APPLIED ARTIFICIAL INTELLIGENCE (COMP-534)

ASSIGNMENT – III

Solving Image Classification Problems with Convolutional Neural Networks

By,

Kshitij Singh (STUDENT ID: 201602136)

Amol Sahebrao Rathod (STUDENT ID: 201593889)

Akshay Shrinivas Munde (STUDENT ID: 201602056)

PROJECT GOAL: This project consists of two goals: (1) compare the performance of multiple CNN pre-trained models studied in class and (2) propose a new model that can improve the performance of the best pre-trained model.

Introduction

The libraries used in the program were:

1. **Numpy** for mathematical calculations on matrices and arrays.
2. **Pandas** for handling data in the form of DataFrames.
3. **Matplotlib.pyplot** and **Seaborn** for data visualization.
4. **Scikit-Learn** for machine learning tools and algorithms.
5. **Tensorflow** is for tasks related to Deep Learning.
6. **Keras** contains tools for the implementation of artificial neural network building blocks. It provides a python interface for tensorflow.

The aim of this assignment is to compare the performances of 2 pre-trained CNN models (out of 5 given) on an image classification task and then to propose a new model with better and improved performance than the best pre-trained model. The models to choose from were given in the assignment description and out of these we choose the following architectures:

1) **ResNet50V2:** ResNet50V2 is a ResNet architecture with 50 layers and version 2.
The ResNet architecture won ILSVRC 2015 with an astounding error rate of under 3.6% [2]. The winning model had 152 layers instead of 50 which we are using for our tasks. The ResNet uses skip connections (or shortcut connections) , i.e., connection feeding signal to a layer is also connected to the output of the layer located a bit higher up the stack[2]. Then the network is forced to model f(x) = h(x) - x instead of h(x). This is called Residual Learning[2]. This speeds up the training process as in the initial stages of training the output is close to 0 since weights are also 0[2]. But now, because of the skipped connection the output gets the signal from input as well, which in turn makes this an identity function[2]. It is generally seen that the target function is very close to the identity function. So the idea is if we add many of these skip connections, the training time will reduce. That's why in this architecture several Residual Units(RUs) are used. RUs are just a short NN with skip connections[2].

