a crucial and challenging problem in the application of ML algorithms due to their influence on model performance and the fact that the ideal collection of values is unknown. In the literature, there are several methods to adjust the hyperparameters.

Manual search is one way to improve these hyperparameters. This may be used when the researcher has a solid understanding of neural network structure and learning data since it determines the hyperparameter value based on the researcher's intuition or skill. However, the standards for choosing hyperparameters are ambiguous and require several experiments.

To choose the ideal hyperparameter values for ML algorithms, designs of experiment (DOE) methods are utilized. DOE evaluates the effects of several experimental components simultaneously, with each experiment consisting of many experimental runs at various hyperparameter values that should be evaluated collectively. The experimental data are statistically examined when the tests are finished to ascertain how the hyperparameters affect

the performance of the classifiers. In other words, a model that empirically connects classification performance to hyperparameters, such as prediction errors (as a response variable) (as predictors of classifier performance).

In the domain of DL, it is established that a technique is trained straight from the data in an end-to-end way, meaning that time-consuming manual feature extraction is not necessary from the human (domain experts). However, the model selection procedure in deep learning requires significant human work. Discovering the hyperparameter settings that result in the optimum performance is the first step in this procedure. The best hyperparameter values can typically be found using one of three methods: manually built on previous knowledge; arbitrarily selected from a set of candidate hyperparameter values; or in-depth grid search. Based on previous research, the authors applied the manual method in this project. The learning rate is reduced when it is noticed during the model training that there is no improvement in the validation training value. The learning rate is set to be reduced every 10 epochs.

The proposed CNN architecture includes several hyperparameters. These hyperparameters should be carefully selected because they control the performance of the suggested technique. The details of the hyperparameter settings are given in **Table 6**. The study experimentally finds that these are the best suitable values of hyperparameters for the proposed Inception V3 model and other pre-trained networks for this application.

Hyperparameters						
DTL(s)	**Optimizer**	**Learning Rate**	**Batch Size**	**Epochs**	**Dropout**	**Activation**
InceptionV3	Adam	0.0010	32	50	0.5	Relu
Resnet50	Adam	0.0002	32	50	0.5	Relu
Densenet121	Adam	0.0003	32	50	0.5	Relu

Table 6:Details of the hyper parameter settings

3.4.3 Activation maps

Gradient-weighted Class Activation Mapping (Grad-CAM) method helping to improve causality and intelligibility was used to create activation maps that highlights the crucial areas of the radiograph [23 – 25]. .In the Grad-CAM method, the gradients of the radiographs flowing into the final convolutional layer to produce a rough localization map in which the important areas are highlighted is used. It has been reported that the deeper presentations in a

convolutional neural network capture upper-level visual constructs [26, 27].Additionally, the convolutional layers able to keep spatial information which is lost in fully-connected layers. Thus, it can be expected the last convolutional layers to have the perfect concurrence between high-level semantics and detailed spatial information. The semantic class-specific information in the image are tried to find by the neurons (i.e. object parts). Grad-CAM uses the gradient information flowing into the last convolutional layer to assign significance values to each neuron.

3.4.4 Performance evaluation criteria

The accuracy, recall (sensitivity) and precision (positive predictive value) parameters are the main parameters to determine the performance of the model. In confusion matrix there is four possible outcomes including true positive (TP), true negative (TN), false positive (FP), false negative (FN). When the case is actually positive, if it is classified as positive it is labeled as TP, in contrast, if it is classified as negative it is labeled as FN. Similarly, in the case of negative, if the case is classified as negative it is TN, if is classified as positive it is labelled as FP [28] . To evaluate the performance of the tailored convolutional neural networks the accuracy, precision, recall ,and F1 score coefficients were determined as shown in the following equations (**Eq.14** , **Eq.15** , **Eq.16** , and **Eq.17**) respectively:

$$\textbf{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

$$\textbf{Precision} = \frac{TP}{TP+FP}$$

$$\textbf{Recall} = \frac{TP}{TP+FN}$$

$$\textbf{F1-score} = 2 * \frac{Precision \times Recall}{Precision+Recall}$$

A confusion matrix is a performance measurement technique for summarizing the performance of a developed deep learning classification model. Calculating a confusion matrix provides a better representation of what classification model is getting right and what types of errors it is making. It also ensures insight not only into the errors being made by the classifier but more importantly the types of errors. It is a matrix including the different combinations of predicted and actual values.

3.5 Process

The training process consists of three basic stages as followings :

i) Different architectures in various depths including ResNet, DenseNet , and inceptionv3 were examined and all convolutional layers were frozen and trained

ii) The most appropriate architecture was determined and fine-tuning process was performed for each architecture

iii) The inference results of the activation map were evaluated, which establishes a bridge between radiologists and artificial intelligence for understanding.

3.5.1 Transfer learning (Model Uncertainty)

Deep learning techniques have received a great deal of attention in practical machine learning. Such methods for regression and classification do not, unfortunately, account for model uncertainty. In comparison, Bayesian models that are method of statistical inference (named for English mathematician Thomas Bayes) that allows one to combine prior information about a population parameter with evidence from information contained in a sample to guide the statistical inference process. A prior probability distribution for a parameter of interest is specified first. The evidence is then obtained and combined through an application of Bayes's theorem to provide a posterior probability distribution for the parameter. The posterior distribution provides the basis for statistical inferences concerning the parameter. offer a mathematically sound framework for evaluating model uncertainty and often have exorbitant processing costs. At the software level, the effectiveness of the model cannot be easily quantified as accuracy. To demonstrate how certain it was the detection and the classification, I, therefore, add a new indicator: the confidence score. A confidence score is a great tool to quantify uncertainty. The model in this report is a non-Bayesian network. For estimating uncertainty, Monte Carlo Dropout (MC Dropout) which is a computationally inexpensive and powerful stochastic regularization technique that performs well on real-world datasets and has been shown to be equivalent to performing variational inference , and Deep Ensembles are the two primary non-Bayesian approaches. One of the most widely used methods to avoid overfitting is the dropout technique. With MC Dropout, the dropout layer is used during the training and then testing phases, and many predictions are made on a single image to calculate the degree of uncertainty. I decided to use MC Dropout for this project. It uses smaller hyperparameters and uses fewer processing resources .

There are only two dropout layers in the study since the dropout layer was added to the model after each fully connected (FC) layer. The dropout layer is often implemented throughout the

training procedure to prevent overfitting. The dropout will be automatically performed during the analysis process to guarantee consistency in the prediction outcome for the same image. The dropout layer must be activated in MC Dropout's prediction phase so that each prediction's softmax value changes, affecting how it is classified. The following stage involves making 100 predictions with each target image, with the majority of the estimates' findings serving as the categorization for the subsequent forecasts. The proportion of the confidence score will be determined by the number of forecasts. If the confidence score value is less than a predetermined threshold, such as 70%, none of the positive and negative forecasts will be greater than 70 times in a sample of 100 predictions. It was believed that this circumstance is difficult to anticipate and calls for accurate manual processing.

The study initially chose the ideal MC dropout rate to quantify the uncertainty of the model. The dropout rate should be balanced so that it is neither too high nor too low. The estimated confidence intervals for the distribution will be excessively large if the dropout rate is too high since this will result in a very diversified predictive distribution. If it is too small, the confidence intervals will be narrow, and the forecast distribution will be too similar. It was conducted experiments to determine the ideal dropout rate for this application, which is 0.452. Finally, the confidence interval, standard deviation (SD), and entropy may be used to calculate the uncertainty of the model.

Chapter 4: Results and Analysis

Here, the study evaluates the efficiency of the recommended technique and contrasts it with cutting-edge approaches. To assess the performance of our suggested model, there are three different cutting-edge applied models, including the inceptionV3, resnet50, and densenet121 models on the posture dataset.

This section has six parts. In **section 4.1**, the implementation authors first provide the settings and assessment methods. In **section 4.2**, it compared the performance of the suggested technique with many current-generation models that are often cited in the literature. In addition, **section 4.3** provides information on how well the suggested model performs in detection settings.**Section 4.4** represent the practical implementation of the project (i.e. The prototype tool, the constraints and opportunities of this project will be illustrated with more details in **sections 4.5 and 4.6** respectively.

4.1 Implementation Settings

To make the evaluation more authentic, I used the same data set to implement three existing techniques, including the conventional inceptionV3. The workstation used for the

implementation of this study is as follows: hp laptop, Intel Core i7 with 16 GB of RAM. The visual studio (VS) that is a Microsoft Integrated Development Environment (IDE)was used with the TensorFlow application.

All necessary libraries were imported into the Jupyter environment, after which the dataset was uploaded, and their width and height sizes were resized to 224 × 224. The images were normalized by making sure that all numeric values are in the same range between 0 and 1, and this helps the large values not overwhelm the smaller values. The normalization function receives an array as an input, uses a formula to normalize the array's values in the range of 0 to 1, and outputs the normalized array. The data set was divided into 20817 for training, 6939 for validation, and 6939 for testing (already explained in Section 2.1). The next step was the introduction of image-augmentation techniques, which are rotation_range (20), width_shift (0.3), height_shift (0.3), shear_range (30), zoom_range (0.2), and horizontal_flip (0.2). The L1 and L2 regularization was then defined and implemented, after which the model was defined for training, validation, and testing of the data set. A batch size of 32 and epochs of 50 were used for model training. The early stopping technique was called and set to avert the model from overfitting. The LR was also set to reduce when there is no improvement in the metrics or the performance is stagnant, this will assist to improve the model metrics.

