Machine Learning Engineer Nanodegree

Capstone Project

Subramanian Vellaiyan
Aug 29th, 2018

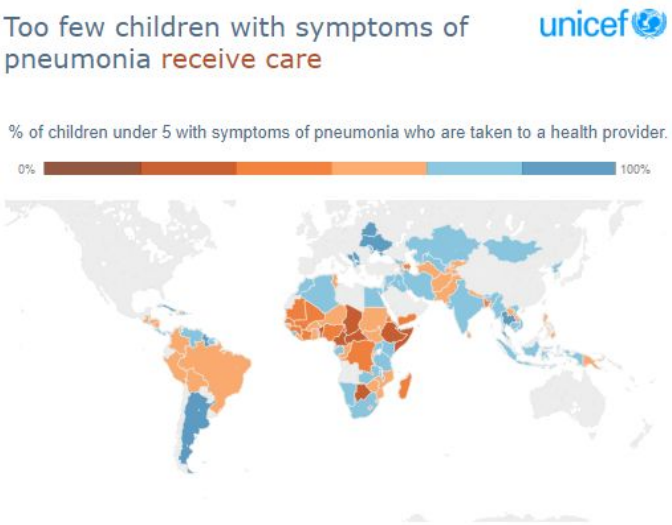
I. Definition

Project Overview

The project proposal is Pneumonia Classification based on Chest X-Ray Images. The lungs fill with sacs of fluid and make breathing difficult. Pneumonia can affect any age group. Signs of pneumonia are a combination of respiratory symptoms, including 'cough and fast or difficult breathing due to a chest-related problem'.

Top Stats:

- Pneumonia is the world's leading cause of death among children under 5 years of age, accounting for 15% of all deaths of children under 5 years old.
- Pneumonia is the #1 most common reason for US children to be hospitalized.
- Viral pneumonias are the leading cause of hospitalization of infants



Motivation

- Chest X-rays are the most common imaging examination and if a machine can tell that there is a probability of Pneumonia before going to Doctor/Radiologist would help the person/guardian to take immediate measure.
- Radiologists to patient ratio might not be the same in all parts of the world. So a deep learning solution would help to detect the disease and send to the respective health provider at the earliest.
- Deep Learning/CNN - ability to solve real world problems. I am highly motivated to delve deep into this subject and understand an end-to-end process of image classification using CNN.

Problem Statement

The problem statement is to classify a Chest X-Ray image into the below three levels

- Lung Opacity
- No Lung Opacity / Not Normal
- Normal

RSNA(Radiological Society of North America) has hosted a Pneumonia detection challenge in Kaggle. The data-set for this problem, has been taken from this competition.

Metrics

The core of understanding the performance matrix for a model is in the basic **confusion matrix**, which tracks the **true positives**, **true negatives**, **false positives**, and **false negatives**. We have a multi class model so let's take the example of opacity. In our case, a true positive is when we correctly predict a case as opacity, while a true negative is when we predict the normal or not normal. False positive is the reverse; when we predict opacity, but it does not occur. A false negative is when we predict no opacity, but then the case is a opacity.

Accuracy

The most basic measure of model performance is accuracy. It simply tells you what percentage of all your predictions were correct.

Formula = (true positive + true negatives) / total prediction

However, accuracy can be misleading, because we are dealing with a relatively rare event. If we classify all the cases as either normal or not normal, we might still have 66.67% so we would consider this metric as the last important metric.
There are two other metrics - **Recall** and **Precision** - which are more useful in this case.

Recall

Recall tells you how many of the total opacity cases were successfully predicted. If there are 10 opacity cases and we predict 7 of those, then we have 70% recall. Of course, if I predict opacity every time, then I will generate perfect recall, but very poor overall accuracy.

Formula = (true positive) / (true positive + false positive)

Precision

Precision balances recall by telling of how many of the times which we predicted opacity, we were correct. So, if we predict opacity 10 times, but we only see opacity 7 times, then we have 70% precision.