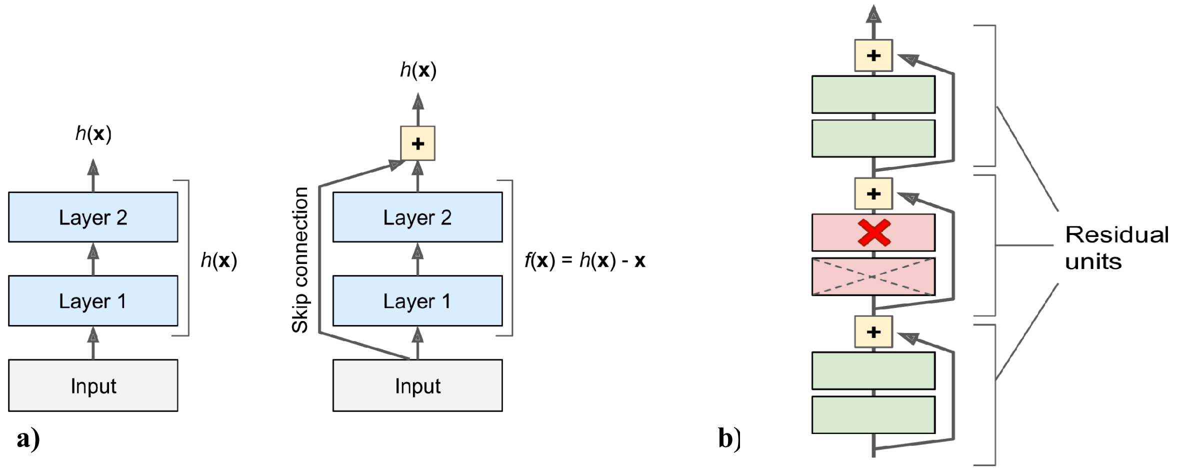
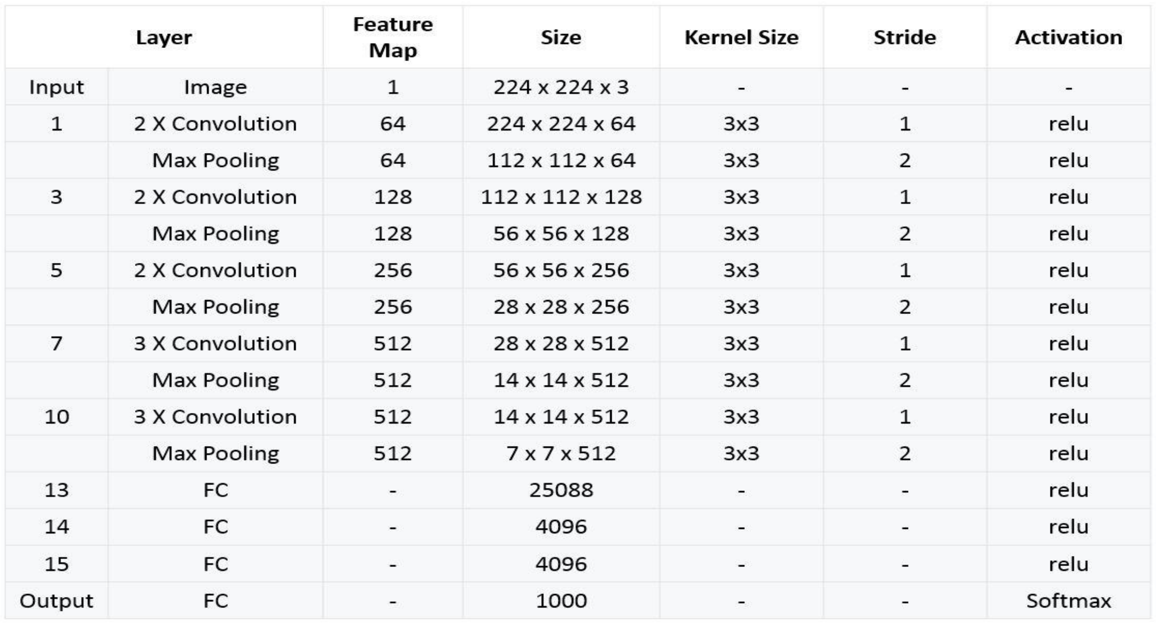


Fig1. a) Illustration of skip connections b) The Residual Units [2]

2) **VGGNet16:** The VGGNet was developed by Visual Geometry Group research lab at Oxford university. It was the runner up in ILSVRC 2014[2]. The advantage of using VGGNet comes

from the fact that it has a very simple architecture as compared to other state of the art architectures[3]. It comes in many variants such as VGG11, VGGNet19 with 11 and 19 layers respectively. VGGNet doesn't have a pooling layer behind each convolution layer but it has 5 pooling layers distributed under different convolutional layers[4].
VGGNet sets itself apart from others as it uses a small sized convolution kernel (3x3) and more layers. Using smaller kernels with multiple layers can reduce parameters as well as provide the model to fit non-linearly which gives it fluidity[4].



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Fig 2. The layers of VGGNet16 architecture.

After selecting the architectures the next task was to implement them in the code. The dataset which we are using is Chest X-Ray Dataset. This dataset contains chest X-rays of healthy patients, patients having Covid-19 and patients having Pneumonia. The aim is to correctly classify the patients.

The dataset had 80% training data and 20% test data. Since we plan to use Cross-Validation, we need to have validation data and the split ratio should be 70:20:10. To do that we take 12.5% of the training data and name it test data. Why 12.5%? 12.5% of training data (80%) will make it 10% of the whole data. Then only validation data is left and the original test can be used for this as it was already 20% of the whole.

The images in the dataset had dimensions Width = 1187 pixels and Height = 1302 pixels
Each pixel had a value ranging [0,255]

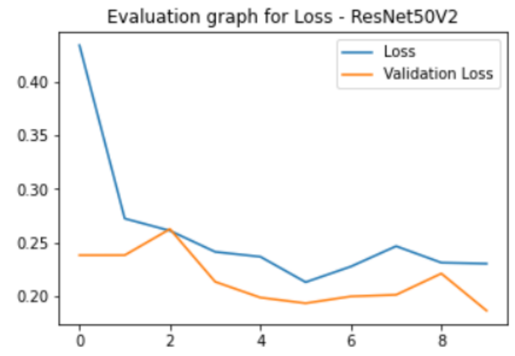
The images were loaded using ImageDataGenerator class which helps in loading data from disk, apply data augmentation, shuffling, resizing to images and even split the dataset accordingly.
The training data was batch wise normalized with batch_size = 10.
The validation and test data were not as the information can be leaked into the model.