4.2 Performance Evaluation

The study offered the results for the execution and the suggested model's training performance was assessed in terms of crucial metrics, such as training accuracy, validation accuracy, training loss, and validation loss at 50 epochs for the suggested models and the three cutting-edge models. The learning rates of 0.0010, 0.0007, 0.00049,and 0.00034 were optimized with adaptive moment estimation (Adam). According to **Table 7** , the suggested model achieved better results with an LR of 0.0010 and an Adam optimizer. These variables are generated to evaluate trained models with an Adam-optimized learning rate of 0.001. These parameters are calculated to estimate the excess fit of the trained model. The graphs of the training loss/validation loss and the training accuracy/validation accuracy of the proposed model and the baseline models are shown in **Figure 31**. Furthermore, the test data set was used for the testing process, and the testing loss and accuracy can be seen in the table. A confusion matrix was also produced for all the models implemented to calculate performance metrics such as precision, recall, f1 score, and accuracy. Each model parameter used for the training and validation of the model is shown in **Table 8**. The results of the models in the training and validation data set are shown in **Table 9**. **Table 10** displays the results of the proposed models

In the test dataset and each class consists of 240 instances. It can also be shown in **Table 10** that the proposed InceptionV3 and DenseNet121 outperform the other baseline models with an ACU of 0.99.

Model	Learning Rate	Epochs	Early Stopping	Loss	Optimizer	Batch Size
InceptionV3	**0.00024**	**50**	**Epoch 45**	**CategoricalCrossentropy**	**Adam**	**32**
ResNet50	**0.00024**	**50**	**Epoch 50**	**CategoricalCrossentropy**	**Adam**	**32**
DenseNet121	**0.00034**	**50**	**Epoch 40**	**CategoricalCrossentropy**	**Adam**	**32**

Table 7:Parameters Used for model training

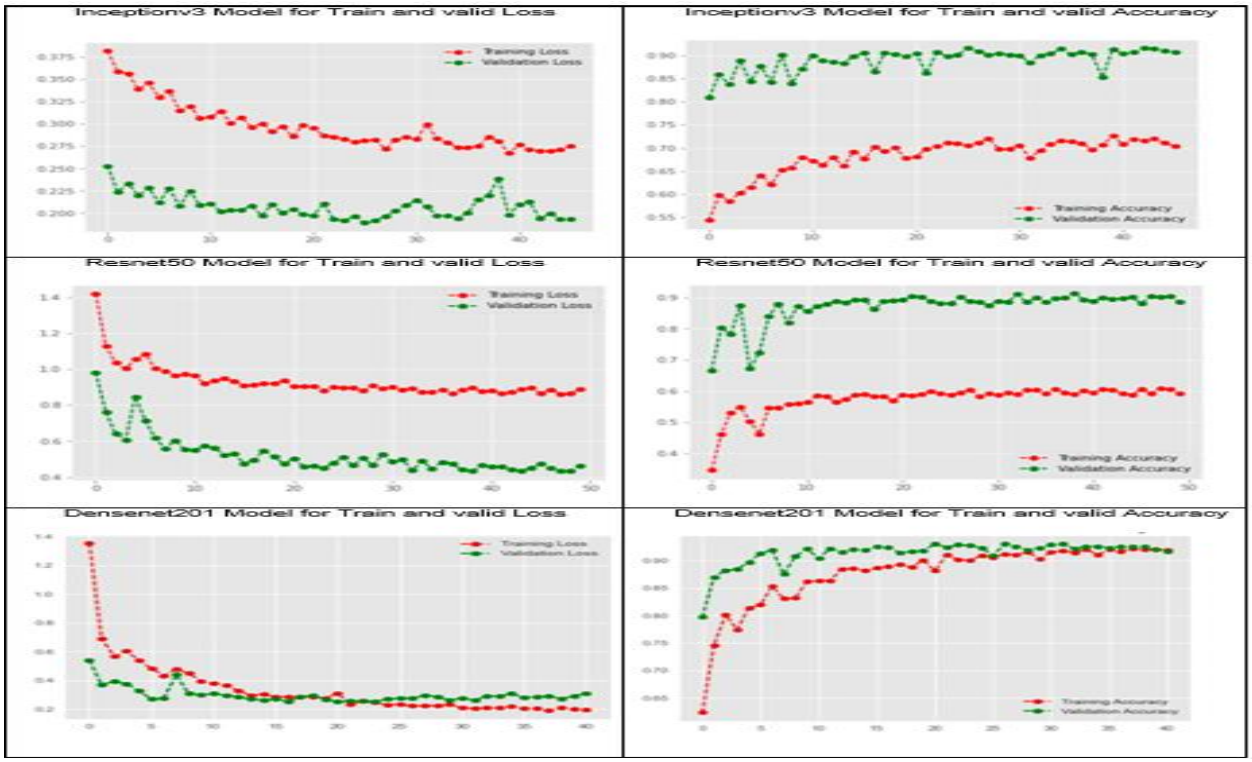


Figure 31: Loss and accuracy graphs of DCNN models: InceptionV3,ResNet50, and DenseNet121 respectively

Model	Model Parameters
InceptionV3	Total params: 24,179,236 Trainable params: 2,376,452 Non-trainable params: 21,802,784
ResNet50	Total params: 30,158,468 Trainable params: 6,570,756 Non-trainable params: 23,587,712
DenseNet121	Total params: 9,151,812 Trainable params: 2,114,308 Non-trainable params: 7,037,504

Table 8:Model Training Parameters

Model	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Training Loss	Validation Loss	Testing Loss
InceptionV3	70.38	90.76	89.58	0.28	0.19	0.21
ResNet50	80.18	88.67	88.44	0.29	0.21	0.24
DenseNet121	98.89	96.67	97.29	0.19	0.20	0.13

Table 9:Results of the models on the training and validation data set

Model	Accuracy	AUC	TP
InceptionV3	0.90	0.96	860
ResNet50	0.88	0.96	849
DenseNet121	0.95	0.99	886

Table 10:Results of the models on the test data set

Confusion matrix is a performance measurement for machine learning classification problem where output contains two or more classes. The confusion matrices for each model with

different architecture were calculated at the points where the models perform at maximum to obtain a better idea of the performance of the developed models. The confusion matrices calculated for both validation dataset and testing datasets were depicted in the figure below . The results revealed that the DenseNet-121 architecture performs better than other architectures with the following graphs (**Figure 32**).

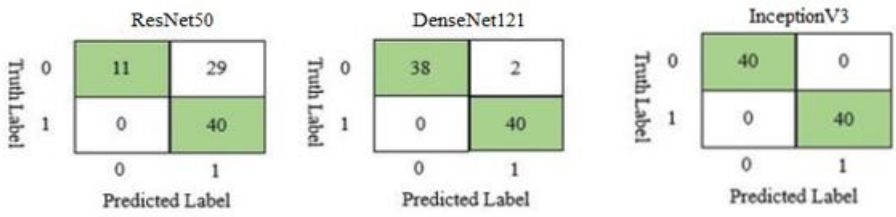


Figure 32:Confusion matrix of different deep learning model for chest X-ray image dataset

4.3 Computational Time

In this experiment, Google Colaboratory is utilized, which provides the Tesla K80 GPU platform. It is observed from **Table 11** that the ResNet50 series demands approximately 27 seconds per period for training, However, according to studies other versions of ResNet like ResNet101, ResNet101V2, ResNet152V2 needs about 33 seconds per epoch, which means they are much slower than ResNet , due to the increment within their layes,which means the depth of these models. In fact , ResNet152 takes the longest time to train per epoch, which is around 39s.InceptionV3 has an average training time of almost 28 seconds per epoch Though the networks demand quiet time for training, the testing time is less, which is around 5 seconds

Classifier	Training time (per epoch)
DenseNet121	22 s
DenseNet169	31 s
DenseNet201	32 s
ResNet50	27 s
ResNet50V2	27 s
ResNet101	34 s
ResNet101V2	32 s
ResNet152	39 s
ResNet152V2	33 s
InceptionV3	28 s

Table 11:Training time per epoch of individual deep TL models

4.4 The Prototype Tool

In the COVID-19 epidemic, radiological imaging plays an important role in addition to the diagnostic tests performed for the early diagnosis, treatment, and isolation stages of the disease. Chest radiography can detect a few characteristic findings in the lung associated with COVID-19. Deep learning models are sensitive in detecting COVID-19 lung involvement and hence the diagnostic accuracy rate is high. During the evaluation of the model, X-ray radiographs of COVID-19 patients confirmed positive by the PCR Test are used. The model can easily detect Ground-glass opacity (GGO), consolidation areas, and nodular opacities, which are the pathognomic findings of patients for COVID-19 on X-ray radiography. In COVID-19, bilateral, lower lobe, and peripheral involvement observed, and the proposed model can detect localization of the lesion. These models are particularly important in identifying early stages of COVID-19 patients. Early diagnosis of the disease is important to provide immediate treatment and to prevent disease transmission. The models can also play an indispensable role in patients lacking early symptoms. There is a margin of error in patients with diffuse late lung parenchyma and in patients with significantly reduced lung ventilation due to poor quality X-ray images. X-rays that are not of optimal quality are difficult to evaluate by radiologists. The clinical and radiological images of later-stage patients are well established and it is easier to detect the findings by experts. The role of deep learning models is more prominent in screening and diagnosis when the infection is at its early stages.