Formula = (true positive) / (true positive + false negative)

F1

So we want to have both high recall and high precision. Fortunately, there is another metric which combines these two called **F1**. A strong F1 score tells you that a model is generating both good recall and good precision. That is, your model correctly predicts a large percentage of opacity and rarely predicts opacity incorrectly.

$$Formula = (1 + \beta^2) \times (precision \times recall) / \beta^2 \times precision \times recall$$

So as explained in the benchmark model, F1-score would be considered the prime metric along with Recall and Precision to evaluate the model for all the classes.

II. Analysis

Data Exploration

The input consists of train and test images in DICOM format. Each image has the patient id in the image name.

File descriptions

- stage_1_detailed_class_info.csv - provides detailed information about the type of positive or negative class for each image.
- Stage_1_train_images - Contains 25K train images with DICOM format
- stage_1_train.csv - the training set. Contains patient Ids and bounding box / target information.

Each dcm image has its own metadata. A sample below shows how the image's metadata looks like

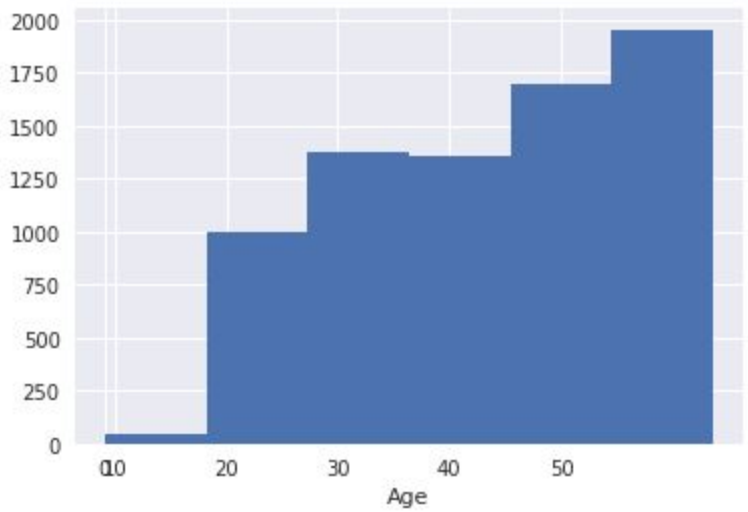
```
(0008, 0005) Specific Character Set             CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                      UI: Secondary Capture Image Storage
(0008, 0018) SOP Instance UID                   UI: 1.2.276.0.7230010.3.1.4.8323329.6379.1517874325.469569
(0008, 0020) Study Date                         DA: '19010101'
(0008, 0030) Study Time                         TM: '000000.00'
(0008, 0050) Accession Number                   SH: ''
(0008, 0060) Modality                           CS: 'CR'
(0008, 0064) Conversion Type                    CS: 'WSD'
(0008, 0090) Referring Physician's Name         PN: ''
(0008, 103e) Series Description                 LO: 'view: AP'
(0010, 0010) Patient's Name                     PN: '00436515-870c-4b36-a041-de91049b9ab4'
(0010, 0020) Patient ID                         LO: '00436515-870c-4b36-a041-de91049b9ab4'
(0010, 0030) Patient's Birth Date               DA: ''
(0010, 0040) Patient's Sex                      CS: 'F'
(0010, 1010) Patient's Age                      AS: '32'
(0018, 0015) Body Part Examined                 CS: 'CHEST'
(0018, 5101) View Position                      CS: 'AP'
(0020, 000d) Study Instance UID                 UI: 1.2.276.0.7230010.3.1.2.8323329.6379.1517874325.469568
(0020, 000e) Series Instance UID                UI: 1.2.276.0.7230010.3.1.3.8323329.6379.1517874325.469567
(0020, 0010) Study ID                           SH: ''
(0020, 0011) Series Number                      IS: '1'
(0020, 0013) Instance Number                    IS: '1'
(0020, 0020) Patient Orientation                CS: ''
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0010) Rows                               US: 1024
(0028, 0011) Columns                            US: 1024
(0028, 0030) Pixel Spacing                      DS: ['0.139', '0.139']
(0028, 0100) Bits Allocated                     US: 8
(0028, 0101) Bits Stored                        US: 8
(0028, 0102) High Bit                           US: 7
(0028, 0103) Pixel Representation               US: 0
(0028, 2110) Lossy Image Compression            CS: '01'
(0028, 2114) Lossy Image Compression Method     CS: 'ISO_10918_1'
(7fe0, 0010) Pixel Data                         OB: Array of 119382 bytes
```

This metadata gives us some patient information along with image information. This data is just used to understand the type of cases we are studying and these are not used in model. Stage_1_detailed_class_info.csv file is taken as the input and it has been used to fetch images from train images folder.