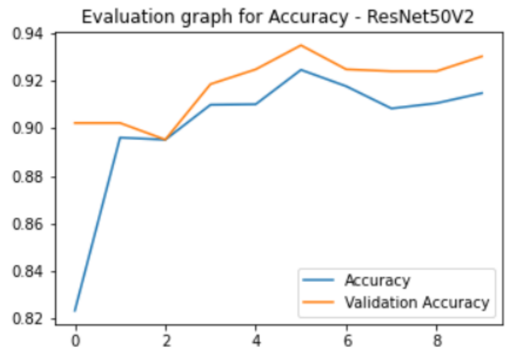
The whole dataset was resized to dimensions (224,224) as both ResNet50V2 and VGGNet16 take input dimensions (224,224)[2].
The models were accessed through Keras. They were called with weights pre-trained from ImageNet, which is an Image Database with more than 1000 classes. Also, the top layers in the models (output layers) were not included as we need to classify between only 3 classes.

Evaluation

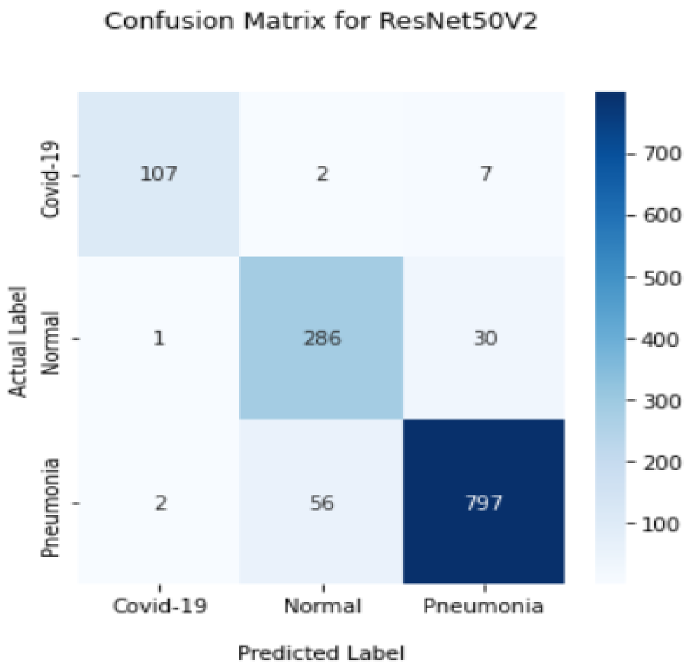
The evaluation is done through metrics such as accuracy graph, loss graph, confusion matrix and classification report.

1) **ResNet50V2:** We added a global pooling layer to the output and fed it to a fully connected dense layer which outputs as the image being in one of the 3 classes as it is in the complete architecture of ResNet50V2. The structure of the model can be seen in the ipython notebook (.ipynb) file.



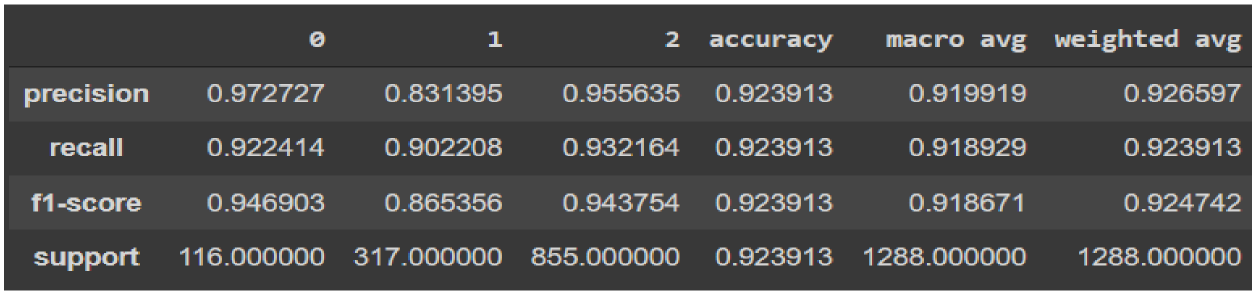


These are the loss and accuracy graphs obtained from ResNet50V2. This data shows that there is no overfitting and the accuracy is also good.



