1. Introduction

1.1. Convolutional Neural Network

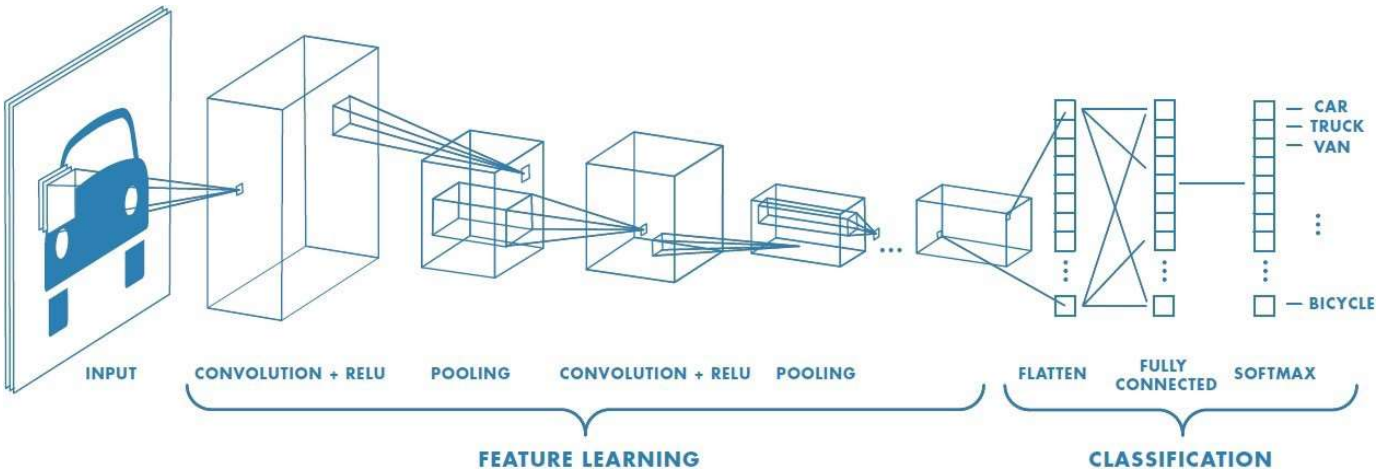

Figure 1 A Convolutional Neural Network (Saha 2018)

A Convolutional Neural Network (**CNN**) is a Deep Learning algorithm capable of capturing an input image, assigning importance aspects in the image, and being able to distinguish one from another (Saha 2018). Compared to other classification algorithms, the pre-processing required in a CNN is much less. While filters are manually engineered in primitive methods, CNN can learn those characteristics with enough training. A CNN design is like that of the human brain communication system of Neurons.

A CNN can capture the Spatial and Temporary dependencies in an image by applying relevant filters. Because of the reduction of the number of parameters involved and the reusability of weights, the architecture performs a better fit to the image dataset. The CNN can be trained to better understand the image sophistication.

1.1.1. Input Image

The input image can be a Grayscale, HSV, RGB, CMYK, etc. The RGB image consists of three arrays of numbers, one for Red, Green and Blue. CNN 's job is to simplify images into an easier-to-process type, without missing features that are crucial to making a successful prediction. This is important when designing an architecture that is not only good in terms of learning features but also scalable to massive datasets (Saha 2018).

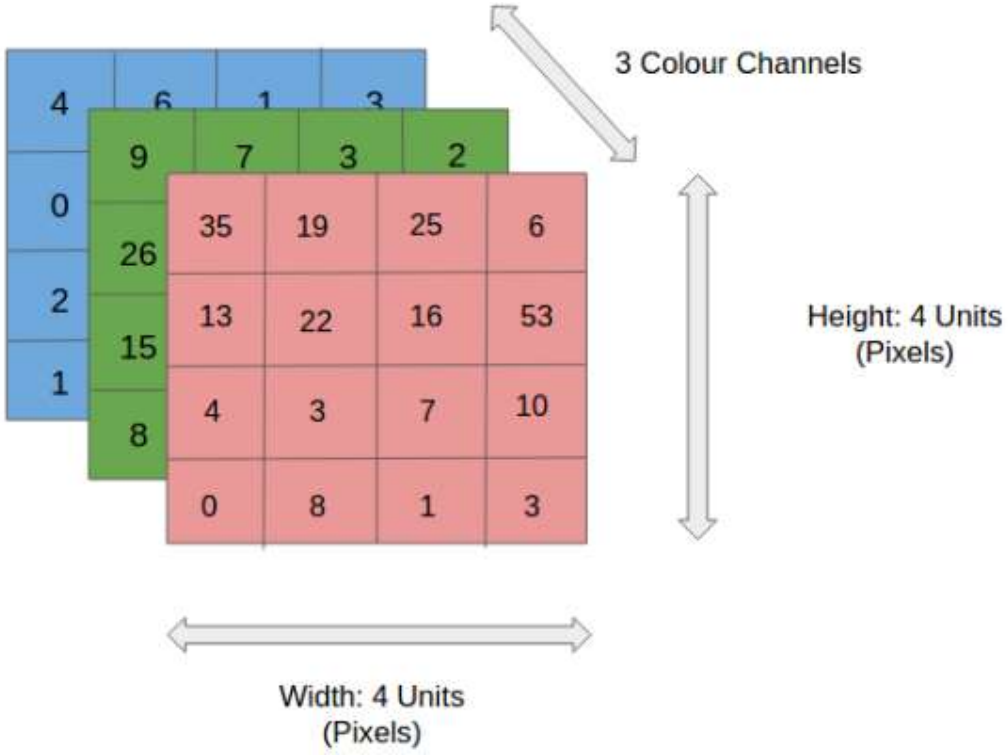

Figure 2 4x4x3 RGB Image (Saha 2018)

1.1.2. Kernel

Kernel is represented by a square matrix which will be relatively smaller than the input image, usually 3x3 or 5x5. Inside kernel convolutions are calculated, which matrix multiplication between kernel matrix and input image matrix (where at present the kernel is placed) happens. Kernel moves along the whole image to calculate the convolutions and for the movement, the value for stride will be decided by the developer. The kernel is moved to the right of the image until the whole width is covered, then kernel is moved down by one stride and convolutions starts again from the left side of the image (Saha 2018). This process is repeated until the whole image is covered. As a result of these convolutions network gets the knowledge about features like edge, gradient orientation and colour of the image. With further layers network gets the wholesome understanding of input.

1.1.3. Padding

When convolutions are applied, it eats away the edges of the image which reduces the size of image (Saha 2018). In order to stop this, edge of the image is padded with zeros in order to preserve the dimensions. Dotted line in the figure 3 represents the padding.