Based on the proposed solution, a simple desktop tool for the detection of COVID-19 positive and negative cases has been developed. This allows any medical personnel to browse a chest X-ray image and feeding it to the application. The application, in turn, will execute the model proposed in this work and provide the label for the given Chest X-Ray image. As a result, this will detect the COVID +ve and COVID –ve cases along with their probabilities as shown in **Figures 35 and 36**. In fact, this could be used on platforms like Windows, Mac, and Linux. This interface could be used in any COVID-19 testing centers or other health facilities for the fast detection of the disease.

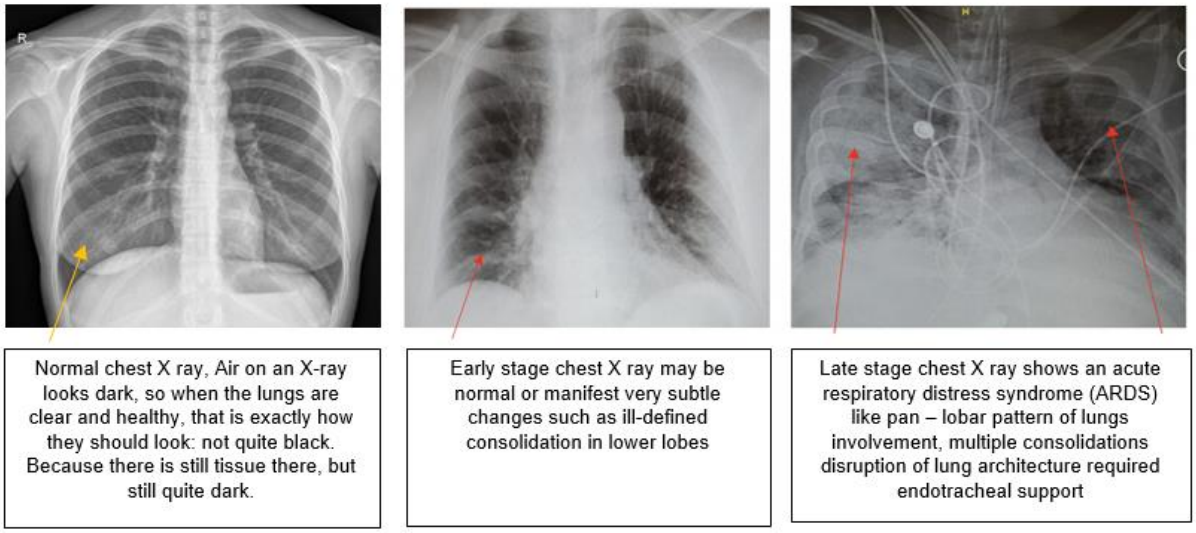


Figure 333:Differences observed by the radiologist between some COVID and pneumonia case images

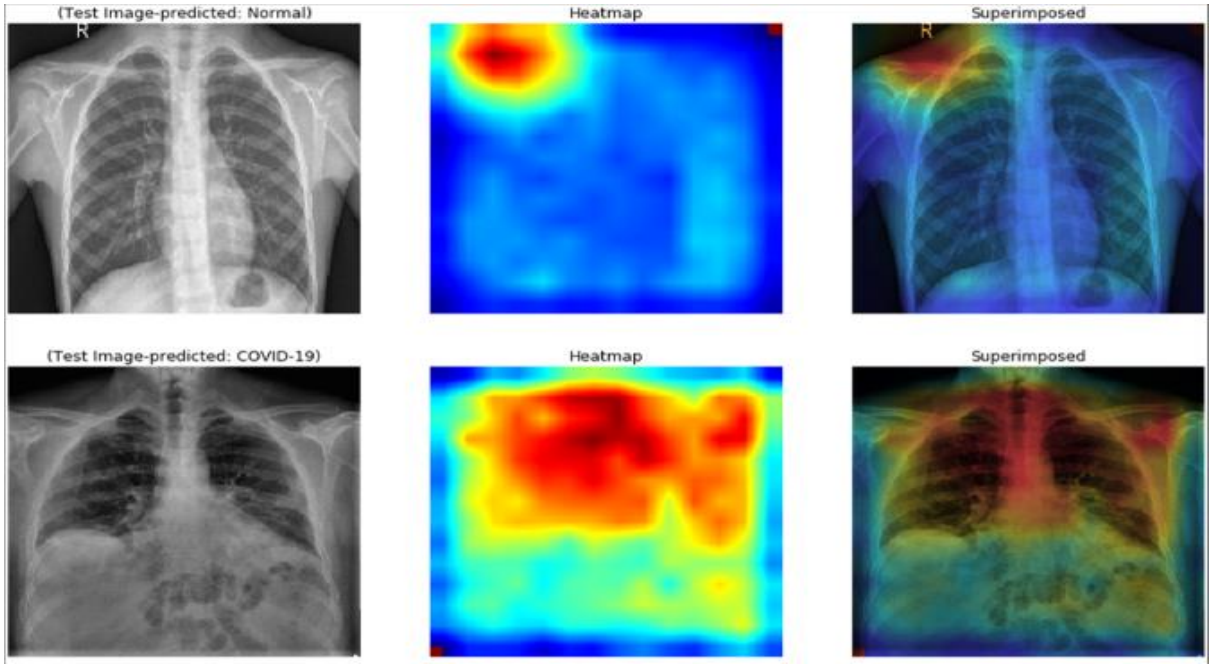


Figure 34:Visualization on chest x-ray of normal and COVID -19 infected using Grad-CAM on the proposed models

Figure 35:GUI-based tool for COVID-19 detection (Positive case (+ve))

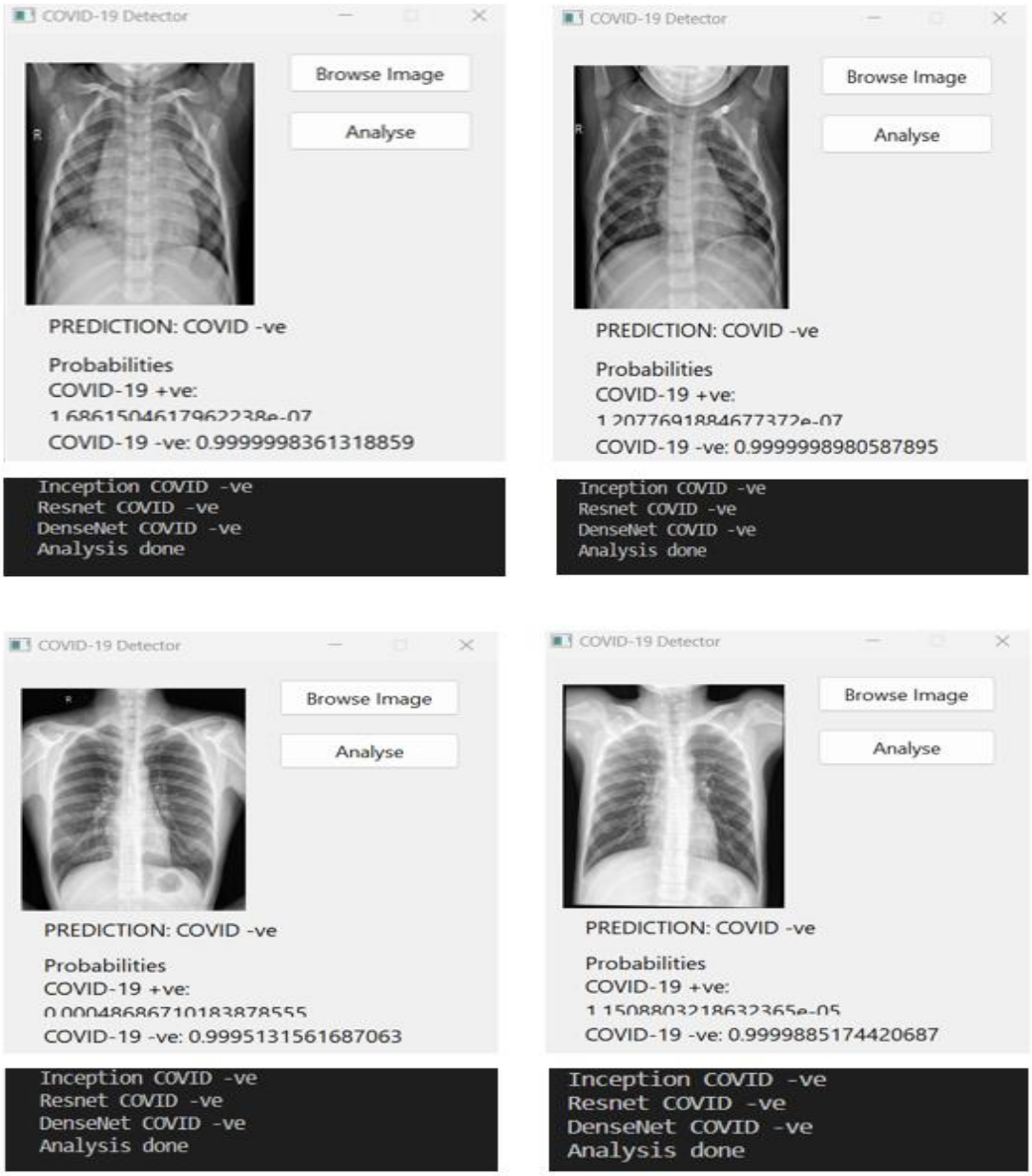


Figure 36 :GUI-based tool for COVID-19 detection (Normal or Negative case (-ve))