	Covid-19	Normal	Pneumonia
Covid-19	107	2	7
Normal	1	286	30
Pneumonia	2	56	797

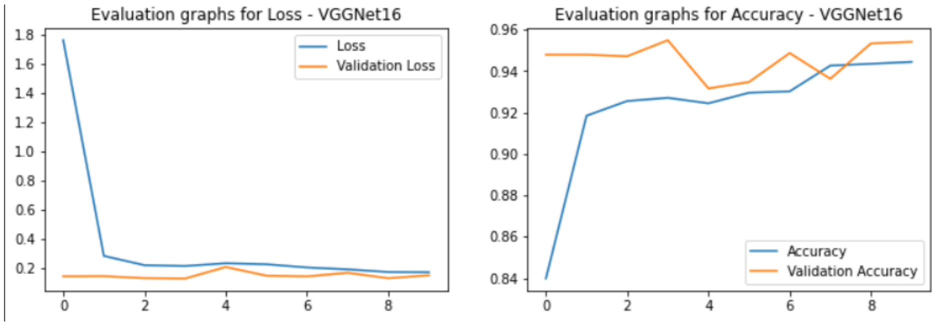
Confusion Matrix:
The data shows that high number of predicted values are accurate



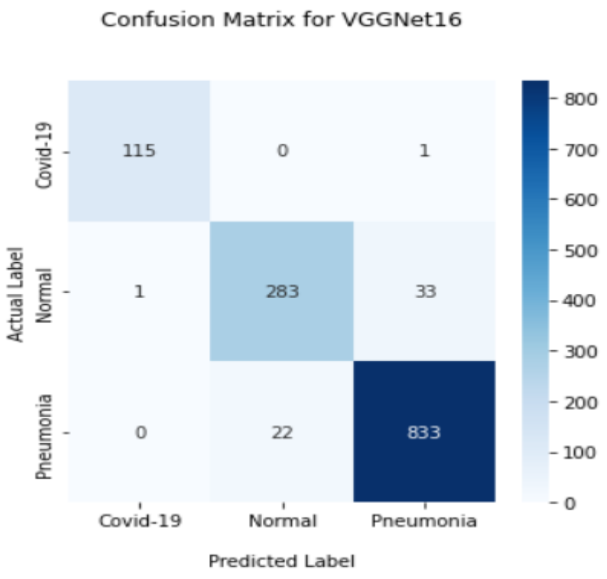
	0	1	2	accuracy	macro avg	weighted avg
precision	0.972727	0.831395	0.955635	0.923913	0.919919	0.926597
recall	0.922414	0.902208	0.932164	0.923913	0.918929	0.923913
f1-score	0.946903	0.865356	0.943754	0.923913	0.918671	0.924742
support	116.000000	317.000000	855.000000	0.923913	1288.000000	1288.000000

[Class 0: Covid-19; Class 1: Normal; Class 2: Pneumonia]
By looking at the classification report above we can find that the precision, recall and f-1 scores for Covid-19 Class(0) are very high.
We can infer from above evaluations that our model is able to to very accurately predict if the patient has Covid-19. Between Normal and Pneumonic patients there are a few cases of false positives and false negatives.

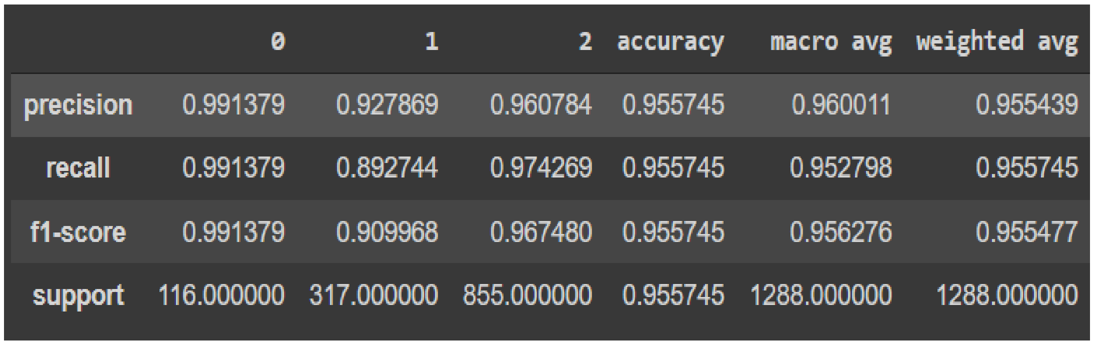
2) **VGGNet16**: If we don't include top layers while calling this class, then we will have to add them manually. So, first a layer with flatten, dense and dropout layers is added, then another layer with dense and dropout which goes into the last dense layer and it predicts the instance in one of the 3 classes. The model summary or structure can be seen in the code ipython notebook (.ipynb) file.