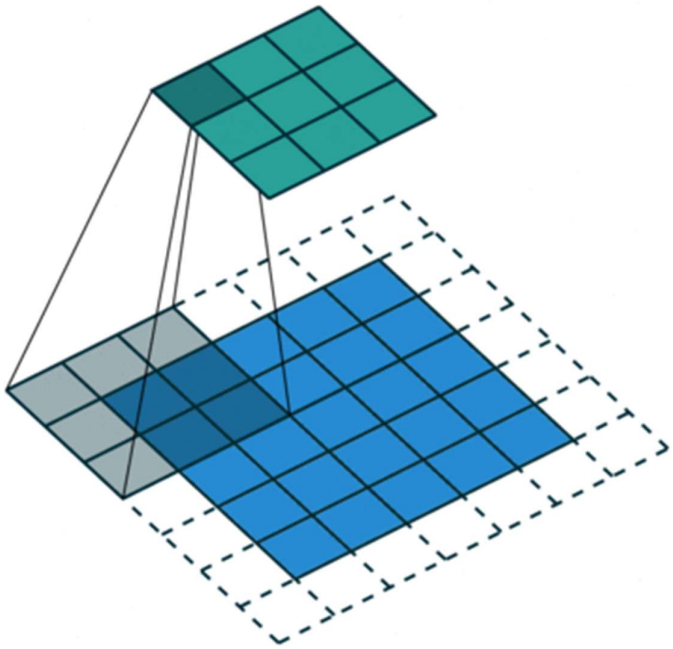

Figure 3 Kernel with padding (Saha 2018)

1.1.4. Activation

Activation functions are used to break the linearity in the model. Tanh, Sigmoid, Relu functions are some of the commonly used activation functions. Earlier sigmoid where used but nowadays Relu activations are the most common, which clips the negative part of the input to zero and keeps the positive as such (Saha 2018).

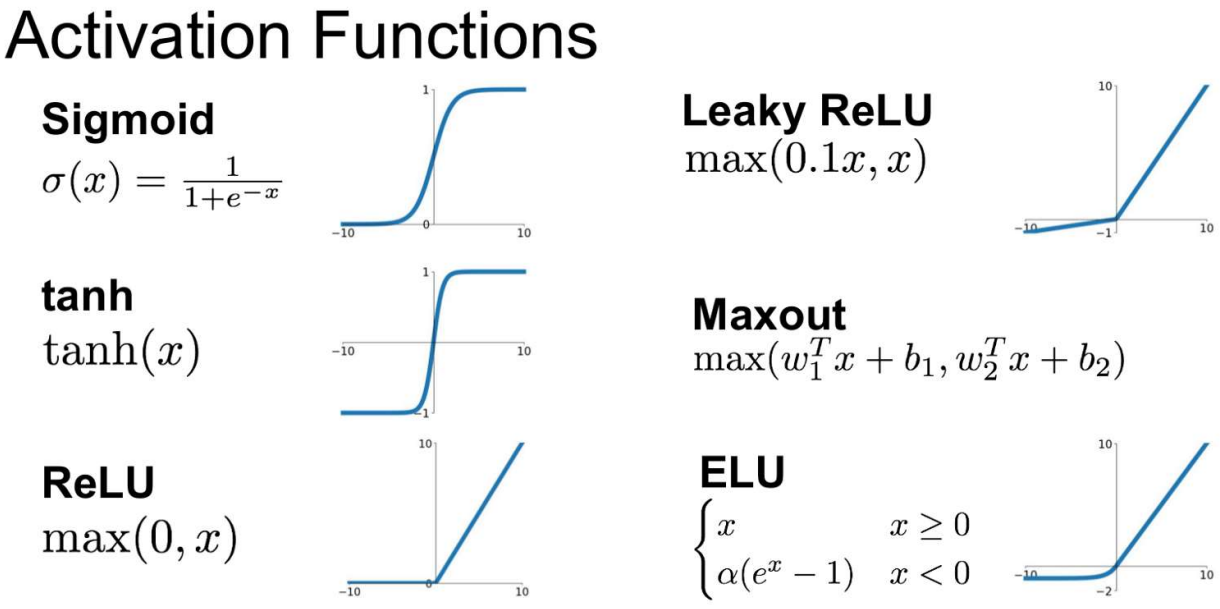

Figure 4 Activation Functions (Udofia 2018)

1.1.5. Pooling Layer

As like the Convolutional Layer, the Pooling layer reduces the spatial dimensions of the convolved image. It is to reduce the computing resources needed for the data analysis by decreasing the dimensionality. Pooling process maintains the effectiveness of training by extracting the important features. Pooling also helps in reducing the noise (Saha 2018).

Two types of pooling are used, **Max pooling** and **Average pooling**. As the name suggests max pooling takes the maximum value and average pooling takes average from the image section which is covered by the kernel.

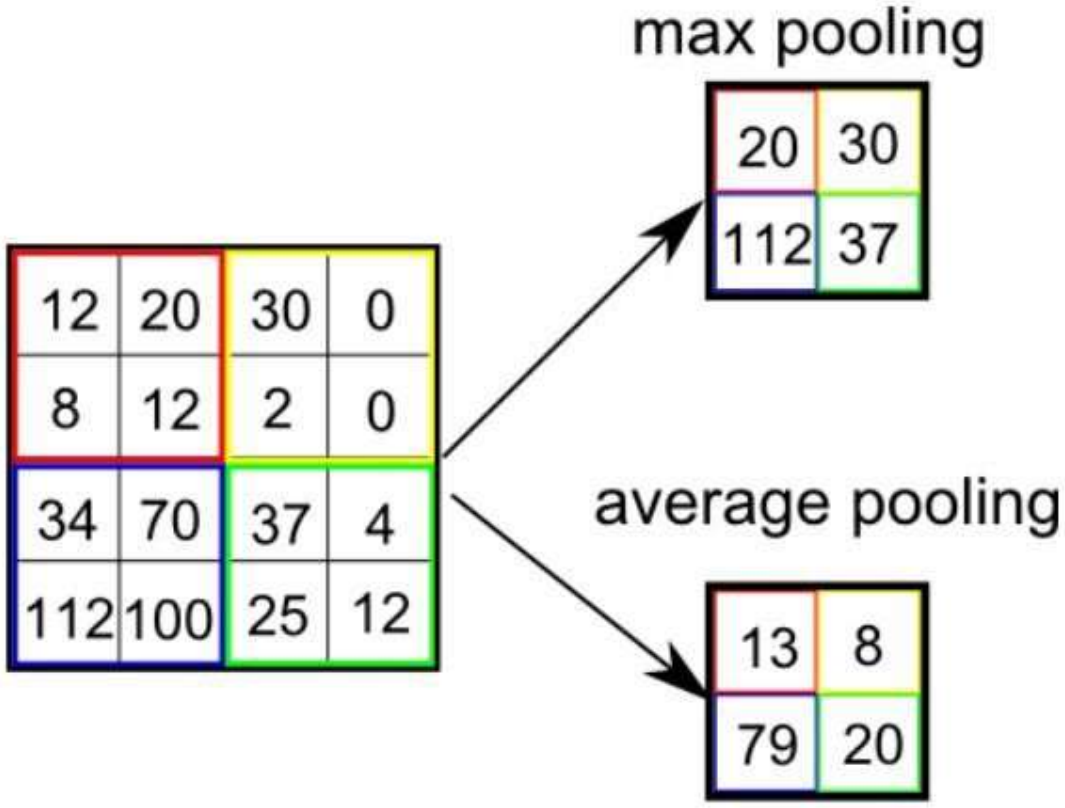

Figure 5 Types of Pooling (Saha 2018)

1.1.6. Fully Connected Layer

After multiple number of convolutions and pooling, the data is feed into a fully connected network. For this the matrix is flatten into a column vector and feed into a multilevel perceptron (Saha 2018). The model will distinguish between dominant and other low-level features in image across a sequence of epochs and classify them using the SoftMax classification technique.