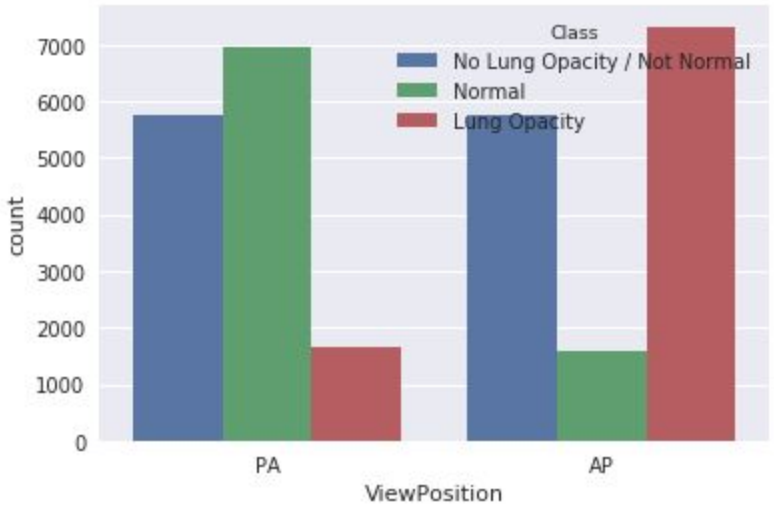
Exploratory Visualization

The metadata of DICOM images have been read into data frame and below is some EDA to understand the use case better.

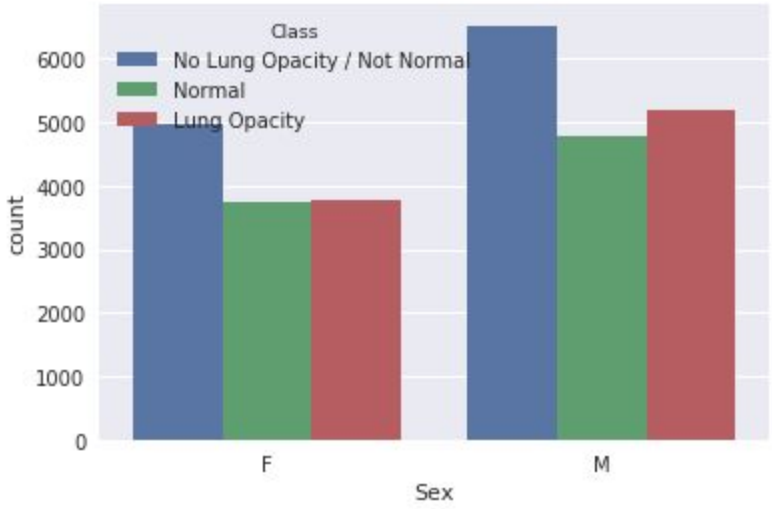
The below distribution depicts the Lung Opacity amongst different age groups. It is to be noted that, as the age increases the pneumonic cases increase in a linear fashion.



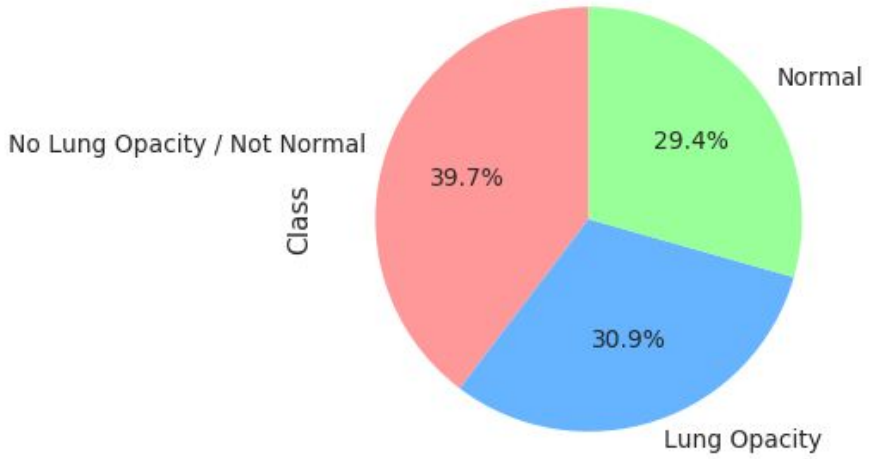
The View Position is another factor which may affect the model. Even though the distribution is even amongst the total images, when compared to target classes AP position seem to have extremely high Lung Opacity cases and PA position seems to have more Normal cases.



