Deep Learning based approach on Health Care Domain

Here, In this project we have developed few deep learning models to classify the person as Covid19, Pneumonia, or normal healthy patient. Initially we have used the pre-trained models in order to train against the dataset. We have used Convolution Neural Networks (**CNN**) against Artificial Neural Networks (**ANN**). Because we will lost the spatial information in 2D image when they flattened for ANN. Also the trainable parameters will be high so because of that computational complexity will be also high. So, ANN is cost ineffective. Also the pre-trained models we have used in our training is AlexNet and GoogleNet which are famously known for image classification. Many Researches provided that GoogleNet is the best use in case of any image classification problem statement. So we are going to prove this with the obtained result at the end.

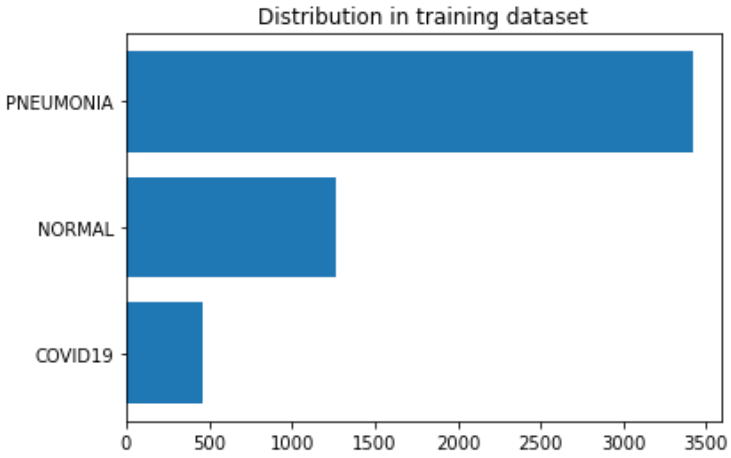
Libraries used - Numpy, Pandas, Matplotlib, Warnings, Tensorflow, Keras, os, re.

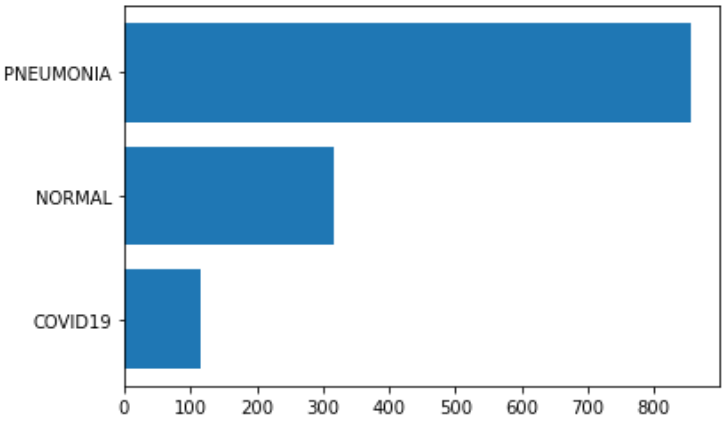
IDE used - Jupyter Notebook.

The Process we have followed is:

1) Data Understanding
2) Data Cleaning
3) Data Visualization
4) Model Building
5) Model Evaluation
6) Comparing Results
7) Improving the best model

The data distribution in train and test are:





Given dataset contains two folders that was given for test and train separately. Now we have observe that each image is of different shape so this may create a problem while training. So first we have to convert image to our custom image size. Now every image is of same shape. Here we are using generator concepts to get the train data and test data. Also we are rescaling the data with the desired factor. So that our train data is ready for model training.

As said above, We used AlexNet and GoogleNet in our model training. Lets talk about AlexNet now.