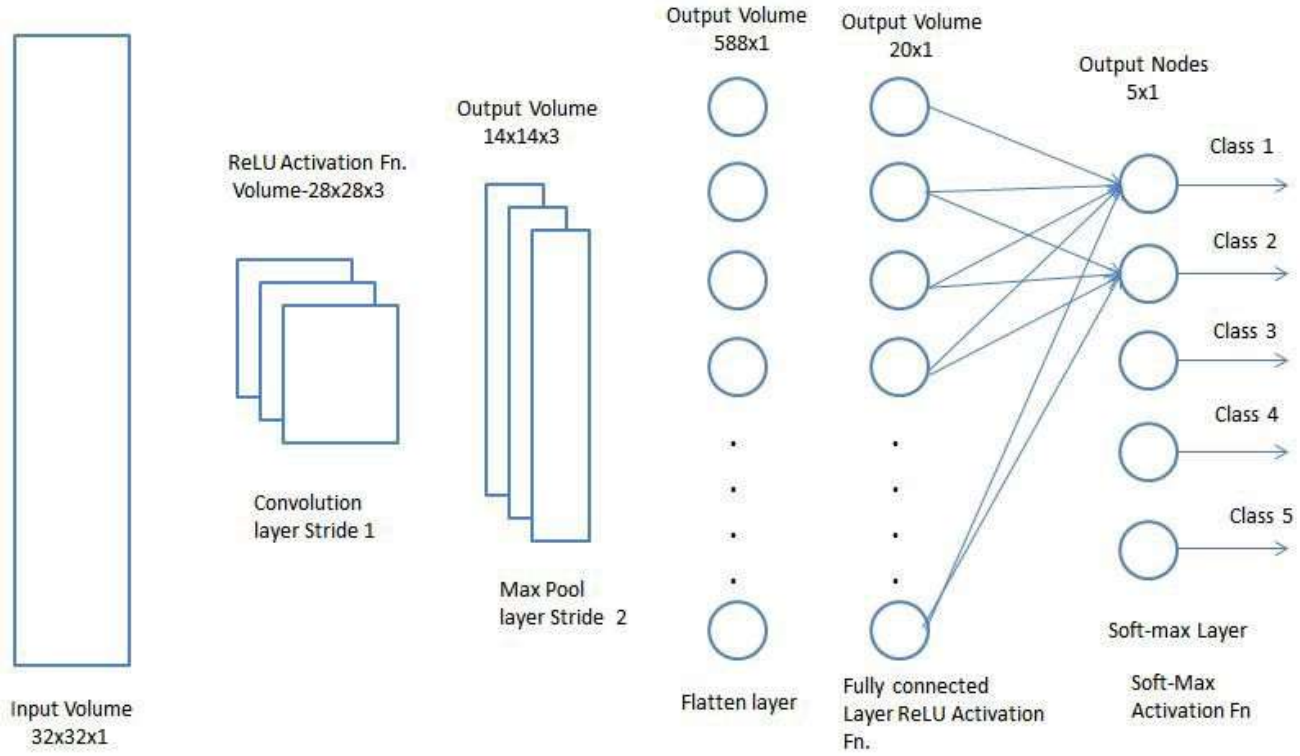

Figure 6 FC Layer (Saha 2018)

1.2. Algorithms

There are many deep CNN algorithms available with the rise of Alexnet in 2012, some prominent examples are VGG-16, ResNet from Microsoft and Inception from Google etc. CNN networks can be trained by different methods like building a CNN algorithm and training from scratch or

Training on already built algorithm but without using any pretrained weights for example algorithms used in the ImageNet competition.

Transfer Learning: Training a deep neural network from scratch requires thousands of images for each class. Using a pretrained network is a common and highly effective approach for deep learning on small image datasets. Pretrained networks can be used for classification problems even if the final labels or classes didn't match. By removing the top layer and adding a network of choice can be done easily which is also computational not so expensive. Here we are going to train 3 algorithms using transfer learning with pretrained weights from ImageNet. The algorithms used are:

1.2.1. ResNet

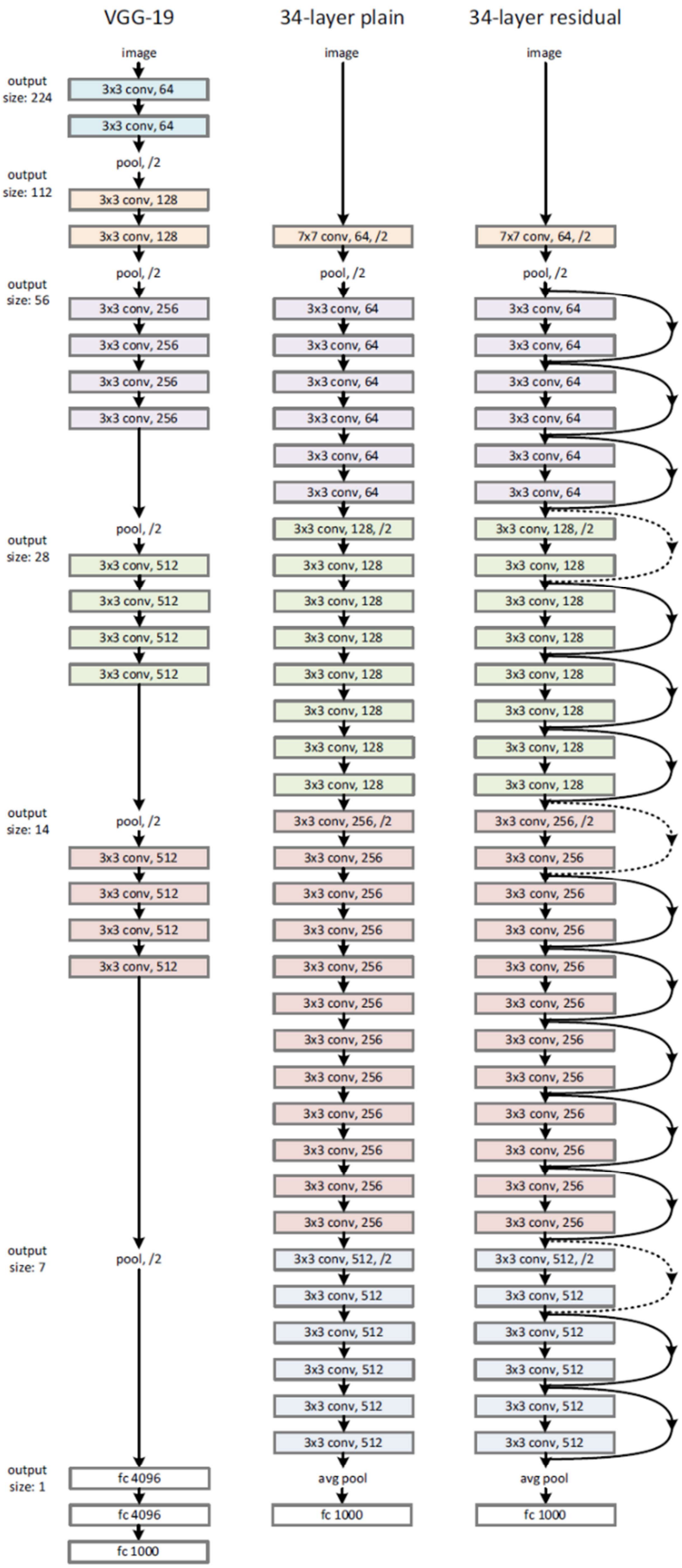

Figure 7 ResNet architecture (Tsang 2018).