The below image shows the histogram of ‘Sex count’ for different class.Male cases are more than in number than female. Sex can play a major role in classifying the output when trained separately as they have different body structure. But this needs some further exploration to prove the statement.



The input taken for our modelling is split by proportion and below is the pie chart showing the percentage of data taken for modelling. Not Normal cases may outweigh Normal and Lung opacity by 10%.



Algorithms and Techniques

Modelling

Keras API has been used for modelling with TensorFlow backend.
The base model is a pre-trained neural network with top layer set to False. The fully-connected layer at the top of the network is not required as we have a custom requirement to classify three levels which would be covered in the next section Fine Tuning. This step is also called as the feature extractor. This step downloads a model which is pre trained on imagenet database. This step would reduce training time heavily as we are using pre-trained weights.

Let's consider the example of VGG 19 which has has 19 weight layers as depicted below and the total number of parameters that has been trained will be 20,024,384. Trainable parameter is set to False as these parameters need not be re-trained. Below is an image which shows both VGG16 and VGG19 which is D and E respectively.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Image processing

Libraries: cv2, pydicom, numpy
cv2 has been used to resize the image.
Preprocess_input from keras.applications to adjust the mean values of RGB
pydicom has been used to read the image in pixel array. Numpy package has been used for the transformation of 2D array into 4D tensor.

Benchmark

The goal of the project is to classify whether a case is affected by Pneumonia or not and also determine if it has signs showing that it has Pneumonia(cases where it is not normal). Identified cases would be moved to a higher priority in the queue for treatment at health centers. A benchmark model would be to achieve at least 70% F-1 score so that both precision and recall is balanced. We don't want the model to classify all the cases as pneumonic and increase the queue of patients and on the other hand we don't want the model to classify all the cases as non pneumonic. This way we can build a model that balances between Precision and Recall.

To create a benchmark analysis - an Xception model was used to do the transfer learning and fine tuning with three layers to do the classification part. The model was run with the settings mentioned below and all these parameters will be explained later in Refinement section.

Run ID	4
learning_name	XceptionTransferLearning
total_images	1450
input_shape	(1024,1024,3)
model_type	FC1024
dropout	0.2
optimizer_type	Adam
learning_rate	0.0001
epochs	30
batch_size	20
rescale	1
test_accuracy	56.5517

Classification Report

Class	precision	recall	f1-score	support
No Lung Opacity / Not Normal	50.00%	50.00%	50.00%	115
Lung Opacity	59.00%	62.00%	60.00%	86
Normal	62.00%	60.00%	61.00%	89
avg / total	57.00%	57.33%	57.00%	290