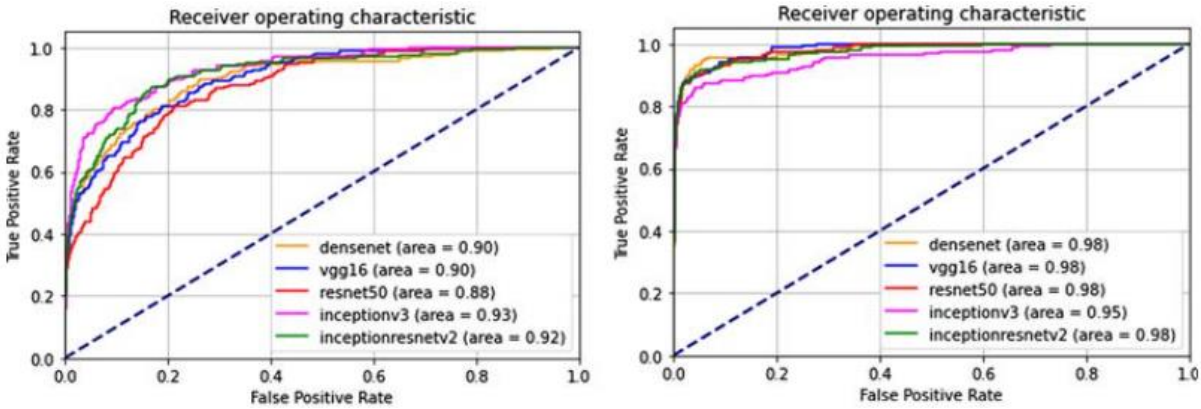


Figure 37:ROC (receiver operating characteristic)curves and AUC values for different implementations in classifying COVID-19 CXR images based on different CNNs

4.5 Constraints

In the last section, I have discussed general several works or studies on COVID-19 image analysis. Although the performance of the proposed algorithms seems promising, there are certain shortcomings that must be addressed. Now, it has been presenting a discussion on some of the challenges and gaps in this area.

4.5.1 Reproducibility and code availability

Reproducibility of DL-based models has emerged as one of the major challenges in the literature. Results can be ascertained if only the dataset and the details of the model architecture and training hyperparameters are made available. In addition, the open-source availability of code helps in reproducing the results and in devising further improvements. Some of the works based on CXR and CT-scan image classification have provided their codes [29,30,31, 32,33,34,35,36,37,38,39,40,41,42,43,44,45].

4.5.2 Unbalanced dataset

It is noted that most of the dataset used for binary class or multiclass classification for COVID-19 diagnosis is highly unbalanced. The skewness in the dataset can introduce bias in the performance of a trained model. These unbalanced datasets pose a major challenge to the AI researchers because the collection of a sufficient number of quality images, especially at the initial stage of the pandemic, was indeed difficult. Although, In Refs. [46,40], random sampling was used to select a balanced multiclass dataset from an unbalanced larger dataset. However, this method reduced the size of the dataset. In Ref. [47], data augmentation, weighted-class batch sampling, as well as stratified data sampling were used to obtain an equal number of samples for each class in every training batch. In Ref. [48], dataset balancing was carried out using the Synthetic Minority Oversampling Technique (SMOTE), which is a machine learning technique that solves problems occurred when using an imbalanced dataset. Authors in Ref. [49] addressed the class-imbalance problem owing to the limited set of COVID-19 CXR images by proposing a novel transfer-to-transfer learning approach, where a highly imbalanced training dataset was broken into a group of balanced minisets, followed by transferring the learned (ResNet50) weights from one set to another for fine-tuning.

4.5.3 Data augmentation

Data Augmentation is employed to increase the size of the training dataset by transforming the images of the original dataset in multiple ways. This enables the learning of diverse features

during the training process and reduces the overfitting of the model. Two important techniques of data augmentation have been observed. First, several variations such as flip, rotate, skew, translation, random clipping, tilting, and scaling have been introduced to the original dataset, increasing the number of training samples. Second, inbuilt software libraries (e.g., Keras ImageDataGenerator function) have been utilized that introduce random variations in the training dataset during each iteration without increasing the number of samples. The range of random variations is a hyperparameter that needs to be fine-tuned for a given problem. Authors in Refs. [50,51 ,52, 53, 54,55,38,56, 57, 58,59,60,61,42,62,63,64] have used the first method, while the authors in Refs. [65,47,66,33,46,67,68,69] have used the second technique of data augmentation.

4.5.4 Quality of images

Medical images are generally low contrast images. Hence, efforts are made to increase the contrast of these images so that the images are better transformed to the feature space while they traverse through a deep learning (DL) model. Moreover, broad heterogeneity in the quality of images captured at different sites using different imaging devices causes' potential bias in image analysis. This challenge emphasizes the need for improving image quality as a pre-processing step. Contrast enhancement techniques are generally used in for enhancing the quality of images and making them visually more appealing. A few studies carried out histogram modification of images for contrast enhancement [70,38,56]. Authors in Ref. [64] utilized local contrast enhancement on thresholded grayscale CXR images for enhancement and also for removing any text from the image

4.5.5 Transfer learning architecture

Transfer learning has been used as either a fixed feature extractor (where the weights of the convolutional layers of the pre-trained architectures are used without alteration) or weights of the few or all convolutional layers of the model are fine-tuned or retrained. The choice of an approach depends upon the size and similarity of the training dataset of the given problem to the dataset used in the training of the original transfer-learning model. Since weights of most of the standard DL models (used for transfer learning) are learned over 1000 classes of the ImageNet dataset consisting of natural images, these DL models may not be completely relevant for the classification of CT or CXR images. Hence, it is recommended to employ transfer learning by retraining the weights of a few convolutional layers. Several studies in papers[50,71,72,52,73,53,74,75,56,76,59,77,78,40,79,80,42,81,63,82,48,83,44,84,45] have

used transfer learning models as fixed feature extractor only. Also, it is important to note that very few studies such as class decomposition with VGGNet [74], attention with VGGNet [43], feature pyramid network with ResNet [50] have proposed architectural changes in the proposed model that is very much required not only to achieve better classification capability but also to have faster and stable learning.

4.5.6 Performance learning curves

Training and validation curves of accuracy and loss function provide a visual assessment of the three aspects of the training/trained model. First, it indicates how rapidly the model learns the objective function in terms of the number of iterations. Second, it informs how well the problem has been learned in terms of under fit/over fit/good fit of the model. Under fitting is shown by the low training accuracy, while overfitting of the model is indicated by a substantial gap between the training and validation curves. The good fit of the model is represented by higher training accuracy and convergence between training and validation curves. Third, there could be random fluctuations or noisy behavior in the training/validation loss curves. This could be due to a number of reasons, including the small size of the dataset compared to the model capacity, need of regularization, feature normalization, etc.

4.5.7 Stratified K-fold cross-validation

When the dataset is small, as is the case in the medical imaging domain, cross-validation is an important technique to assess the robustness of a model. Here, every sample of the dataset is used once as the test sample. The complete dataset is divided into k number of folds. In the literature study, very few studies have been undertaken to incorporate K-fold cross-validation. For a small dataset, this is a highly recommended training strategy.

4.5.8 Distinction from other viral diseases

During the COVID-19 pandemic, it has been observed that people were being infected symptomatically as well as asymptomatically, where the latter is less contagious. A CXR or CT scan is taken at a later stage to determine the degree of infection so that proper medication can be advised to a patient. In such scenarios, it becomes imperative to differentiate not only between COVID-19 versus healthy but also between COVID-19 and the other viral diseases such as pneumonia that affect human organs in a similar manner. The development of an efficient and optimal AI-based solution to specifically and exclusively detect COVID-19 is still a prime challenge.

4.5.9 Generalization

Generalization is the ability of a DL model to perform well on an unseen dataset. A model to classify dog and cat trained using black cats only may not perform well when tested on white-colored cats. This requires the training of the model on a diverse dataset. Apart from the dataset, generalization ability can also be ascertained through the choice of hyperparameters of a network that cater to the high variance (overfitting) and high bias (under fitting) issues. Regularization, dropout, batch normalization, early stopping are some techniques that can be incorporated to achieve better generalization abilities. To demonstrate the generalization ability of the proposed network, a few works like [35,75,56,81] have demonstrated the performance of their proposed model on more than one dataset.

4.5.10 Use of explainable AI

Convolutional neural network-based architecture possesses automatic feature extraction capability leading to the representation of DL models as a black box. To achieve wider acceptability of the automated solutions, it becomes imperative to have an interpretability and behavioral understanding of the model. The transparency and explain ability of AI solutions are very critical in the medical domain, especially when used for life-threatening COVID-19 diseases.