Basically AlexNet is a deep architecture. Here the input image is of size (250,250,3). Now we are having first convolution layer with 96 filters with activation function as ‘relu’. Now the output of the convolution layer is determined by formula

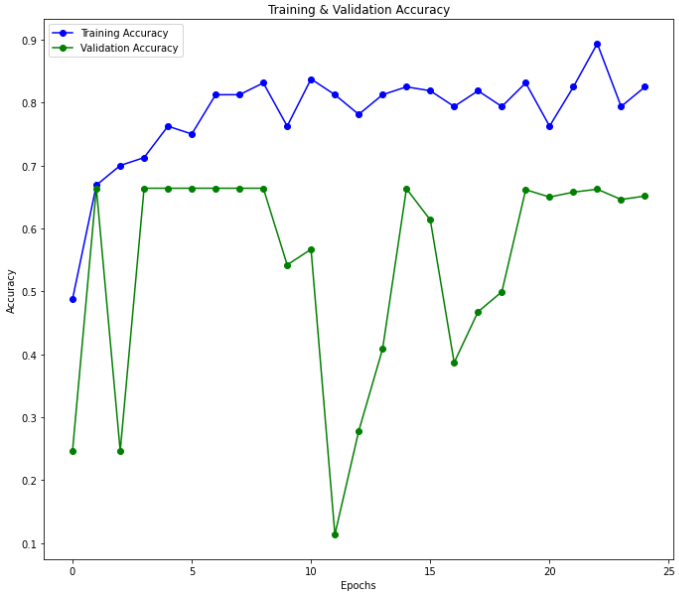
Output of the convolution layer = ((Input_size-filter_size)/stride)+1

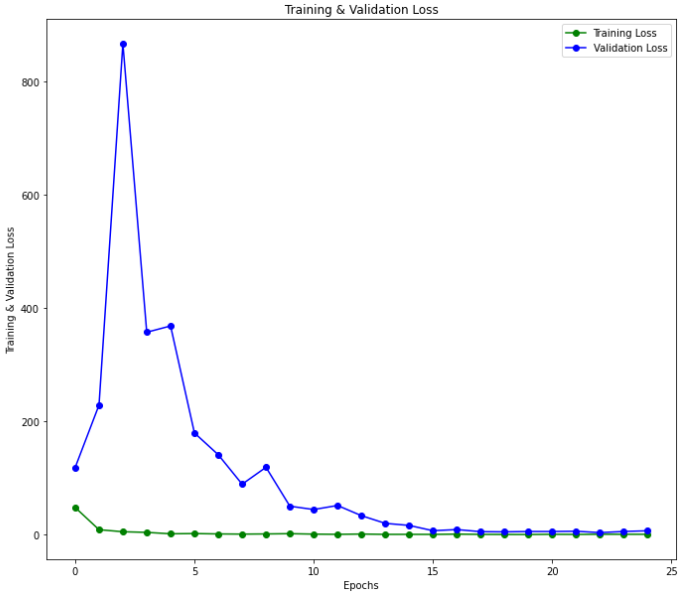
Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Following by next, we are applying the max-pooling layers. Basically it is used to reduce the size of the image. And next again a convolution layer and max-pooling layer. Next followed by three convolution layers and one max pooling layers. Finally we are giving dropout as 0.5. Basically the major problem in neural networks is overfitting. In case of machine learning, we use regularization techniques such as lasso and ridge. In deep learning we have dropout to avoid overfitting. Basically, dropout is leaving some neurons away from training. The activation we used is “Relu”. Next we have flatten the image and given as input for FC(fully connected) layers. And output layer contains 3 neurons because we are having three different classes such as ‘Normal’, ‘COVID19’, ‘PNEUMONIA’. Here it is the categorical problem statement so we used as `‘categorical_crossentropy’` as loss function and `'adam’` as optimizer. Adam is the best optimizer because it take cares of the momentum and learning_rate also gets updated. In the output layer we used `‘softmax’` as activation function as it is multi class in label column. Total 8 layers with the learnable parameters and 3 full y connected layers and activation function is relu and 5 convolution layers along with the max-pooling layers. Note that input is RGB Images. Total trainable parameters in our training is : `58,299,139.`

```
Training accuracy is: 0.82
Testing accuracy is : 0.65
```

Results of AlexNet is shown below: (epochs=25 and steps_per_epoch=5)





We can conclude that, validation loss goes on decreasing. This is most significant one. But testing accuracy was not proper (unstable). Due to this drawback we came to GoogleNet also known as Inception. Let us talk about it now.