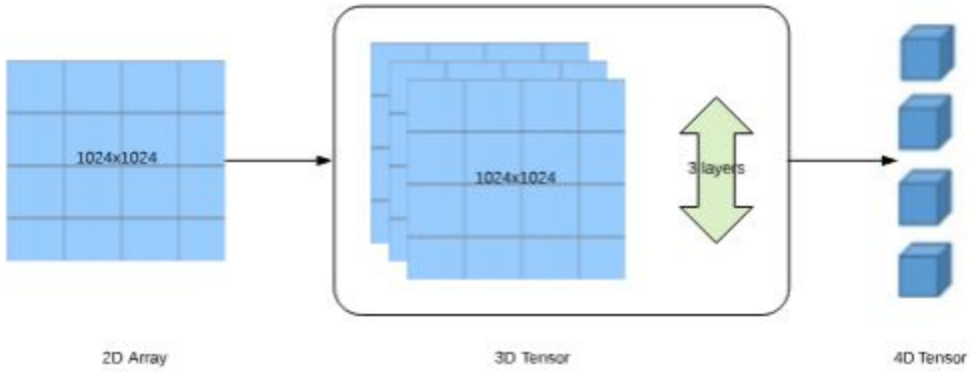
For initial analysis the result looked

III. Methodology

Data Preprocessing

The input data is a set of Chest X-Ray images in DICOM format. The size of each image is 1024x1024 pixels. Keras with tensorflow backend is being used. So the input has to be a 4D tensor with the shape (samples, rows, columns, channels). A python library - pydicom is used to

load the image into pixel array (1024,1024). The image is in grayscale which is just 1D so it has to be converted to RGB mode which is 3 layers. To achieve that the numpy array is repeated thrice to get 3 channels but all the 3 channels will have the same values. Now we have each image with shape (1024,1024,3), when this is stacked for multiple images it is converted into a 4D tensor.



Now the data is ready to be fed into Keras. But there is one more step for supplying the model to pre-trained network which is normalization of mean pixel expressed in RGB which will be implemented by the preprocess_input function within each keras application.

After that there are 3 optional data processing steps which can be used in tuning the model. These can turned ON/OFF before running the script by setting the respective indicators.

Image Augmentation

This is done using the keras ImageDataGenerator function and only few parameters have been used and others are left at their default values.

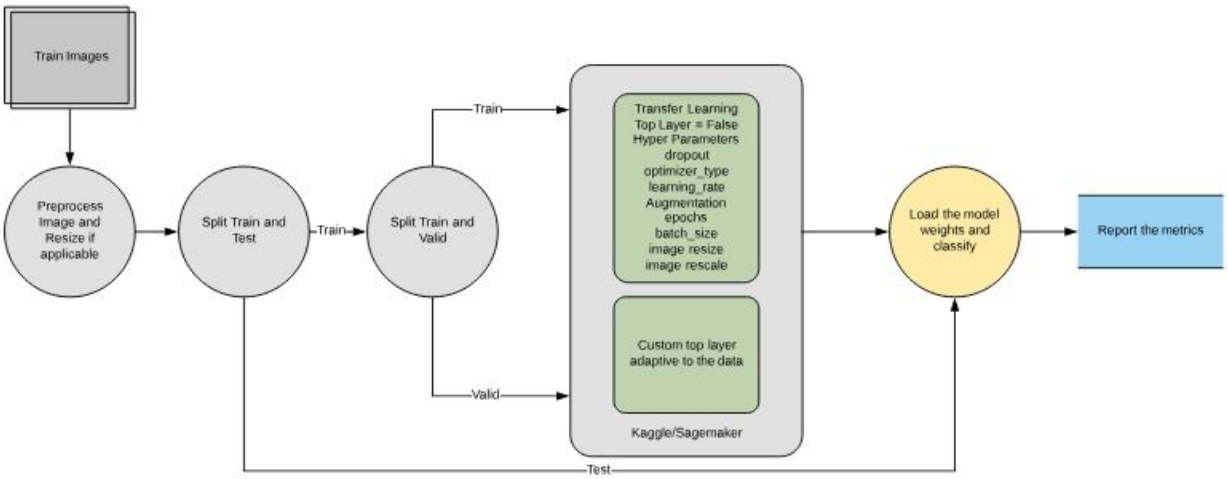
Image Resize

The image size which 1024x1024 which is too big to process so this option allows us to resize the image so that we can process more data.

Pixel Value scaling

The pixel values are already re-scaled to value between 0 and 1 from preprocess_input function. But this option might be useful if we want to scale to different values and see how it impacts the model's performance.