The graph shows at the start of training the loss was high but as the model goes through a few more epochs the score begins to converge with very low loss and high accuracy.



Confusion Matrix:
The distribution is more precise than we saw on ResNet. Covid-19 class has very high precision and recall.



	0	1	2	accuracy	macro avg	weighted avg
precision	0.991379	0.927869	0.960784	0.955745	0.960011	0.955439
recall	0.991379	0.892744	0.974269	0.955745	0.952798	0.955745
f1-score	0.991379	0.909968	0.967480	0.955745	0.956276	0.955477
support	116.000000	317.000000	855.000000	0.955745	1288.000000	1288.000000

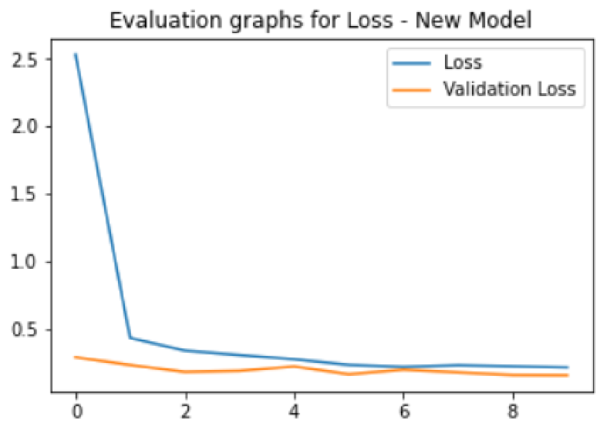
[Class 0: Covid-19; Class 1: Normal; Class 2: Pneumonia]
The classification report shows that the model has high statistics for Covid-19 as we saw in the case of ResNet also. Even for other classes they have improved from what they were on ResNet.

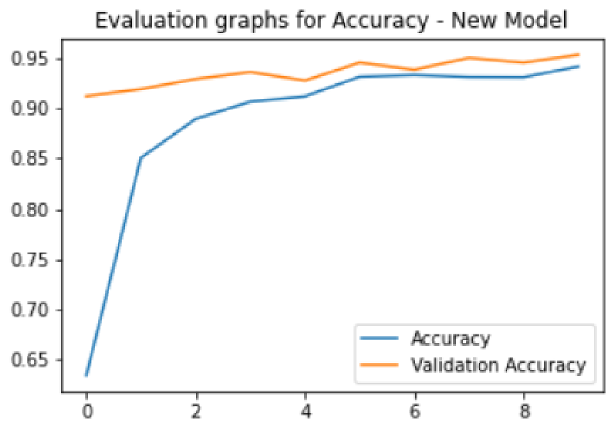
After evaluation, VGGNet has performed better than ResNet for our problem. The evaluation scores and classification scores are also better.

Now the next task for us is to take inspiration from this model and develop over the architecture of this model to try to get improved statistics and evaluation.