4.5.11 Lack of comparison

Instead of considering, different datasets while evaluating and training any new model, methods should be trained on the same data for comparison. Again, this poses the need to create larger and more heterogeneous datasets that can be used to train both large and small neural networks.

4.5.12 Multimodal architecture

Multimodal studies undertaken using both CXR and CT have shown great potential in learning various features and improved performance. Further, it has been observed that most of the studies used a single sequential architecture that is trained on a mix of CXR and CT datasets. It is expected that the model would perform better by employing two parallel feature extractors, one each for CT and CXR, respectively. These separately extracted features can be combined before feeding to the classification (Dense) layer.

4.6 Opportunities and scope for future work

Based on what has presented above, it has been provided some suggestions for future researchers. Some of these suggestions are apparent from the above discussion, while some entail the existing scenarios in the COVID era.

4.6.1 Availability of structured and authentic dataset

During the study of literature, it is observed that a one-to-one performance comparison between two reference papers cannot be undertaken due to lack of uniformity in the datasets and the performance metrics used. It is worth noting that the current public datasets have a limited number of images for the training and testing of AI algorithms. This necessitates the creation of one or more big, authentic, publicly available quality datasets that can be used, compared, or evaluated against by future researchers.

4.6.2 Generalization of a detection model

From the latter, it has been learnt that the datasets used by researchers are highly unbalanced. It raises concerns about the generalizability of the trained models on the prospective patients' data. Some studies utilized a few methods for combating unbalancing problems. However, a vast majority of work has suffered from the challenge of unbalanced data. Secondly, any model developed for detecting COVID-19 should perform the same with the claimed accuracy on the unseen or prospective subjects' data or data of a different hospital. Thus, It believed that a cross-dataset study is of paramount importance to ascertain the generalizability of the model with respect to variation in images across sites.. For the successful classification of a new test image, it is assumed that this new test image will consist of features similar to those learned during the training of the classification model. It necessitates the creation of a global training dataset that includes or captures major features. Furthermore, a proper benchmarking of different methods (or cross-method analysis) on the same dataset should be carried out to ascertain the efficacy of the proposed methods.However, the proposed model would be valid for other diagnoses such as kidneys (specially kidney stones) , brain and heart diagnoses with the varying of datasets among them .

4.6.3 Multimodality scope

A viral infection affects different parts of a body with different severity that leads to multiple symptoms. The accuracy of detection or diagnosis of a disease depends on the effectiveness of identifying and measuring the symptoms or patterns. Different diagnostic tools are used to identify these symptoms, measured at varying degrees and levels. Accumulation of patterns from various modalities can provide diverse features compared to the individual variables that can be utilized to learn a DL model better. For the detection of COVID-19, besides CXR and CT scan images, cough and thermal images can be used to augment the detection capabilities of the model. Any model can have practical application if it has a high degree of generalization ability, and multimodal data analysis provides a better approach towards its achievement.

4.6.4 Explainable AI

An explanation of how a DL model has reached a certain conclusion is crucial for ensuring trust and transparency, especially when one deals with identifying life-threatening COVID-19 disease. In order to be sure of the decision, doctors would like to know how AI decides whether someone is suffering from a disease by analyzing CXR and CT scan images. In this paper, we survey some existing techniques used for explaining the interpretability of DL models trained for COVID-19 classification. There is a need to explore more methods of Explainable AI for COVID-19 diagnosis as used in other applications.

4.6.5 Semi-supervised and reinforcement learning

Annotation of medical images is one of the laborious works due to the shortage of radiologists and technicians who can label the images. Deep learning has a great power to extract features from images, but its performance depends heavily on the size of labeled training data. However, one can still train deep networks by utilizing semi-supervised and reinforcement learning methods that consider a mixture of unlabelled and limited labeled data for training deep models. This could address the problem of highly imbalanced data, one of the major challenges in COVID-19 image analysis, if arising of the difficulties in labeling or annotations.

4.6.6 Severity of disease

It is important to not only predict COVID-19 but also the degree of its severity in a patient for deciding appropriate treatment. A more comprehensive understanding of the severity of the

disease can aid doctors in curing this disease carefully. In all, future improvements would require the collection of hundreds or thousands of labeled image data of severe COVID-19 and other pneumonia data. The dataset should be collected considering geographic diversity in mind, which will help to increase its applicability worldwide. In addition, future work should also be considered in the direction of identifying the infected pulmonary region as belonging to the left lung, right lung, or bi-pulmonary region. One study has already been done in this direction by employing a residual attention network as a basic block

4.6.7 Generic suggestions on COVID research

Besides the above suggestions based on the AI/ML work in COVID, a few more suggestions are in order, as discussed below.

4.6.7.1 Study on regional variation

It has been noted that the COVID-19 virus is highly mutant, and several variants have evolved over the course of time. Hence, a scaling-up of diagnostic capabilities of AI-based automated solutions quickly and widely will be critical for diagnosing new variants of COVID-19, in decision making, and in choosing the way ahead. Regional variations in the impact of the virus on human organs may be studied. This can assist in a better understanding of the identification of a robust global or local solution.

4.6.7.2 Regulation and transparency

Global solution needs global participation. As this pandemic has affected every corner of humanity, any strategy or measure to handle the crisis relies on a wider public acceptance. In order to have a better public trust, it is required that the decisions and information be transparent and available openly, especially when things are related to people's health. Any vaccine or medicine development program needs a high degree of acceptance of public confidence and should be in the common interest. At the same time, international legislation and regulatory bodies will play a crucial role in ensuring the needs of individuals, preserving intellectual rights, and resolving disputes. It is also required to ensure accessibility, availability, and affordability to everyone.

Chapter 5: Discussion

Recently, Deep learning technique in the field of machine learning has been fascinating researchers significantly. The improvement of its utilization presents an opportunity to analyze medical images and solve critical tasks. Identification and classification of COVID-19 cases from Normal and Pneumonia cases using CXR images can help to isolate the infectious subjects, which is a significant step to fight against the virus. In this project, I have trained and tested 3 different deep CNN models with different versions in a transfer learning process and their comparative evaluation is presented.

There are some essential factors for an individual model that influence their performance, such as imaging modality, image content, image quantity, distribution of the dataset, the structure of the model, model complexity, loss function, optimizer, number of epochs and so on. The accuracy records in **Tables 9 and 10** illustrate that densenets perform relatively well compared to ResNets, and InceptionNets, Therefore, it is evident that a shallow network performs well than very deep networks in this type of image dataset.

If it was analyzed the architecture of Densenet, it is observed that the network architecture is very simple and straightforward. Every layer in densenet is adding to the previous volume these 32 new feature maps. That is why feature maps might go from 64 to 256 after 6 layers. In addition, Transition Block performs as 1×1 convolution with 128 filters. Followed by a 2×2 pooling with a stride of 2, resulting on dividing the size of the volume and the number of feature maps on half. As a summary, the volume within a Dense Block remains constant, and the volume and the feature maps are halved after every Transition Block.

As known, with the progress of network depth, the vanishing gradients and degradation problem become prominent. So it is remarked that network depth can improve the performance but at a certain label. The improved version of ResNet implies Batch normalization and ReLU activation before the convolution operation, which can be the reason behind its improved performance.

However, the DensNet series provide good performance with the increase of the network layer. It can be because DenseNet concatenates the output from the previous layers, whereas ResNet does the additive operation among the previous layer with the future layer. It is also recognized that the lower version of the Inception network (InceptionV3) performs better on the dataset, compared to ResNet50. ResNet and InceptionNet demonstrate that extremely deep networks are not suitable for the dataset.Table 9 displays that DesneNets are computationally

less expensive than other types of networks, with its less time consuming Therefore it is easier to train.

Chapter 6: Conclusion and Recommendation

The chest X-ray images (COVID-19, healthy) were mostly applied to analyze lung problems. The primary diagnosing for COVID -19 depending on symptoms with different mathematical weights is vital but insufficient, because of that, the project attempts to understand the specific strengths and weaknesses of common deep learning models in order to identify COVID-19 with acceptable accuracy. This is critical for a doctor's decision-making, since each has benefits and drawbacks. Furthermore, when time, resources, and the patient's condition are restricted, the doctors may be forced to make a choice based on only one modality. Moreover, Fast and timely detection of COVID +ve patients is necessary to avoid spreading the disease and keeping it in control.So, this project work has been done to detect the COVID +ve patients from Chest X-Ray images in a simple and inexpensive way. In the work proposed in this paper, three state-of-the-art deep learning models have been adopted and ensembled. The proposed models has achieved a training accuracies are 70.38% , 80.18% , 98.89% , validation accurecies are 90.76% , 88.67% , 96.67% ,and testing accuracies are 89.58% , 88.44% , 97.29% respectivily . The highest performance achieved by the existing models is from DemseNet due to many factors such as low loss and ease of implementation, which makes it a quit efficient technique than its other competitors.

References

[1] Fang Y. , Zhang H. , Xie J. , Lin M. , Ying L. , Pang P. , Ji W. , Sensitivity of chest CT for COVID-19: comparison to RT-PCR, Radiology (2020), 200432.

[2] Bernheim A. , Mei X. , Huang M. , Yang Y. , Fayad Z.A. , Zhang N. , Diao K. , Lin B. , Zhu X. , Li K. , et al., Chest CT findings in coronavirus disease-19 (COVID-19): relationship to duration of infection, Radiology (2020), 200463.