Implementation

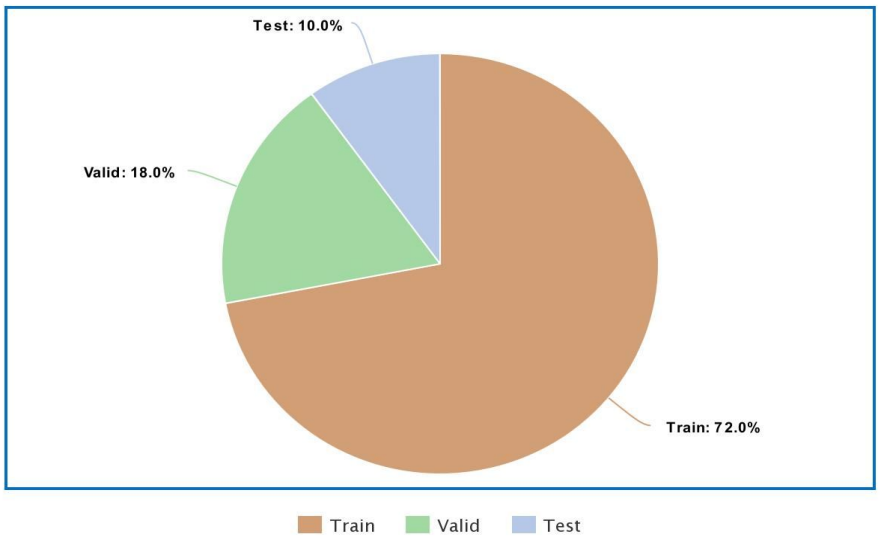


For the modelling we are going to take only a sample out of the 26000 images. This can be configured with the sample ratio, i.e., sample_ratio 0.35 would have around 9100 images for the modelling.

Split data

Once the data is preprocessed and resized in required format, the data is split into train, valid and test with stratified sampling. Below are the ratios of the train, valid and test.

Train: 72% Valid: 18% Test: 10%



Transfer Learning

For our implementation we have taken 3 types of pre-trained neural networks. They are:

- Xception

  It is an extension of the Inception architecture which replaces the standard Inception modules with depth-wise separable convolutions.

- VGG16
- VGG19

  Both VGG16 and VGG19 have similar architecture. This network has only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4096 nodes are then followed by a softmax classifier.

Fine Tuning

We have the pre-trained neural network with top layer set to False. Since the top fully connected layer has been removed,there has to be a custom top layer to classify the three classes. We have multiple options that can be experimented to see how each one affects the model's performance. Below are a few options:

FC - Type1

This type consists of 5 layers.. The pre-trained network is the main feature extractor and the set of layers below will be the classifier.

1. Batch Normalization
2. Dense with dropout - 256
3. Global average pooling 2D
4. Dense - 128
5. Dense - 3 (soft-max)

FC512 - Type2

This type consists of 12 layers. These layers may have some additional impact with feature extraction of Chest X-Rays as they are configured with 3 Convolution layers and 1 dense layer. The training might consume more time than the previous one as the additional parameters have to be trained.

1. Conv2D - 1024
2. Batch Normalization
3. Dropout
4. Conv2D - 512
5. MaxPooling2D
6. Conv2D - 256
7. Dropout
8. Batch Normalization
9. Dense - 256

10. Dropout
11. Global average pooling 2D
12. Dense - 3 (soft-max)

FC1024 - Type3

This type is similar to the second type, but consists of an additional layer except that there is no dense (256 nodes) layer towards the end. So there would be four Convolution layers.

1. Conv2D - 1024
2. Batch Normalization
3. Dropout
4. Conv2D - 512
5. Dropout
6. MaxPooling2D
7. Conv2D - 256
8. Dropout
9. Batch Normalization
10. Conv2D- 256
11. Dropout
12. Global average pooling 2D
13. Dense - 3 (soft-max)

Environment

To implement this project there were two platforms chosen, Kaggle's kernel and AWS Sagemaker. This project requires GPU access with at least 16GB RAM. So most of the runs have been executed in Kaggle and only a few have been executed in AWS Sagemaker.

Refinement

In the initial analysis in the benchmark section the model was reported with 57% F-1 score. Below are some steps followed to boost the model's performance.

Image Resize

The image resizing helped to bring more data as with the initial image size the model was consuming too many resources and as a result only a sample size close to 1500 images was fed into model. After the image resizing, training bandwidth was able to go 11.5K images.