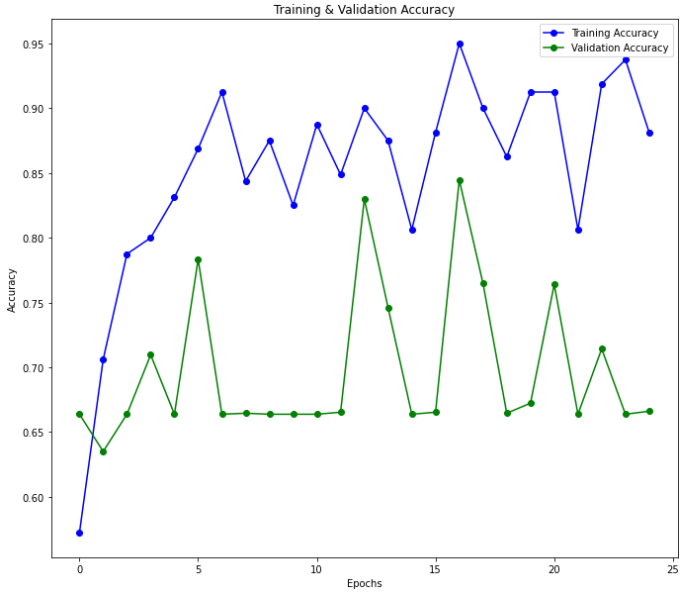
Architecture of Inception is:

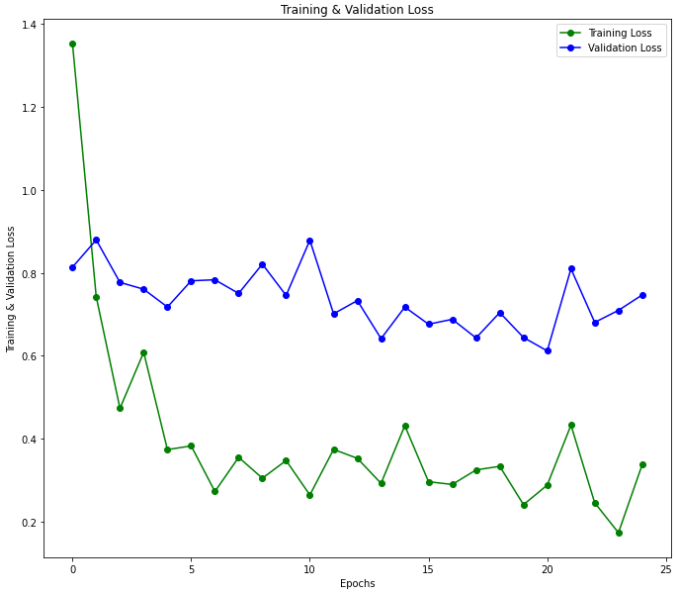
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

GoogleNet is a new type of architecture also known as Inception. It is a basically a convolutional neural network with 27 layers deep. Here also we are using activation function as relu and softmax in ouput layer as it is related to multiclassification report. Here author of the GoogleNet calls a term known as ‘Hebbian Principle’. Basically it says that “Neurons that fire together, wire together”. Major focus of this algorithm is, while creating a layer we have to keep an eye on the learnings of the previous layer. Here we have optimizer as ‘RMS Prop’. And learning as 0.001. And loss is as earlier as before. Metrics to evaluate is accuracy.

```
Training accuracy is: 0.74
Testing accuracy is : 0.67
```

```
Results obtained in case of GoogleNet Classifier is:
```