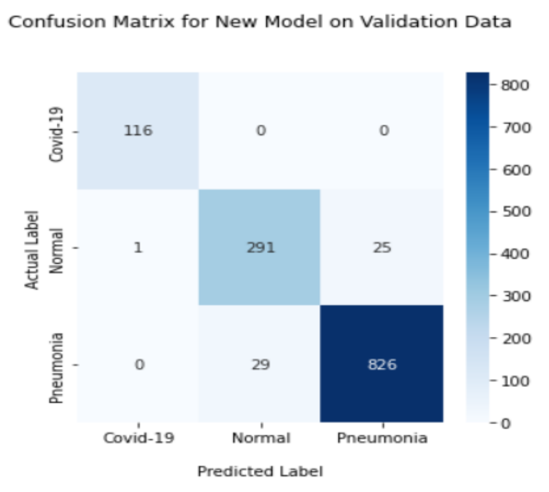
New Model :

IDEA: Add more layers of Dropout and Dense in the output VGGNet16 looks like a good idea The reasoning is that dropout layer will add regularization and adding dense layer will help us in smoothing out the no. of outputs from previous dense layers, .i.e, instead of from 4096 to 512 to 3, addition of dense layers will helps in going from 4096 to 3 a little less abruptly. This will help in preserving the patterns extracted from the image.
So we are making a new model by adding 2 more layers of Dense and Dropout.
The structure of this model can be seen in the ipython notebook(.ipynb) file.

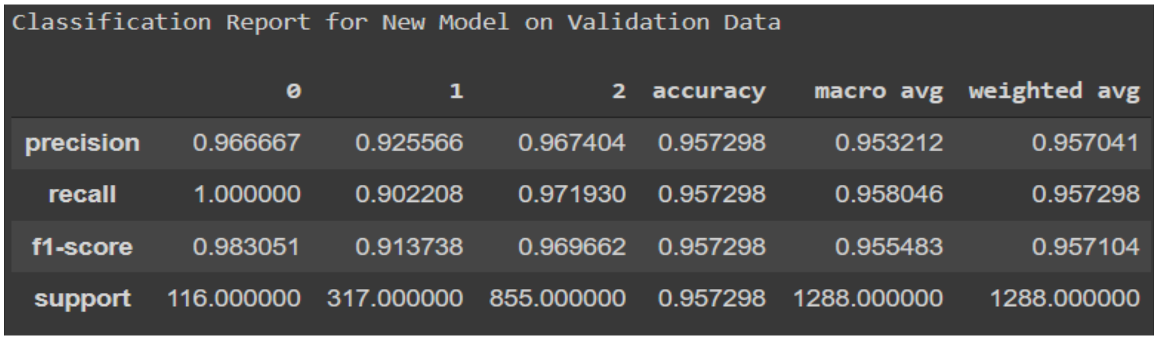




The loss and accuracy graph looks so elegant now, there's less randomness then the previous models. The model has a good generalisability.



Confusion Matrix: Covid-19 scans in validation data have been correctly classified with only 1 false positive and all its actual cases are correctly classified that is it has recall = 1.