Re-scaling Pixel

The pixel value is already re-scaled to a value between 0 and 1. By scaling the values to a higher value the model's performance got boosted. The most probable reason being, the grayscale being differentiable with a higher pixel value.

Batch Normalization

Batch normalization was added in the custom top layer. This normalizes the output of previous activation layer and help stabilizes the neural network.

Augmentation

Augmentation had little to any improvement in the final results. So in this case avoiding augmentation simplifies the model and also saves time.

Hyperparameters

Dropout

As dropout increases, the F-1 score is also found to improve. Maximum dropout that can be set is at 0.45 as after that it would have no effect. This helps reduce the overfitting of the model.

Epochs and Batch Size

The batch size had a trade off at 15-16 where it was optimal. Epochs greater than 15 did not have any effect as the validation loss did not improve beyond a point. So we can achieve the same results with less number of epochs.

Optimizer and learning rate

Four different type of optimizers have been tried with a learning rate of 1e-4. Adam and Adadelta have been found to perform better than others.

IV. Results

Model Evaluation and Validation

Thirty nine runs have been performed with varied configuration of parameters. All the results are attached in the file named 'run_details.xlsx.' Sorting through the results, the best model for each case is listed below.

The models did well coming up with a good score in classifying the normal cases and decently well in Lung Opacity cases with 70% F-1 score. In all the cases dropout is 0.4 - 0.45. Epochs and batch size remain the same amongst the top models. Input shape is a resized image with dimensions 256x256. VGG was the clear winner when compared to Xception in this case.

Thus from the above observation, it can be noted that a best model would be a combination of features listed below and can guarantee an F1-score of 70% - 80%

Normal

Run ID	32
learning_name	VGG19TransferLearning
total_images	10147
input_shape	(256,256,3)
model_type	FC514
dropout	0.4
optimizer_type	Adam
learning_rate	0.0001
epochs	15
batch_size	16
test_accuracy	69.95073892
rescale	100

Class	precision	recall	f1-score	support
No Lung Opacity / Not Normal	66.00%	64.00%	65.00%	415
Lung Opacity	58.00%	75.00%	65.00%	245
Normal	88.00%	74.00%	81.00%	355
avg / total	70.67%	71.00%	70.33%	1015

No Lung Opacity / Not Normal

Run ID	37
learning_name	VGG16TransferLearning
total_images	11596
input_shape	(256,256,3)
model_type	FC514
dropout	0.45

Class	precision	recall	f1-score	support
No Lung Opacity / Not Normal	71.00%	66.00%	69.00%	388
Lung Opacity	49.00%	68.00%	57.00%	329

optimizer_type	Adam
learning_rate	0.0001
epochs	15
batch_size	16
test_accuracy	68.36206897
rescale	100

Normal	92.00%	71.00%	80.00%	443
avg / total	70.67%	68.33%	68.67%	1160

Lung Opacity

Run ID	30
learning_name	VGG19Transfer Learning
total_images	10147
input_shape	(256,256,3)
model_type	FC1024
dropout	0.4
optimizer_type	Adadelta
learning_rate	0.0001
epochs	15
batch_size	16
test_accuracy	68.1773399
rescale	100

Class	**precision**	**recall**	**f1-score**	**support**
No Lung Opacity / Not Normal	57.00%	62.00%	59.00%	371
Lung Opacity	75.00%	67.00%	71.00%	354
Normal	76.00%	78.00%	77.00%	290
avg / total	69.33%	69.00%	69.00%	1015