ResNet can have a very deep network of up to 152 layers, learning the functions of residual representation instead of directly learning the representation of the signal.

ResNet introduces skip connection to fit the input from the preceding layer to the next layer without any input changes. Skip connection makes it possible to have a deeper network and finally ResNet becomes the winner of ILSVRC 2015 in the fields of image classification, detection and localization, as well as the winner of MS COCO 2015 detection (Tsang 2018).

1.2.2. Inception

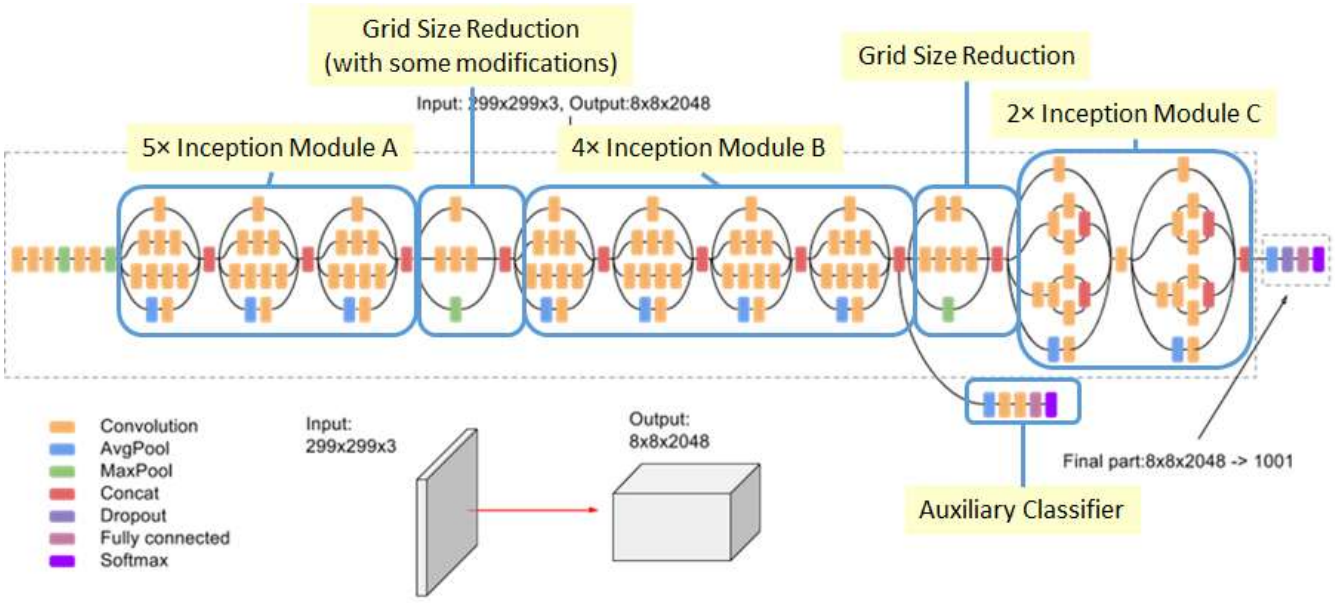

Figure 8 Inception architecture (Cloud TPU documentation 2020).

Inception v3 is a widely used model of image recognition that has been demonstrated to achieve greater than 78.1 percent accuracy on the ImageNet dataset. The model is the result of a multitude of ideas produced over the years by numerous researchers.
The model itself consists of symmetrical and asymmetrical building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and layers that are fully connected. Batchnorm is commonly used in the model and applies to inputs for activation. Losses are measured through Softmax (Cloud TPU documentation 2020).

1.2.3. DenseNet121

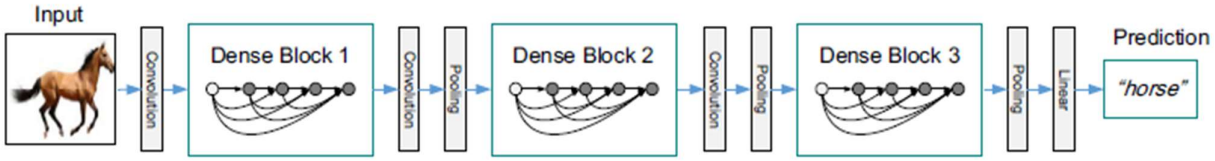

Figure 9 A deep DenseNet with three dense blocks (Huang 2017).

DenseNets exploits the network 's potential through reuse of features, yielding condensed models that are easy to train and highly efficient parameters. Concatenating feature maps learned from various layers increases the input variation of subsequent layers and

improves efficiency. This reflects an essential distinction between DenseNets and ResNets. Compared with inception DenseNets are simpler and more efficient networks which also concatenate features from different layers (Huang 2017).

1.3. Working Environment

Google Colab is a free cloud service, supporting free GPU now. Anyone can develop the coding skills for the Python language programming. Deep learning frameworks are built utilising common libraries, such as Keras, TensorFlow, PyTorch and OpenCV.

2. Problem Definition

Multiple respiratory diseases can diagnose using chest x-ray, which includes COVID-19, Cardiomegaly, Consolidation, Nodule, Pneumonia, Pneumothorax and many more (Gupta 2017).

Our project deals with mainly 2 problems:

I. Identifying whether the person is healthy or affected with any of the above mentioned 6 diseases.

II. Predicting the chances of an individual getting effected by COVID-19 if the person is already affected by any other respiratory disease.

We think this is very useful in the current situation where the world is going through an epidemic where doctors, nurses and other healthcare workers are trying to contain COVID-19. As there is an exponential increase in COVID-19 positive individuals where there is a huge shortage of testing kits as well. So, as a primary test x-rays of individuals showing symptoms of COVID-19 can be assessed and then refer to a doctor. Secondly, people with other respiratory symptoms can first use this technique to diagnose the disease and then consult the doctor if diagnosed to be ill. It is a better way than a healthy man going to a clinic and wasting the time of the doctor hence an automated analysis system is required to save medical professionals valuable time. Finally, there is a popular belief that only elder people die due to corona and younger ones are safe, but which isn't the case. So, if an individual diagnosed with any other respiratory disease, the person is highly vulnerable to coronavirus attacks irrespective of his/her age. By seeing the predicted value of the probability of the virus attack the person will be more serious about being in quarantine and precautions needed to be taken against the virus attack, which is the only way to eradicate any future spread of COVID-19 at the present stage as there isn't any medication developed against the virus.