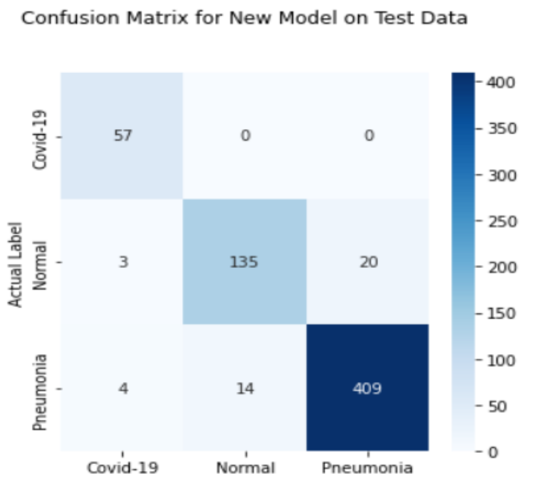


Classification Report for New Model on Validation Data

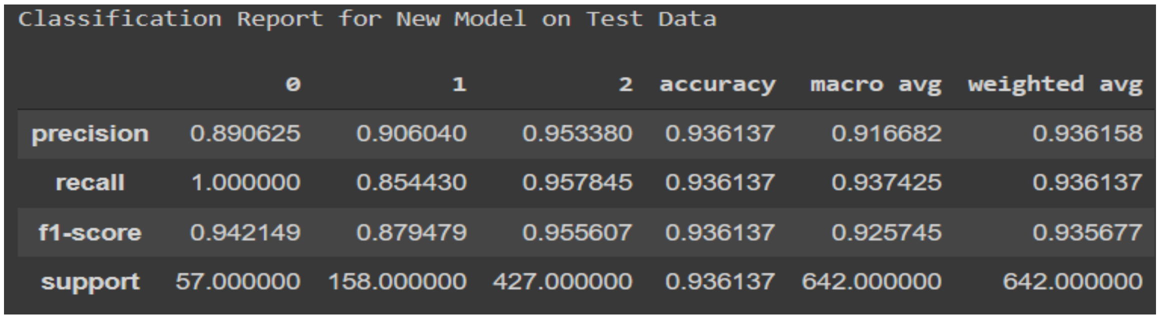
	0	1	2	accuracy	macro avg	weighted avg
precision	0.966667	0.925566	0.967404	0.957298	0.953212	0.957041
recall	1.000000	0.902208	0.971930	0.957298	0.958046	0.957298
f1-score	0.983051	0.913738	0.969662	0.957298	0.955483	0.957104
support	116.000000	317.000000	855.000000	0.957298	1288.000000	1288.000000

[Class 0: Covid-19; Class 1: Normal; Class 2: Pneumonia]
As we can see the recall for Covid-19 class is 1. The classification report also shows that the accuracy has been improved from before in case of validation data. The precision, recall and f-1 scores for all classes have also increased and none is below(0.9).

The metrics show that the model works well with the validation data and the scores have also improved. We can now plug in the test data to see how it works with it.



Confusion Matrix: The trend seems to continue to test data Covid-19 scans as well, the recall is 1. There are a few more false positives in this data.



Classification Report for New Model on Test Data

	0	1	2	accuracy	macro avg	weighted avg
precision	0.890625	0.906040	0.953380	0.936137	0.916682	0.936158
recall	1.000000	0.854430	0.957845	0.936137	0.937425	0.936137
f1-score	0.942149	0.879479	0.955607	0.936137	0.925745	0.935677
support	57.000000	158.000000	427.000000	0.936137	642.000000	642.000000

[Class 0: Covid-19; Class 1: Normal; Class 2: Pneumonia]
The scores on the test data are little less than on the validation data but that is to be expected considering that the model has not seen the data before. The scores have not decreased much which means that our new model is well generalized. The scores of Pneumonic class are very high as compared to other classes since their sample size is also larger.

Conclusion: The new model looks promising as it performs better than both ResNet50V2 and VGGNet16 on chest X-ray image classification. The VGGNet16 outperformed ResNet50V2 as ResNet50V2 had RUs which inputs the identity function to the output. This helped it in achieving high accuracy on the ImageNet Database since the identity function helps the model to focus more on the picture as a whole. But our dataset contained X-ray images which worked well with VGGNet16 as VGGNet has smaller kernels that means it focused on the details more. Hence, it being the inspiration for a new model. The idea behind the new model also comes from the fact that since VGGNet was a simpler architecture than other state of the art models as it achieved high performance with less depth. Then, we can modify it by adding some more layers to enhance its performance.
For the future, we can try testing using a larger dataset for our model. We can also try changing the kernel size of the top few layers to (1x1) from (3x3) as this can make the model learn minute details.

Final Conclusions

- One of the many challenges we faced was managing the schedule and meetings because we had different modules this semester and we had to rely on online meetings which were a little less productive.
- While developing the program we had to give a lot of time for understanding and comparing the networks(VGGNet and ResNet).
- Another challenge we faced was we did not have a higher computing power so it was taking longer to train the models.

Tasks Allocation:

Downloading the Dataset	Amol Rathod
Analysis of Images: Quantitative	Kshitij Singh
Analysis of Images: Qualitative	Akshay Munde

Preparing the data for Model	Amol Rathod
ResNet50V2 Training	Kshitij Singh
ResNet50V2 Evaluation	Kshitij Singh
VGGNet16 Training	Akshay Munde
VGGNet16 Evaluation	Akshay Munde
Creating a new CNN Model: Training	Kshitij Singh
Creating a new CNN Model: Evaluation	Kshitij Singh
Report	Kshitij Singh, Amol Rathod, Akshay Munde

References

[1] Rahimzadeh, M., Attar, A. (2020). 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. https://doi.org/10.1016/j.imu.2020.100360
[2] Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (Book)
[3] He, K., Zhang, X., Ren, S., Sun J. (2015). Deep Residual Learning for Image Recognition. https://arxiv.org/abs/1512.03385v1
[4] VGGNet-16 Architecture: A Complete Guide