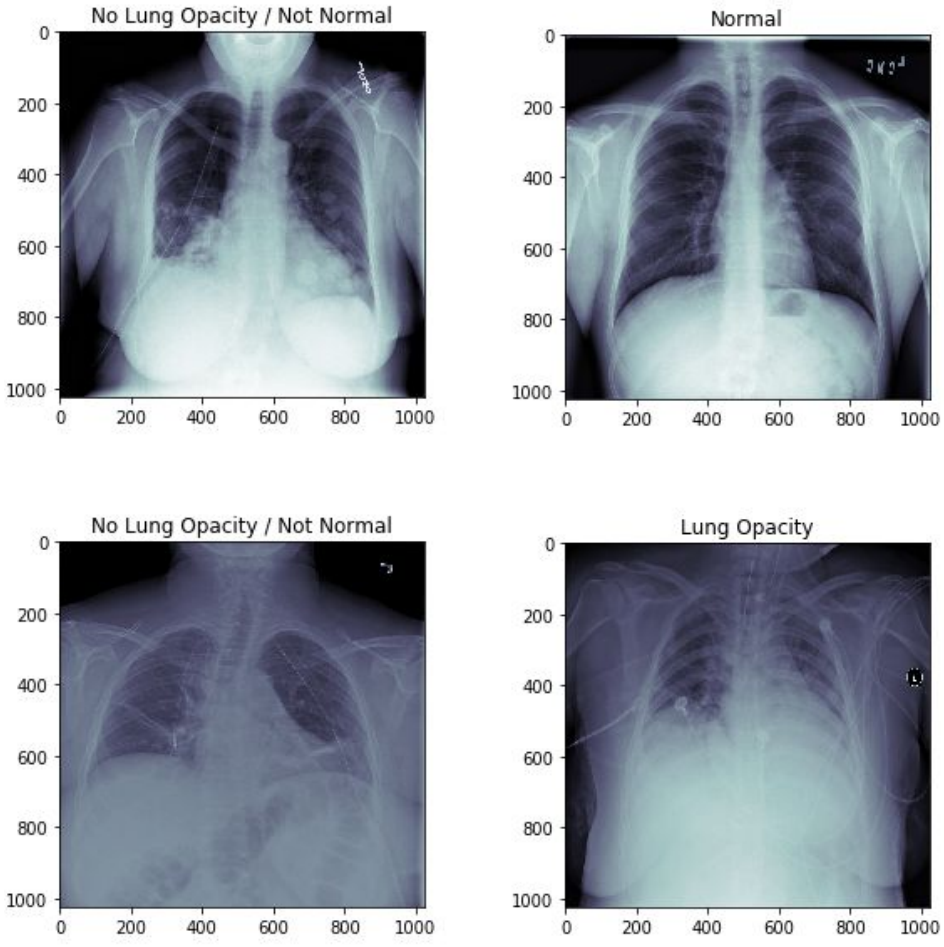
Justification

In the initial analysis, the F-1 score was around 55%. After applying refinements step by step, F-1 score was found to reach 70%. For normal lung cases the F-1 score has touched a high of 85%. For Lung Opacity cases, which is our main objective, we were able to reach 70% F-1 score which is a 15 % improvement from the initial analysis.

V. Conclusion

Free-Form Visualization

Below is a sample of Chest X-ray images visualized. The visuals outrightly denote that Normal cases can be clearly differentiated and knocked out in most situations. Also, classifying the Lung Opacity could be successful in 70% of the cases. The challenging part here (for both our model and the radiologists) is to categorize the inbetween level which is ‘Not Normal/ No Lung Opacity’.



Reflection

This project was a very good experience and this will be a stepping stone into deep learning/ AI field. The three most interesting aspects of the project were, firstly processing the image as per keras input requirement. Secondly, the process of tuning the hyperparameters and seeing how it affected the model; this task required the most patience too. Thirdly, the basic understanding of when to build what kind of deep learning layer and what would be the implication of that.

Improvement

The model is doing very well with the classification of Normal cases, good with Lung Opacity. But it still needs an improvement in Not Normal Cases which is in between level of Lung Opacity and Normal. The model is finding difficult to find the tradeoff point between Lung Opacity and Normal.

The X-Ray image consists of both female and male cases. There might be some analysis required to see if the sex of the patient affects the model's classification as the structure of the male and female differ in X-Ray Image. If it does then we need to pass the sex inforamtion to the model or do the classification separately based on sex.

Currently due to limitation of resources the maximum sample that can be passed into model is close to 12K images. The next big improvement would be to send the whole data available in the project. As the training data increases, the model is expected to see improvement. This can be easily achieved by a bigger machine or serverless applications like AWS Sagemaker.

Ref:

https://www.thoracic.org/patients/patient-resources/resources/top-pneumonia-facts.pdf

https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data

https://keras.io/applications/

https://arxiv.org/pdf/1409.1556.pdf