3. Methodology

3.1. Collecting Data

There isn't any standard dataset for this problem, but fortunately there were images available on the internet of covid-19 patients. Normal and other respiratory diseases data is available in various datasets. So, we collected both data and merged into a single dataset.

3.1.1. NIH Dataset

National Institutes of Health (NIH) is under the U.S. Department of Health and Human Services and they have released a public dataset (https://nihcc.app.box.com/v/ChestXray-NIHCC) of 40GB containing more than 30,000 patients in which many having advanced lung diseases (News Releases NIH, 2017). Even though there is a huge availability of data, annotated data is very limited, due to this issue a subset of this dataset was downloaded from Kaggle (Beltrán 2019). By doing so we had to compromise on the image resolutions as the subset was having images resized to 224x224 instead of 1080x1080 from NIH dataset.

3.1.2. COVID Dataset

A research team Qatar University, Doha, Qatar and University of Dhaka, Bangladesh with help of doctors from Pakistan and Malaysia created a database of COVID-19 positive patients (Rahman 2020). The dataset contains 1080x1080 images of 219 COVID-19, 1341 normal and 1345 pneumonia images.

3.2. Dataset Creation

Now we have all the images, we need to train the network, but first it's better to arrange the data. Primarily, dataset was created with 16 classes, which adversely affected the performance of the algorithm, so the classes were reduced to 7. In order to create these 7 classes, we utilised both datasets which were downloaded in the previous step. Normal, COVID-19, Pneumonia images were taken from COVID dataset and remaining four classes (Cardiomegaly, Consolidation, Nodule, Pneumothorax) were taken from NIH dataset. A directory named Dataset was created and inside it each class were assigned to separate directories. Training, Validation and Testing split could be done here manually but it is done during the execution. Finally, the dataset was upload to Google Drive and Dataset is ready for loading into the algorithm to start train the network.

3.3. Loading Dataset and Pre-processing

3.3.1. Loading Images

First step for reading any file from Google drive is to mount drive with Collaboratory. After mounting all file in the drive can be accessed through colab. Loading dataset into the program was achieved with the help of OpenCV functions. OpenCV reads images as NumPy arrays of uint8 ranging from 0 to 255 using imread function. The input argument for the function is file path to the corresponding image. 219 images per class were only loaded into the colab memory for pre-processing. This was done to achieve class balancing where 219 is the number of images in the smallest class. Loading images from the disk took almost 1400 seconds since disk transfer are the lowest compared to processes inside Ram or Gpu. So, to tackle this minor issue, parallel function was introduced. As a result, with 80 process running parallelly image loading time was reduced to 32 seconds from 1400 seconds.

3.3.2. Pre-processing

The loaded images are of two resolutions, and so the 1080x1080 images are resized into 224x224 since the other way doesn't make any sense. OpenCV resize function is used for resizing. This function takes image to be resized and the output resolution as the input argument and returns the resized image.

Why not data augmentations?

Data augmentations were done on the images in order to increase the dataset. But validation and testing accuracy were significantly lower than training without augmentation. As the normal X-ray itself is confusing even for a human being after augmentations we can't recognize if we change any characterises of data. So, data augmentation wasn't performed before the training.

As these two processes are repeated for all classes, **a user defined function** '**read_img**' is created. This function takes file path as input argument. Then reads images parallelly using imread function and this image is given as input to the resize function. Then the resized image is returned from the read_img function and is stored inside a list.

3.3.3. Creating Labels

A total of 7 lists with 219 elements are created to store each class. Now these NumPy elements are in the range between 0 and 255 and type is uint8. It is converted into float32 and range is altered to 0 and 1. And these 7 lists are combined into list 'X' which contains 1533 elements of all classes. Importantly these elements are not shuffled and are in the same order a they were read from the drive. For creating label, list 'Y' is created. Elements of Y are '0' 219 times, then '1' 219 times and so on till '6' 219 times where 219 is the number of elements in each class. Now Y is the list of labels corresponds to the list of images X.

3.3.4. Training and Testing Split

Training and testing split is done using train_test_split () from sklearn library. Parameters of this function are list of images (X) and labels (Y), size of test set (ranging from 0 to 1) and random_state which controls the shuffling before splitting data which can be none or an integer. Passing an integer can help to maintain same shuffle across all function calls (Pedregosa, 2007). Output of function is 4 lists (training data + corresponding labels and testing data + corresponding labels). The test split used is 10% which smaller than the ideal scenario.

3.4. Loading Algorithms and Weights

Algorithms are downloaded from ImageNet using tensorflow.keras.applications function. "Keras Applications are canned architectures with pre-trained weights" (Tensorflow documentation 2013).

3 algorithms are downloaded without including the top layer but the weights from ImageNet were include.

3.4.1. ResNet152V2

It had **Total parameters**: 58,331,648

Trainable parameters: 58,187,904

Non-trainable parameters: 143,744

3.4.2. InceptionV3

It had **Total parameters**: 21,802,784

Trainable parameters: 21,768,352

Non-trainable parameters: 34,432

3.4.3. DenseNet121

It had **Total parameters**: 21,802,784

Trainable parameters: 21,768,352

Non-trainable parameters: 34,432

3.5. Adding Final Layer

After the getting the architecture and the weights the top layer is replaced with a combination of dense layers for classification for the current problem. First the output of the algorithm is flattened to a column vector using GlobalAveragePooling2D. Then combination of dense layer with relu activation and dropouts are added twice into the model. Finally, for the classification a dense layer with 7 outputs and SoftMax activation function is added.

Now combining both the algorithms downloaded and the final layer, neural network for training is ready.

3.6. Training and Validation

Train is done using model.fit function imported from tensorflow.keras.models. Inputs for the function are Training data, training labels, batch size, epochs, verbose, validation_split. Training data is the list images after train-test split and training labels are the corresponding label. During training images are not passed one by one, instead group of images are passed and the number of images in a group is batch size. An epoch is the process by which the network goes through the complete data set. Epoch is set to 30. Verbose is a visualisation technique which can be 0,1,2. Validation_split is used to split between train and validation data. This validation set is used to validate the model after each epoch. Validation set is set to 10% of the training set. After each epoch training loss, training accuracy, validation loss and validation accuracy are shown. These validation loss and accuracy is used to optimize the algorithm.

3.7. Optimization of Algorithm

3.7.1. Loss Function

The error for the current state of the model must be estimated repeatedly as part of the optimisation algorithm. It involves the collection of an error feature, conventionally named a loss function, which can be used to measure the model's loss such that the weights can be changed to decrease the loss on the next measurement.

3.7.2. Optimizer

The objective of optimizer is to minimise the loss function. Optimizer helps to mould the most accurate model. The optimizer used is Stochastic Gradient Descent. Learning Rate ensures the weights are changed in correct pace, it shouldn't be too large or too small (Algorithmia 2018).

