**Motivation**

With the Covid-19 pandemic, hospitals are overwhelmed with patients and imaging data has become plentiful. One complication of Covid-19 is getting pneumonia. This project focuses on creating image classification models to detect pneumonia.

**Data**

This data is from Kaggle: CoronaHack - Chest X-Ray Dataset, which sourced its data from GitHub: Covid Chest Xray Dataset

It contained 5935 images (5910 unique images, 22 duplicates) and a spreadsheet that mapped each image to 4 labels. One of those labels split the data into a training dataset vs a test dataset. The other 3 labels classified the data:

- Pneumonia vs Normal
- Covid-19 vs Other vs NULL
- Bacteria vs Other vs NULL

The 2nd and 3rd labels are intended to describe the cause of the pneumonia.

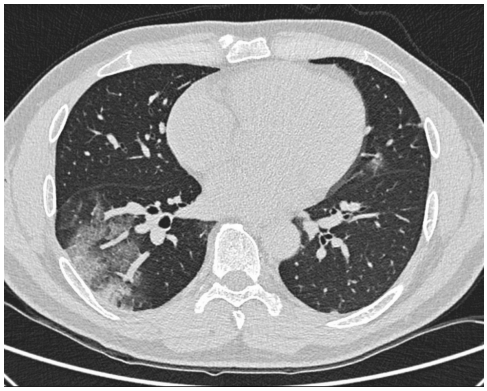
Here's details of the unique label groups:

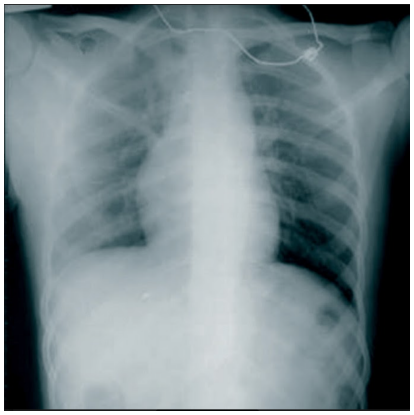
| **Label 1** | **Label 2** | **Label 3** | **Image Count** |
|---|---|---|---|
| Normal | NULL | NULL | 1576 |
| Pneumonia | Stress-Smoking | ARDS | 2 |
| Pneumonia | Virus | NULL | 1493 |
| Pneumonia | Virus | COVID-19 | 58 |
| Pneumonia | Virus | SARS | 4 |
| Pneumonia | Bacteria | NULL | 2772 |
| Pneumonia | Bacteria | Streptococcus | 5 |

Given the data is pre-labeled, there's a few constraints:

- The chest x-rays don't all show the same view. The photos below show:
    - Different angles: front view, top view, side view
    - Different colors: black/white, blue tone
    - Different zoom: variable
    - Different file shapes – this will be tackled in preparing the data for modeling.
- There's class imbalance
    - Label 1: 73% pneumonia vs 27% normal
    - Label 2: 1% Covid-19 vs 99% null vs 0.1% other

    - Label 3: 47% bacteria vs 27% null vs 26% other
- Data already split into train (89%) and test (11%)

- 





**Import Data**

This project is in Google Colab. To import the data:

1. Store the dataset in Google Drive.
2. Mount Google Drive to the Colab project.
3. Import images with pathlib.
4. Import labels with pandas.
5. Link images to labels with ImageDataGenerator.

**Data Cleaning**

First, I eliminated low count images from the dataset. These were for the image groups that had less than 10 images, i.e. images with Label 3 = {ARDS, SARS, Streptococcus}.

Second, I relabeled NULL to either Normal or Other. For Label 2, all NULL became Normal. For Label 3, Normal carried over, and everything else was some “Other” virus or bacteria that was not Covid-19. In the end, the dataset had 5899 non-null data points. At this stage of the project, re-labeling Labels 2 and 3 was not valuable because this project solely focuses on Label 1.

Third, I created a validation dataset. With train_test_split, I reserved 10% of the training data to be used as validation data. As a result, the train data had 4747 images, the validation data had 528 images, and the test data had 624 images.