[3] Rubin G.D. , Ryerson C.J. , Haramati L.B. , Sverzellati N. , Kanne J.P. , Raoof S. , Schluger N.W. , Volpi A. , Yim J.-J. , Martin I.B. , et al., The role of chest imaging in patient management during the covid-19 pandemic: A multinational consensus statement from the fleischner society, Chest, (2020).

[4] Farooq M. , Hafeez A. , Covid-resnet: A deep learning framework for screening of covid19 from radiographs, arXiv preprint arXiv:2003.14395 (2020).

[5] Afshar P. , Heidarian S. , Naderkhani F. , Oikonomou A. , Plataniotis K.N. , Mohammadi A. , Covid-caps: A capsule network-based framework for identification of covid-19 cases from x-ray images, arXiv preprint arXiv:2004.02696 (2020).

[6] Wang L. , Wong A. , COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest radiography images, arXiv preprint arXiv:2003.09871 (2020).

[7] Hemdan E.E.-D. , Shouman M.A. , Karar M.E. , Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images, arXiv preprint arXiv:2003.11055 (2020).

[8] Sethy P.K. , Behera S.K. , Detection of coronavirus disease (covid-19) based on deep features, Preprints 00, (2020), 2020.

[9] Maghdid H.S. , Asaad A.T. , Ghafoor K.Z. , Sadiq A.S. , Khan M.K. , Diagnosing COVID-19 pneumonia from X-ray and CT images using deep learning and transfer learning algorithms, arXiv preprint arXiv:2004.00038 (2020).

[10] Apostolopoulos I. , Aznaouridis S. , Tzani M. , Extracting possibly representative COVID-19 Biomarkers from X-Ray images with Deep Learning approach and image data related to Pulmonary Diseases, arXiv preprint arXiv:2004.00338 (2020).

[11] Loey M. , Smarandache F. , Khalifa N.E.M. , Within the Lack of COVID-19 Benchmark Dataset: A Novel GAN with Deep Transfer Learning for Corona-virus Detection in Chest X-ray Images, (2020).

[12] Zhang L. , Lu L. , Nogues I. , Summers R.M. , Liu S. , Yao J. , DeepPap: deep convolutional networks for cervical cell classification, IEEE Journal of Biomedical and Health Informatics 21(6) (2017), 1633–1643.

[13] Li C. , Xue D. , Zhou X. , Zhang J. , Zhang H. , Yao Y. , Kong F. , Zhang L. , Sun H. , Transfer Learning Based Classification of Cervical Cancer Immunohistochemistry Images, in: Proceedings of the Third International Symposium on Image Computing and Digital Medicine (2019), pp. 102–106.

[14] Song Y. , Zou J.J. , Chang H. , Cai W. , Adapting fisher vectors for histopathology image classification, in: 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017), IEEE (2017), pp. 600–603.

[15] Hall L.O. , Paul R. , Goldgof D.B. , Goldgof G.M. , Finding COVID-19 from Chest X-rays using Deep Learning on a Small Dataset, arXiv preprint arXiv:2004.02060 (2020).

[16] Zhang J. , Xie Y. , Li Y. , Shen C. , Xia Y. , Covid-19 screening on chest x-ray images using deep learning based anomaly detection, arXiv preprint arXiv:2003.12338 (2020).

[17] Abbas A. , Abdelsamea M.M. , Gaber M.M. , Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network, arXiv preprint arXiv:2003.13815 (2020).

[18] Narin A. , Kaya C. , Pamuk Z. , Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks, arXiv preprint arXiv:2003.10849 (2020).

[19] Open Source Computer Vision (OpenCV) Library. Available at: https://github.com/opencv/opencv/wiki/CiteOpenCV Accessed July 17, 2020.

[20] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [**Google Scholar**]

[21] Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708. [**Google Scholar**]

[22] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2818–2826. [**Google Scholar**]

[23] Huang G., Li Y., Pleiss G., et al., Snapshot ensembles: Train 1, get M for free, arXiv (2017), preprint arXiv:1704.00109. [Google Scholar]

[24] Holzinger A., Langs G., Denk H., et al., Causability and explainability of artificial intelligence in medicine, *WIRES Data Mining and Knowledge Discovery* 9 (2019), 1312. [PMC free article] [PubMed] [Google Scholar]

[25] Selvaraju R.R., Cogswell M., Das A., et al., Grad-CAM: Visual explanations from deep networks via gradient-based localization, In Proceedings of the IEEE international conference on computer vision, (2017), 618–626. [Google Scholar]

[26] Bengio Y., Courville A., Vincent P., Representation learning: A review and new perspectives, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), 1798–1828. [PubMed] [Google Scholar]

[27] Mahendran A., Vedaldi A., Salient deconvolutional networks, *In European Conference on Computer Vision* (2016), 120–135. [Google Scholar]

[28] Fawcett T., An introduction to ROC analysis, *Pattern Recognition Letters* 27 (2006), 861–874. [Google Scholar]

[29] Javaheri T., Homayounfar M., Amoozgar Z., Reiazi R., Homayounieh F., Abbas E., Laali A., Radmard A.R., Gharib M.H., Mousavi S.A.J., Ghaemi O., Babaei R., Mobin H.K., Hosseinzadeh M., Jahanban-Esfahlan R., Seidi K., Kalra M.K., Zhang G., Chitkushev L.T., Haibe-Kains B., Malekzadeh R., Rawassizadeh R. CovidCTNet: an open-source deep learning approach to diagnose covid-19 using small cohort of CT images. *npj Digit. Med.* Feb 2021;4(1):29. [PMC free article] [PubMed] [Google Scholar]

[30] Jin C., Chen W., Cao Y., Xu Z., Tan Z., Zhang X., Deng L., Zheng C., Zhou J., Shi H., Feng J. Development and evaluation of an artificial intelligence system for COVID-19 diagnosis. *Nat. Commun.* Oct 2020;11(1):5088. [PMC free article] [PubMed] [Google Scholar]

[31] Li L., Qin L., Xu Z., Yin Y., Wang X., Kong B., Bai J., Lu Y., Fang Z., Song Q., Cao K., Liu D., Wang G., Xu Q., Fang X., Zhang S., Xia J., Xia J. Using artificial intelligence to detect COVID-19 and community-acquired pneumonia based on pulmonary CT: evaluation of the diagnostic accuracy. *Radiology.* Aug 2020;296(2):E65–E71. [PMC free article] [PubMed] [Google Scholar]

[32] Wang B., Jin S., Yan Q., Xu H., Luo C., Wei L., Zhao W., Hou X., Ma W., Xu Z., Zheng Z., Sun W., Lan L., Zhang W., Mu X., Shi C., Wang Z., Lee J., Jin Z., Lin M., Jin H., Zhang L., Guo J., Zhao B., Ren Z., Wang S., Xu W., Wang X., Wang J., You Z., Dong J. AI-assisted CT imaging analysis for COVID-19 screening: building and deploying a medical AI system. *Appl. Soft Comput.* 2021;98:106897. [PMC free article] [PubMed] [Google Scholar]

[33] Zheng C., Deng X., Fu Q., Zhou Q., Feng J., Ma H., Liu W., Wang X. medRxiv; 2020. Deep Learning-Based Detection for COVID-19 from Chest CT Using Weak Label. [Google Scholar]

[34] Toğaçar M., Ergen B., Cömert Z. COVID-19 detection using deep learning models to exploit social mimic optimization and structured chest X-ray images using fuzzy color and stacking approaches. *Comput. Biol. Med.* 2020;121:103805. [PMC free article] [PubMed] [Google Scholar]

[35].Das A.K., Ghosh S., Thunder S., Dutta R., Agarwal S., Chakrabarti A. Automatic COVID-19 detection from X-ray images using ensemble learning with convolutional neural network. *Pattern Anal. Appl.* Mar 2021:1111–1124. [Google Scholar]

[36] Ozturk T., Talo M., Yildirim E.A., Baloglu U.B., Yildirim O., Rajendra Acharya U. Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Comput. Biol. Med.* 2020;121:103792. [PMC free article] [PubMed] [Google Scholar]

[37] Afshar P., Heidarian S., Naderkhani F., Oikonomou A., Plataniotis K.N., Mohammadi A. COVID-CAPS: a capsule network-based framework for identification of COVID-19 cases from X-ray images. *Pattern Recogn. Lett.* 2020;138:638–643. [PMC free article] [PubMed] [Google Scholar]

[38]. Abbas A., Abdelsamea M.M., Gaber M.M. Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. *Appl. Intell.* Feb 2021;51(2):854–864. [PMC free article] [PubMed] [Google Scholar]

[39] Rahimzadeh M., Attar A. A modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2. *Inform. Med. Unlock.* 2020;19:100360. [PMC free article] [PubMed] [Google Scholar]

[40] Khan A.I., Shah J.L., Bhat M.M. CoroNet: a deep neural network for detection and diagnosis of COVID-19 from chest X-ray images. *Comput. Methods Progr. Biomed.* 2020;196:105581. [PMC free article] [PubMed] [Google Scholar]