3.7.3. Metrics

Metrics is which the algorithm depends to understand whether the change is in the right direction or not. Accuracy is set as metrics in this problem.

3.8. Testing

Testing done after optimizing the algorithm and when the developer is 100% confident with the model. Testing data should be only used once and as the final step for evaluation of the model. It should be an unseen dataset to get the most accurate measure of the algorithm. Model.evalute() is used to run the test data. Test dataset and label are passed as the input arguments for this function.

3.9. Evaluation

Training accuracy, loss and validation accuracy, loss is stored in history which is the output of the training function. A user defined function 'plot_acc_loss' is created for the plotting of these value in two separate graphs. One graph plots training accuracy and loss over epochs and second graph plots validation accuracy and loss over epochs. The input of the function is history and number of epochs. Plotting is done using matplotlib function.

Confusion matrix and classification report are generated for test data set. Both functions are imported from sklearn.metrics module. Confusion matrix is a matrix containing correct predictions along the identity axis and rest of the elements are incorrect predictions. Classification report

generates the Precision, Recall, F-1 score and Support. Precision is the fraction of true positive by true positive + false positive. Recall is the fraction of true positive by true positive + false negative. F1 score is the harmonic mean of precision and recall (Wikipedia 2020). Three of these can be also be manually calculated from the confusion matrix. Support is the number of samples per class.

4. Experiments & Discussion

There algorithms were trained and test on the same dataset to compare the performances.

4.1. Training Layers

	Trainable parameters	Total parameters	Time taken per Epoch
ResNet152V2	58,458,887	58,602,631	65
InceptionV3	22,039,335	22,073,767	23
DenseNet121	7,093,767	7,177,415	27

As the figures shows, ResNet152 is more than 58 million learnable parameters. Inception is having more than 22 million learnable parameters which very less compared to the ResNet152. Least number of parameters is for DenseNet121, which above 7 million. Time taken to run one epoch is significantly different between ResNet which takes 65s and Inception which takes only 23s. But even with one-third of parameters Densenet is taking 4s more than Inception.

4.2. Training,Validation Accuracy and Loss

Graphs are plotted between accuracy over epochs for both training and validation set. Training plot is used to understand how well the model is learning and validation plot to understand the generalization capacity of the model. The gap between validation and training loss is called the generalisation gap (Brownlee 2019).

4.2.1. ResNet152V2

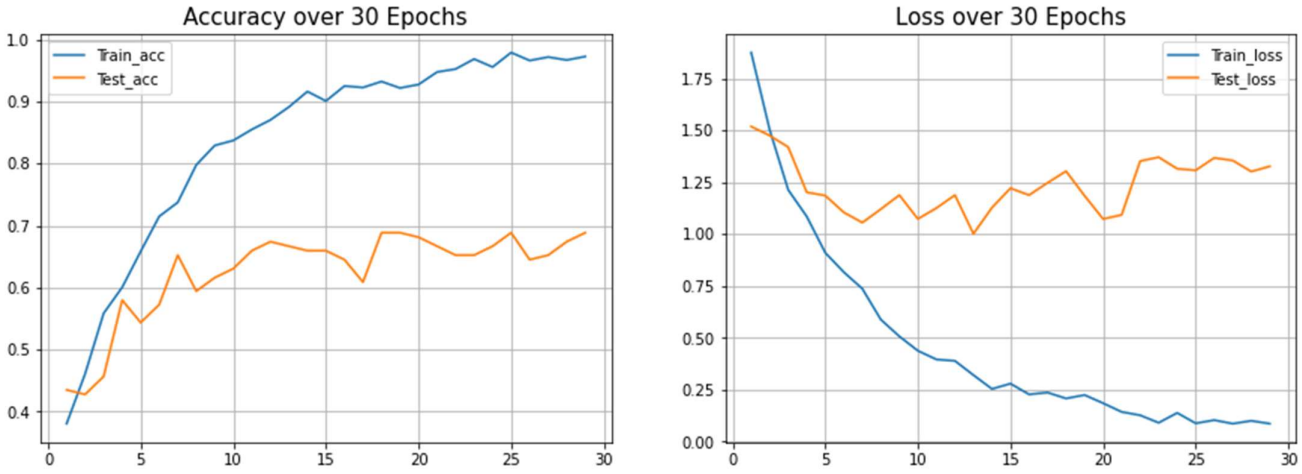

Figure 10 Training Accuracy and Loss of ResNet152V2

The graphs here clearly show overfitting of the model. The model is learning too well, and it is even learning the noises also. Training is loss is decreasing but after the 12th or 13th epoch, validation loss is starts to increase. There is large generalisation gap, which clearly indicates that the model is learning the dataset well but not generalising the key features.

4.2.2. InceptionV3

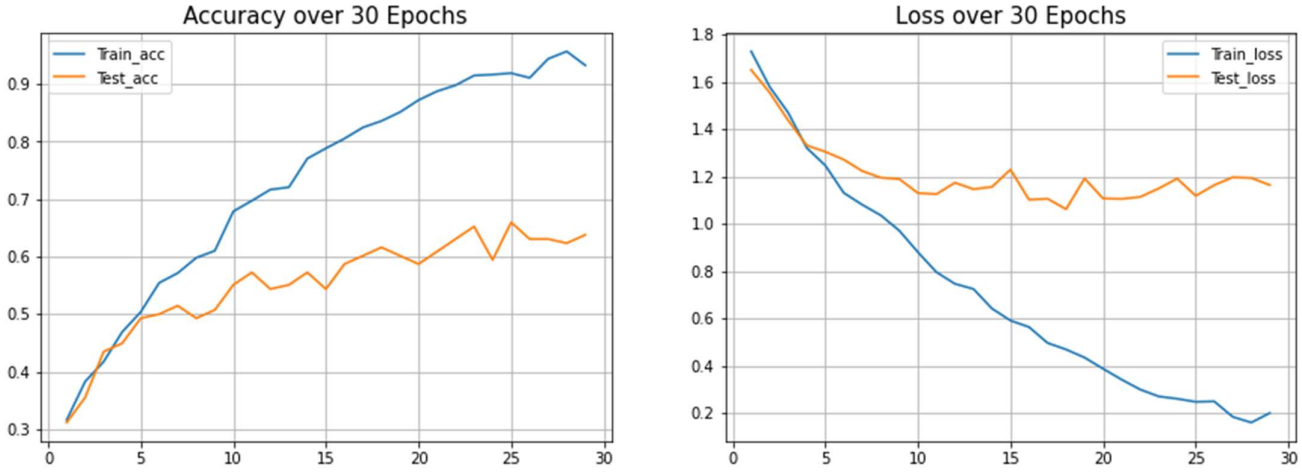

Figure 11 Training Accuracy and Loss of InceptionV3

The graphs illustrate better generalisation of the validation set compared to ResNet plot. Even though this model tends to overfit towards the back end of the training. As shown by the graph training accuracy is rising even if the validation accuracy achieves almost a stable curve. The training loss is also decreasing in a steady manner for the entire time period, but validation loss curve doesn't change much 12th or 13th epoch.