**Exploratory Data Analysis**

Here's the breakdown of Label 1 for train vs test data:

| | **Normal** | **Pneumonia** | **TOTAL** |
|---|---|---|---|
| **Train** | 1196<br>*25%* | 3551<br>*75%* | 4747<br>*100%* |
| **Test** | 234<br>*38%* | 390<br>*62%* | 624<br>*100%* |

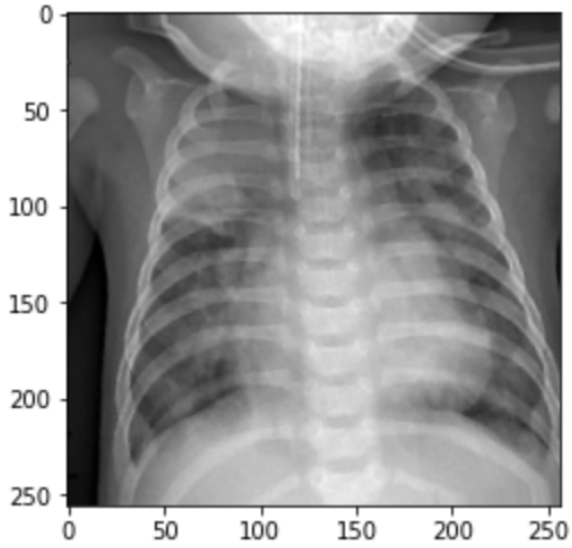
**Prepare Data for Modeling**

Other ways of preparing the data:

- Ensuring all images were the same size
    - Input shape = (32, 256, 256, 3)
- Setting batch size = 32

Here's a sample of one of the images from the notebook after resized to 256x256.

```
<matplotlib.image.AxesImage at 0x7fd534e1b150>
```




**Model Fundamentals**

I made 6 models using TensorFlow.  Each image classification models are composed of:

- Convolution layers = Conv2D + MaxPool
- Flatten layer
- Dense layer to 2 units because there's 2 classifications

At the end, each model is compiled:

- Optimizer = adam
- Loss = categorical cross entropy
- Metrics = accuracy

We set the number of epochs. With each epoch, the model is minimizing loss and maximizing accuracy.

Each model varies by:

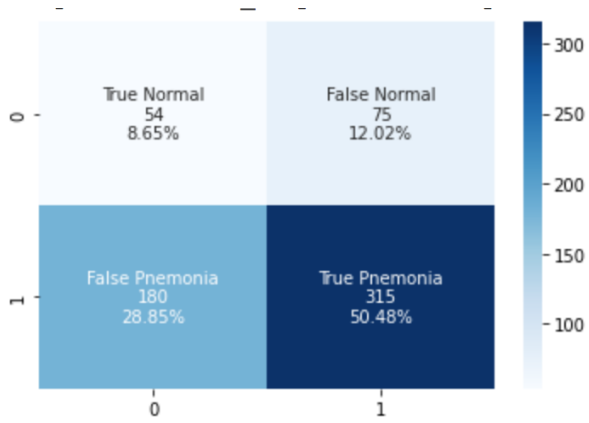
- Number of convolution layers
- Number of epochs
- Filter sequences

The expectation is that an increase in layers or epochs will improve the model's performance. There's no expectation for differences between filter sequences. Only experimentation can help drive results.

After testing each model, the model is judged by accuracy and recall, which is calculated based on the results in the confusion matrix. Accuracy helps compare test and train's performances. Since the test data isn't symmetric, recall might be a good metric for the test data because False Normal isn't acceptable, but False Pneumonia can be tolerated.