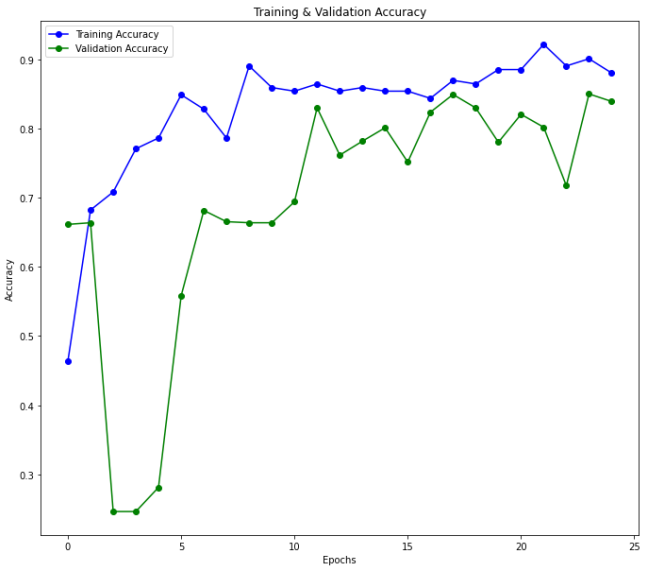


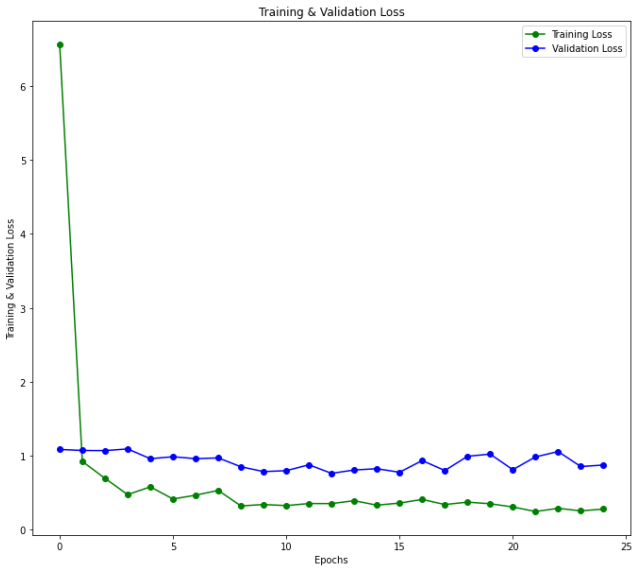


So from above two pre-trained networks we can say that “GoogleNet” is better to use. Also GoogLeNet was the winner at ILSRVRC 2014 taking 1st place in both classification an detection task. It has top-5 error rate of 6.67% in classification task. So experimental results are proven that googlenet classifier is best. So let us improve that algorithm now.

In order to increase the performance , we can use optimizer as ‘adam’. Adam is best optimizer so we are replacing RMS prop by adam. Also applying batchnormalization increases accuracy a bit. After changing this all, we are getting accuracy as:

```
Training accuracy is: 0.88
Testing accuracy is : 0.84
```





From the above picture we can depict that, training and testing accuracy is quite stable. Loss is also getting low in our case. We are using dropout inorder to avoid overfitting. Therefore it is best algorithm to use now.

In order to increase the performance more better, add images in COVID19 and NORMAL folder in train dataset. So the data would be more balanced and we will get results more appropriate.

Final Conclusion:

During the project, it has been observed that code is time taking process. We have to make sure that input image size is passed correctly and mentioned correctly in input layer also. Make sure to use Softmax only instead of sigmoid to avoid low performances. Main challenge will observed while training the results and make sure that filters in convolution neural network is fine and enough. Each training in pre-trained model long last up to lot of time. So pls be patient while running the code. Main challenge occurred in case of this project is writing code for pre-trained model in order to understand the results better. A README file is attached to how to run the file. So please follow the instructions accordingly.