4.2.3. DenseNet121

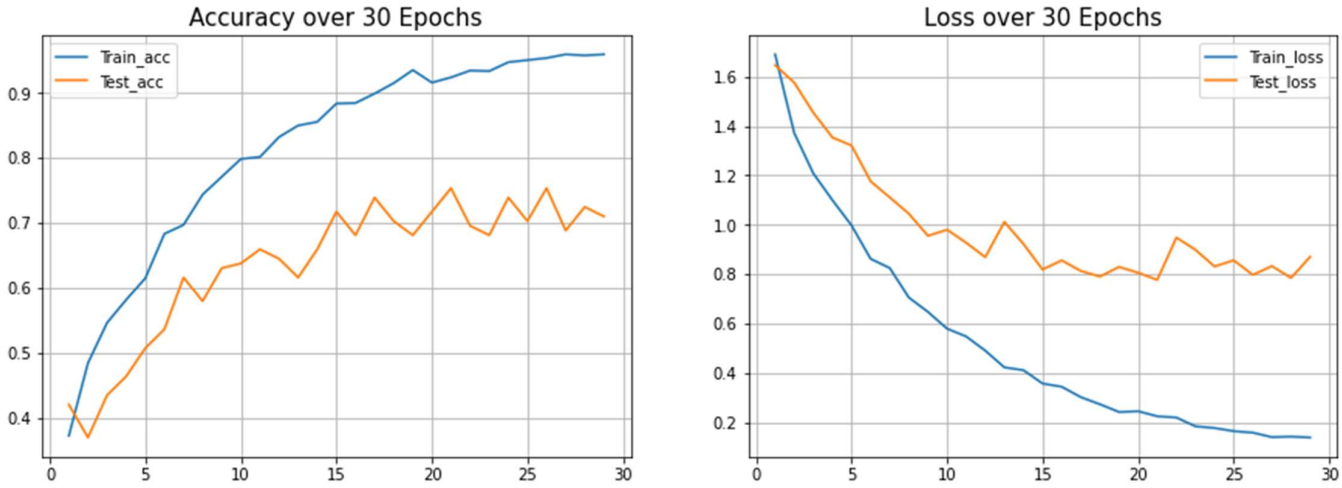

Figure 12 Training Accuracy and Loss of DenseNet121

DenseNet shows relatively better model in comparision with other two model. It is showing a better learning curve as well as generalisation curve. But still there are plenty of room for improvement especially in the case of generalisation.

4.3. Test Accuracy and Loss

	Accuracy	Loss
ResNet152V2	0.7013	1.3384
InceptionV3	0.6948	1.0830
DenseNet121	0.6429	1.0588

All three agorithms were tested on a similar test set to find out the performance of the model on an unseen dataset. The table illustrates the test result with accuracy of prediction and loss. The ResNet and Inception model is having almost similar accuracy with 70.13% and 69.48% respectively. Densenet is having the least accuracy as well as loss with the figures of 64.29% accuracy and 1.05 loss. The loss for Inception is slightly higher with 1.08 and ResNet is having the highest figure with 1.33.

4.4. Confusion Matrix

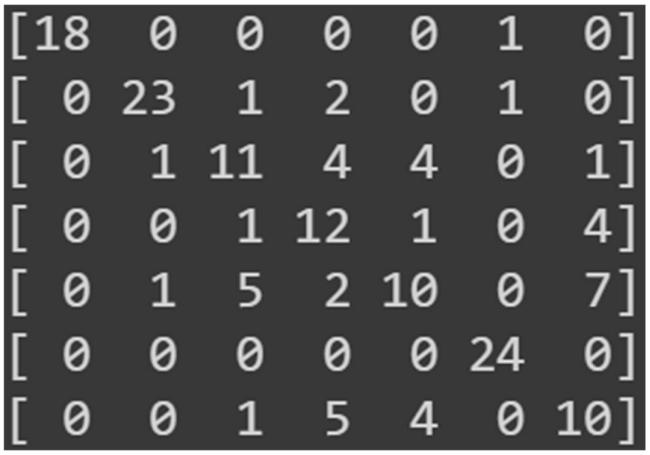

Figure 13 Confusion Matrix ResNet152V2

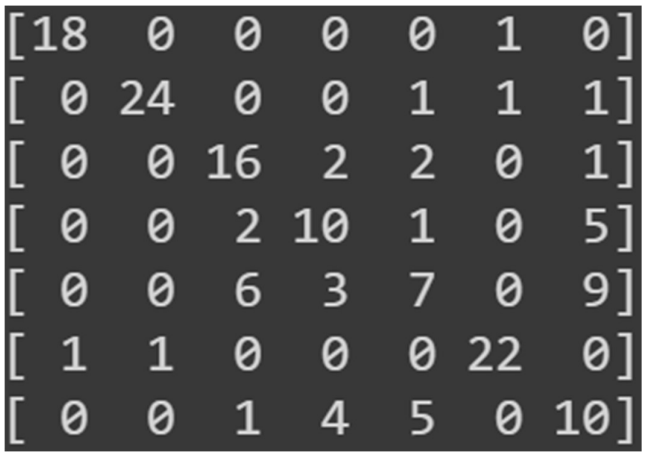

Figure 14 Confusion Matrix InceptionV3

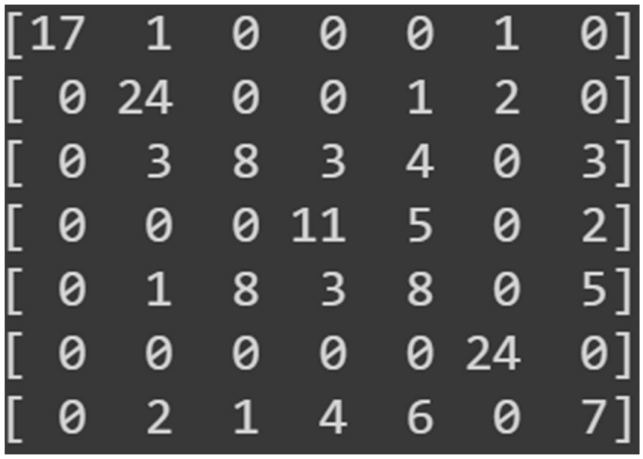

Figure 15 Confusion Matrix DenseNet121

When comparing the 3 confusion matrix the algorithms are able to predict Normal, COVID-19, Pneumonia with minimum error. But when comparing

the rest of the classes the algorithms are still not able to recognize the diseases with good accuracy.

4.5. Precision

	Precision
ResNet152V2	0.71
InceptionV3	0.70
DenseNet121	0.63

The precision values are almost similar to the accuracy values achieved by each algorithm. ResNet shows 71% and Inception shows 70%, but the DenseNet shows only 63%.

4.6. Recall

	Recall
ResNet152V2	0.70
InceptionV3	0.69
DenseNet121	0.64

The recall values also tend to resemble the precision values. As Densenet is having the lowest figure with 64% and Inception and ResNet showing 69% and 70% respectively.