[41] Mahmud T., Rahman M.A., Fattah S.A. CovXNet: a multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization. *Comput. Biol. Med.* 2020;122:103869. [PMC free article] [PubMed] [Google Scholar]

[42] Luz E., Silva P., Silva R., Silva L., Guimarães J., Miozzo G., Moreira G., Menotti D. Research on Biomedical Engineering; Apr 2021. Towards an Effective and Efficient Deep Learning Model for COVID-19 Patterns Detection in X-Ray Images. [Google Scholar]

[43] Sitaula C., Hossain M.B. Attention-based VGG-16 model for COVID-19 chest X-ray image classification. *Appl. Intell.* May 2021;51(5):2850–2863. [PMC free article] [PubMed] [Google Scholar]

[44] Wang S., Zha Y., Li W., Wu Q., Li X., Niu M., Wang M., Qiu X., Li H., Yu H., et al. A fully automatic deep learning system for COVID-19 diagnostic and prognostic analysis. *Eur. Respir. J.* 2020;56(2) [PMC free article] [PubMed] [Google Scholar]

[45] Song Y., Zheng S., Li L., Zhang X., Zhang X., Huang Z., Chen J., Wang R., Zhao H., Zha Y., Shen J., Chong Y., Yang Y. Deep learning enables accurate diagnosis of novel coronavirus (COVID-19) with CT images. *IEEE ACM Trans. Comput. Biol. Bioinf.* 2021;(1–1) [PMC free article] [PubMed] [Google Scholar]

[46] Gupta A., Anjum, Gupta S., Katarya R. InstaCovNet-19: a deep learning classification model for the detection of COVID-19 patients using Chest X-ray. *Appl. Soft Comput.* 2021;99:106859. [PMC free article] [PubMed] [Google Scholar]

[47] Karthik R., Menaka R., M H. Learning distinctive filters for COVID-19 detection from chest X-ray using shuffled residual CNN. *Appl. Soft Comput.* 2021;99:106744. [PMC free article] [PubMed] [Google Scholar]

[48] El-Kenawy E.-S.M., Ibrahim A., Mirjalili S., Eid M.M., Hussein S.E. Novel feature selection and voting classifier algorithms for COVID-19 classification in CT images. *IEEE Access.* 2020;8:179 317–179 335. [PMC free article] [PubMed] [Google Scholar]

[49] Narayanan B.N., Hardie R.C., Krishnaraja V., Karam C., Davuluru V.S.P. Transfer-to-transfer learning approach for computer aided detection of COVID-19 in Chest Radiographs. *A&I.* 2020;1(4):539–557. [Google Scholar]

[50] Wang Z., Xiao Y., Li Y., Zhang J., Lu F., Hou M., Liu X. Automatically discriminating and localizing COVID-19 from community-acquired pneumonia on chest X-rays. *Pattern Recogn.* 2021;110:107613. [PMC free article] [PubMed] [Google Scholar]

[51] Wang B., Jin S., Yan Q., Xu H., Luo C., Wei L., Zhao W., Hou X., Ma W., Xu Z., Zheng Z., Sun W., Lan L., Zhang W., Mu X., Shi C., Wang Z., Lee J., Jin Z., Lin M., Jin H., Zhang L., Guo J., Zhao B., Ren Z., Wang S., Xu W., Wang X., Wang J., You Z., Dong J. AI-assisted CT imaging analysis for COVID-19 screening: building and deploying a medical AI system. *Appl. Soft Comput.* 2021;98:106897. [PMC free article] [PubMed] [Google Scholar]

[52] Ismael A.M., Şengür A. Deep learning approaches for COVID-19 detection based on chest X-ray images. *Expert Syst. Appl.* 2021;164:114054. [PMC free article] [PubMed] [Google Scholar]

[53] Panwar H., Gupta P., Siddiqui M.K., Morales-Menendez R., Singh V. Application of deep learning for fast detection of COVID-19 in X-rays using nCOVnet. *Chaos, Solit. Fractals.* 2020;138:109944. [PMC free article] [PubMed] [Google Scholar]

[54] Agrawal T., Choudhary P. FocusCovid: automated COVID-19 detection using deep learning with chest X-ray images. *Evolv. Syst.* 2021:1–15. [Google Scholar]

[55] Toraman S., Alakus T.B., Turkoglu I. Convolutional capsnet: a novel artificial neural network approach to detect COVID-19 disease from X-ray images using capsule networks. *Chaos, Solit. Fractals.* 2020;140:110122. [PMC free article] [PubMed] [Google Scholar]

[56] Heidari M., Mirniaharikandehei S., Khuzani A.Z., Danala G., Qiu Y., Zheng B. Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms. *Int. J. Med. Inf.* 2020;144:104284. [PMC free article] [PubMed] [Google Scholar]

[57] Jain G., Mittal D., Thakur D., Mittal M.K. A deep learning approach to detect COVID-19 coronavirus with X-ray images. *Biocybern. Biomed. Eng.* 2020;40(4):1391–1405. [PMC free article] [PubMed] [Google Scholar]

[58] Madaan V., Roy A., Gupta C., Agrawal P., Sharma A., Bologa C., Prodan R. New Generation Computing; 02 2021. XCOVNet: Chest X-Ray Image Classification for COVID-19 Early Detection Using Convolutional Neural Networks. [PMC free article] [PubMed] [Google Scholar]

[59] Nayak S.R., Nayak D.R., Sinha U., Arora V., Pachori R.B. Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: a comprehensive study. *Biomed. Signal Process Control.* 2021;64:102365. [PMC free article] [PubMed] [Google Scholar]

[60] Keles A., Keles M.B., Keles A. Cognitive Computation; Jan 2021. COV19-CNNet and COV19-ResNet: Diagnostic Inference Engines for Early Detection of COVID-19. [PMC free article] [PubMed] [Google Scholar]

[61] Ibrahim D.M., Elshennawy N.M., Sarhan A.M. Deep-chest: multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases. *Comput. Biol. Med.* 2021;132:104348. [PMC free article] [PubMed] [Google Scholar]

[62] Arora V., Ng E.Y.-K., Leekha R.S., Darshan M., Singh A. Transfer learning-based approach for detecting COVID-19 ailment in lung CT scan. *Comput. Biol. Med.* 2021;135:104575. [PMC free article] [PubMed] [Google Scholar]

[63] Turkoglu M. IRBM; 2021. COVID-19 Detection System Using Chest CT Images and Multiple Kernels-Extreme Learning Machine Based on Deep Neural Network. [PMC free article] [PubMed] [Google Scholar]

[64] Wang S.-H., Govindaraj V.V., Górriz J.M., Zhang X., Zhang Y.-D. COVID-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network. *Inf. Fusion.* 2021;67:208–229. [PMC free article] [PubMed] [Google Scholar]

[65] Hilmizen N., Bustamam A., Sarwinda D. *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems.* ISRITI); 2020. The multimodal deep learning

for diagnosing COVID-19 pneumonia from chest CT-scan and X-ray images; pp. 26–31. [Google Scholar]

[66] Zheng C., Deng X., Fu Q., Zhou Q., Feng J., Ma H., Liu W., Wang X. medRxiv; 2020. Deep Learning-Based Detection for COVID-19 from Chest CT Using Weak Label. [Google Scholar]

[67] kamil M. y. A deep learning framework to detect COVID-19 disease via chest X-ray and CT scan images. *Int. J. Electr. Comput. Eng.* 02 2021;11:844–850. [Google Scholar]

[68] Mishra N.K., Singh P., Joshi S.D. Automated detection of COVID-19 from CT scan using convolutional neural network. *Biocybern. Biomed. Eng.* 2021;41(2):572–588. [PMC free article] [PubMed] [Google Scholar]

[69] Polsinelli M., Cinque L., Placidi G. A light CNN for detecting COVID-19 from CT scans of the chest. *Pattern Recogn. Lett.* 2020;140:95–100. [PMC free article] [PubMed] [Google Scholar]

[70] Oh Y., Park S., Ye J.C. Deep learning COVID-19 features on CXR using limited training data sets. *IEEE Trans. Med. Imag.* 2020;39(8):2688–2700. [PubMed] [Google Scholar]

[71] Jin C., Chen W., Cao Y., Xu Z., Tan Z., Zhang X., Deng L., Zheng C., Zhou J., Shi H., Feng J. Development and evaluation of an artificial intelligence system for COVID-19 diagnosis. *Nat. Commun.* Oct 2020;11(1):5088. [PMC free article] [PubMed] [Google Scholar]

[72] Li L., Qin L., Xu Z., Yin Y., Wang X., Kong B., Bai J., Lu Y., Fang Z., Song Q., Cao K., Liu D., Wang G., Xu Q., Fang X., Zhang S., Xia J., Xia J. Using artificial intelligence to detect COVID-19 and community-acquired pneumonia based on pulmonary CT: evaluation of the diagnostic accuracy. *Radiology.* Aug 2020;296(2):E65–E71. [PMC free article] [PubMed] [Google Scholar]

[73] Ouyang X., Huo J., Xia L., Shan F., Liu J., Mo Z., Yan F., Ding Z., Yang Q., Song B., Shi F., Yuan H., Wei Y., Cao X., Gao Y., Wu D., Wang Q., Shen D. Dual-sampling attention network for diagnosis of COVID-19 from community acquired pneumonia. *IEEE Trans. Med. Imag.* 2020;39(8):2595–2605. [PubMed] [Google Scholar]