4.7. F1 Score

	F1 Score
ResNet152V2	0.70
InceptionV3	0.69
DenseNet121	0.63

The F1 score is the harmonic mean of the precision and recall values. So F1 scores are similar to both these values, since there is not much difference recall and precision. ResNet has the highest value with 70%, followed by Inception with 69% and finally DenseNet with 63%.

4.8. Probability of COVID-19 affection

Studies involving pulmonary pathology and clinical correlation gave insights into the COVID-19 comorbidity. Although limited investigations focused only on this novel outbreak, we could draw meaningful inferences based on pulmonary disease and nCov2019 chest radiographic findings. Typical computed tomography findings of COVID-19 pneumonia showed the primary characteristics of pulmonary consolidation and bilateral ground-glass opacities (GGOs) (Chung et al. 2020). In a study involving SARS, MERS and COVID-19 patients with pneumonia in Korea, the

proportional distribution of patients with abnormal initial radiographic findings in SARS was 78.3–82.4% (Antonio et al. 2005; Wong et al. 2003), 83.6% in MERS (Das et al. 2015), and COVID-19 pneumonia, 33 percent. In a recent study of radiographic abnormalities of COVID-19 patients in China using chest CT scans, it was found that the grade 2 lesion and one of grade 3 lesions mimic faint GGO and post-inflammatory focal atelectasis, respectively. In 45–67 percent of Chinese COVID-19 patients (Chung et al. 2020; Song et al. 2019), 14–40 percent of MERS patients (Ajlan et al. 2014; Das et al. 2015), and 50% of SARS patients (Wong et al. 2003), GGO lesions were also present on CT. The COVID-19 pneumonia CT findings in Korea were broadly compatible with the COVID-19 pneumonia findings in China and established a favourable link between COVID-19 pneumonia, atelectasis, and consolidation.

5. Conclusion

Three popular algorithms where used to detect between Normal, COVID-19 and 5 lung diseases. The performance between these 3 algorithms where compared and key findings are listed below.

1. More number of training layers always doesn't contribute to better model. This was clearly show by Inception model with 22 million parameters achieving similar accuracy to ResNet with 57 million parameters.
2. All three models tend to overfit the training data. In our case it would be optimal to reduce the number of epochs to somewhere between 15 – 20. This clearly show increased number of epochs doesn't always gives a positive result.
3. Input images with high resolutions tends to produce better predictions by the model. This was clearly shown by all three model even though the images were resized to the smaller resolution, the network predicted the resized images (Normal, COVID, Pneumonia) better than the un-resized images.
4. Different studies show that there is 40% chance of getting affected by COVID-19 if there is any previous lung condition, which is irrespective of age of the individual. Even though further studies should be done to predict more precise values for each disease.

References

Ajlan AM., Ahyad RA., Jamjoom LG., Alharthy A. and Madani TA., 2014. Middle East respiratory syndrome Coronavirus (MERS-CoV) infection. Chest CT findings[online], 782–787.

Algorithmia, 2018. *Introduction to Optimizers.* [Online]
Available at: https://algorithmia.com/blog/introduction-to-optimizers [Accessed 29 May 2020].

Antonio GE., Ooi CG., Wong KT., Tsui EL., Wong JS. and Sy AN., 2005. Radiographic-clinical correlation in severe acute respiratory syndrome. Study of 1373 patients in Hong Kong. Radiology[online], 1081–1090.

Beltrán, J., 2019. *NIH Chest X-rays 224 Rgb.* [Online]
Available at: https://www.kaggle.com/jbeltranleon/nih-chest-xrays-224-rgb [Accessed 02 June 2020].

Brownlee, J., 2019. *How to Choose Loss Functions When Training Deep Learning Neural Networks.* [Online]
Available at: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/#:~:text=The%20mean%20squared%20error%20loss,function%20when%20compiling%20the%20model.&text=It%20is%20recommended%20that%20the,linear%20activation [Accessed 29 May 2020].

Brownlee, J., 2019. *How to use Learning Curves to Diagnose Machine Learning Model Performance.* [Online]
Available at: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/ [Accessed 02 June 2020].

Chung M, Bernheim A., Mei X., Zhang N., Huang M. and Zeng X., 2019. CT imaging features of 2019 novel Coronavirus. Radiology[online], 1148.

Cloud TPU documentation, 2020. *Advanced Guide to Inception v3 on Cloud TPU.* [Online]
Available at: https://cloud.google.com/tpu/docs/inception-v3-advanced [Accessed 25 May 2020].

Das KM., Lee EY., Al Jawder SE., Enani MA., Singh R. and Skakni L., 2015. Acute Middle East respiratory syndrome Coronavirus. Temporal lung changes observed on the chest radiographs of 55 patients[online], 267–274.

Gupta, S., 2017. *Detecting Diseases in Chest X-ray Using Deep.* [Online]
Available at: https://healthcareinamerica.us/detecting-diseases-in-chest-x-ray-usingdeep- [Accessed 15 March 2020].

Huang, G., Zhuang L. and Maaten L., 2017. DenseNets. Densely Connected Convolutional Networks*[online], 3*

News Releases NIH, 2017. *NIH Clinical Center provides one of the largest publicly available chest x-ray datasets to scientific community.* [Online]
Available at: https://www.nih.gov/news-events/news-releases/nih-clinical-center-provides-one-largest-publicly-available-chest-x-ray-datasets-scientific-community [Accessed 15 March 2020].

Pedregosa, F., 2007. *sklearn.model_selection.train_test_split.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html [Accessed 02 June 2020].

Rahman, T., 2020. *COVID-19 Radiography Database.* [Online]
Available at: https://www.kaggle.com/tawsifurrahman/covid19-radiography-database/data [Accessed 09 May 2020].

Saha, S., 2018. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.* [Online]
Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [Accessed 16 May 2020].

Song F., Shi N., Shan F., Zhang Z., Shen J. and Lu H., 2020. Emerging coronavirus 2019-nCoV pneumonia. Radiology[online], 10.

Tensorflow documentation, 2013. *Module: tf.keras.applications.* [Online]
Available at: https://www.tensorflow.org/api_docs/python/tf/keras/applications [Accessed 20 May 2020].

Tsang, S.-H., 2018. *ResNet — Winner of ILSVRC 2015 (Image Classification, Localization, Detection).* [Online]
Available at: https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8 [Accessed 22 May 2020].

Udofia, U., 2018. *Basic Overview of Convolutional Neural Network (CNN).* [Online]

Available at: https://miro.medium.com/max/2800/0*44z992IXd9rqyIWk.png [Accessed 16 May 2020].

Wikipedia, 2020. *F1 score.* [Online]
Available at: https://en.wikipedia.org/wiki/F1_score#:~:text=In%20statistical%20analysis%20of%20binary,measure%20of%20a%20test's%20accuracy.&text=The%20F1%20score%20is,(perfect%20precision%20and%20recall). [Accessed 02 June 2020].

Wong KT., Antonio GE., Hui DS., Lee N., Yuen EH. and Wu A., 2003. Severe acute respiratory syndrome. Radiographic appearances and pattern of progression in 138 patients. Radiology[online], 401–406.