[74] Ucar F., Korkmaz D. COVIDiagnosis-net: deep bayes-squeezenet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images. *Med. Hypotheses.* 2020;140:109761. [PMC free article] [PubMed] [Google Scholar]

[75] Abraham B., Nair M.S. Computer-aided detection of COVID-19 from X-ray images using multi-CNN and Bayesnet classifier. *Biocybern. Biomed. Eng.* 2020;40(4):1436–1445. [PMC free article] [PubMed] [Google Scholar]

[76] Pereira R.M., Bertolini D., Teixeira L.O., Silla C.N., Costa Y.M. COVID-19 identification in chest X-ray images on flat and hierarchical classification scenarios. *Comput. Methods Progr. Biomed.* 2020;194:105532. [PMC free article] [PubMed] [Google Scholar]

[77] Brunese L., Mercaldo F., Reginelli A., Santone A. Explainable deep learning for pulmonary disease and coronavirus COVID-19 detection from X-rays. *Comput. Methods Progr. Biomed.* 2020;196:105608. [PMC free article] [PubMed] [Google Scholar]

[78] Dhiman G., Chang V., Singh K.K., Shankar A. ADOPT: automatic deep learning and optimization-based approach for detection of novel coronavirus COVID-19 disease using X-ray images. *J. Biomol. Struct. Dyn.* 2021:1–13. 0, no. 0. pMID: 33475019. [PMC free article] [PubMed] [Google Scholar]

[79] Serte S., Demirel H. Deep learning for diagnosis of COVID-19 using 3D CT scans. *Comput. Biol. Med.* 2021;132:104306. [PMC free article] [PubMed] [Google Scholar]

[80] Pathak Y., Shukla P., Tiwari A., Stalin S., Singh S., Shukla P. IRBM; 2020. Deep Transfer Learning Based Classification Model for COVID-19 Disease. [PMC free article] [PubMed] [Google Scholar]

[81] Alshazly H., Linse C., Barth E., Martinetz T. Explainable COVID-19 detection using chest CT scans and deep learning. *Sensors.* 2021;21(2) [PMC free article] [PubMed] [Google Scholar]

[82] Shah V., Keniya R., Shridharani A., Punjabi M., Shah J., Mehendale N. Diagnosis of COVID-19 using CT scan images and deep learning techniques. *Emerg. Radiol.* Jun 2021;28(3):497–505. [PMC free article] [PubMed] [Google Scholar]

[83] Wang S., Kang B., Ma J., Zeng X., Xiao M., Guo J., Cai M., Yang J., Li Y., Meng X., et al. European radiology; 2021. A Deep Learning Algorithm Using CT Images to Screen for Corona Virus Disease (COVID-19) pp. 1–9. [PMC free article] [PubMed] [Google Scholar]

[84] Zhou T., Lu H., Yang Z., Qiu S., Huo B., Dong Y. The ensemble deep learning model for novel COVID-19 on CT images. *Appl. Soft Comput.* 2021;98:106885. [PMC free article] [PubMed] [Google Scholar]

Appendices

```
from pathlib import Path
from PyQt5 import QtCore, QtGui, QtWidgets
import sys
from utils import *
import cv2
import tensorflow as tf
import numpy as np
import os

categories = ["COVID +ve","COVID -ve"]
image_size = 224
# model_dir = Path(path.dirname(__file__), "data", "model", "mymodel.h5")
inception_path = 'C:/Users/hp/Desktop/VS/SavedModels/inception_v3.h5'
resnet_path = 'C:/Users/hp/Desktop/VS/SavedModels/resnet50_v2.h5'
densenet_path = 'C:/Users/hp/Desktop/VS/SavedModels/densenet201.h5'
print("Loading Models.....")
inception_model = tf.keras.models.load_model(inception_path)
resnet_model = tf.keras.models.load_model(resnet_path)
densenet_model = tf.keras.models.load_model(densenet_path)
models = [inception_model,resnet_model,densenet_model]
models_name = ["Inception","Resnet","DenseNet"]
print("Models Loaded")

def ensemble(x, weights, models):
    '''
    returns a weighted average of predictions made by the models\n
    x -> input image \n
    weights -> a list of weights \n
    models -> a list of models\n
    '''
    outputs = []
    for model in models:
        outputs.append(list(model.predict(x)[0]))

    outputs = np.array(outputs)
    avg = np.average(a=outputs,axis=0,weights=weights)
    return avg

def predict(input_image):
    '''
    returns a predicted class, associated probablity
    input_image = an image
    models = list of models
    '''
    weights = [0.172,0.601,0.228]
    input_image = input_image/255.0
    img = input_image.reshape(-1,image_size,image_size,3)
```

```
    model_predictions = []
    model_probs = []
    for i in range(len(models)):
        print(models_name[i], categories[np.argmax(models[i].predict(img))])
        model_predictions.append(categories[np.argmax(models[i].predict(img))]
)
        model_probs.append(models[i].predict(img))
    avg_pred = ensemble(img,weights,models)
    if model_predictions.count(categories[0]) == 1:
        print(categories[0],
model_probs[model_predictions.index(categories[0])][0])
    return categories[np.argmax(avg_pred)], avg_pred

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(447, 492)
        MainWindow.setMaximumSize(QtCore.QSize(447, 600))
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.xrayImage = QtWidgets.QLabel(self.centralwidget)
        self.xrayImage.setGeometry(QtCore.QRect(20, 30, 200, 251))
        self.xrayImage.setText("")
        self.xrayImage.setObjectName("xrayImage")

        self.predictedLabel = QtWidgets.QLabel(self.centralwidget)
        self.predictedLabel.setGeometry(QtCore.QRect(40, 280, 361, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(12)
        self.predictedLabel.setFont(font)
        self.predictedLabel.setObjectName("predictedLabel")

        self.browseImageBtn = QtWidgets.QPushButton(self.centralwidget)
        self.browseImageBtn.setGeometry(QtCore.QRect(250, 20, 161, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(11)
        self.browseImageBtn.setFont(font)
        self.browseImageBtn.setObjectName("browseImageBtn")
        self.browseImageBtn.clicked.connect(self.browseImage)

        self.predictBtn = QtWidgets.QPushButton(self.centralwidget)
        self.predictBtn.setGeometry(QtCore.QRect(250, 80, 161, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(11)
        self.predictBtn.setFont(font)
```

```
        self.predictBtn.setObjectName("predictBtn")
        self.predictBtn.clicked.connect(self.prediction)


        self.probLabel = QtWidgets.QLabel(self.centralwidget)
        self.probLabel.setGeometry(QtCore.QRect(40, 320, 111, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(12)
        self.probLabel.setFont(font)
        self.probLabel.setObjectName("probLabel")

        self.covidPositive_Prob = QtWidgets.QLabel(self.centralwidget)
        self.covidPositive_Prob.setGeometry(QtCore.QRect(40, 360, 361, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(12)
        self.covidPositive_Prob.setFont(font)
        self.covidPositive_Prob.setObjectName("covidPositive_Prob")
        self.covidPositive_Prob.setWordWrap(True)


        self.covidNegative_Prob = QtWidgets.QLabel(self.centralwidget)
        self.covidNegative_Prob.setGeometry(QtCore.QRect(40, 400, 361, 41))
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(12)
        self.covidNegative_Prob.setFont(font)
        self.covidNegative_Prob.setObjectName("covidNegative_Prob")
        self.covidNegative_Prob.setWordWrap(True)

        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "COVID-19
Detector"))
        self.predictedLabel.setText(_translate("MainWindow", "PREDICTION:"))
        self.browseImageBtn.setText(_translate("MainWindow", "Browse Image"))
        self.predictBtn.setText(_translate("MainWindow", "Analyse"))
        self.probLabel.setText(_translate("MainWindow", "Probabilities"))
        self.covidPositive_Prob.setText(_translate("MainWindow", "COVID-19
+ve:"))
        self.covidNegative_Prob.setText(_translate("MainWindow", "COVID-19 -
ve:"))
```

```
    def browseImage(self):
        fm = QtWidgets.QFileDialog.getOpenFileName(None,"OpenFile")
        filename = fm[0]
        self.image = cv2.imread(filename)
        self.xrayImage.setPixmap(QtGui.QPixmap(filename))
        self.xrayImage.setScaledContents(True)

    def prediction(self):
        self.image = cv2.resize(self.image, (image_size,image_size))
        print("Analysis....")
        try:
            label, probabilities =
predict(self.image)   #("COVID",[0.98,0.02])
            self.predictedLabel.setText("PREDICTION: "+label)
            #print(probabilities)
            self.covidPositive_Prob.setText("COVID-19 +ve: " +
str(probabilities[0]))
            self.covidNegative_Prob.setText("COVID-19 -ve: " +
str(probabilities[1]))
            print("Analysis done")
        except:
            msgError = QtWidgets.QMessageBox()
            msgError.setIcon(QtWidgets.QMessageBox.Critical)
            msgError.setWindowTitle("Error")
            msgError.setText("Oops!! Error")
            msgError.exec_()


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Attachment A:

Disclaimer Statement

This report was written by students at the Telecommunications Engineering Department, Faculty of Engineering and IT, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the student(s). An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.