- Accuracy = True / All data
- Recall = True_Pneumonia / (True_Pneumonia + False Normal)

Here's the confusion matrix of Model 1:




**Model Comparison**

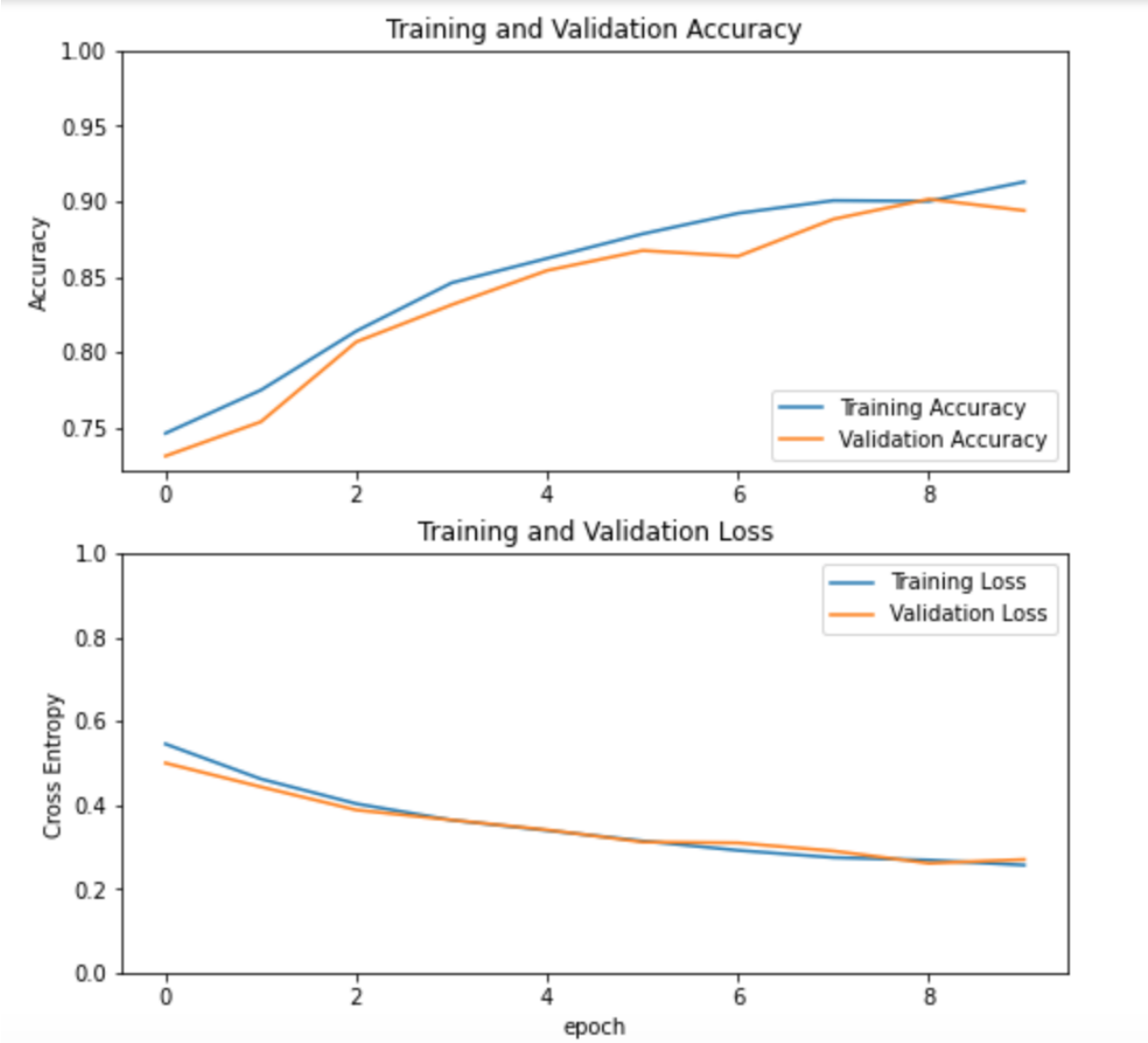
To clarify, Model 5.2 is the same as Model 5.1, but with training on an additional 20 epochs. Model 5.1 has two results because I collected all of the results from all of the models, but missed collecting results after 5.2. I re-ran Model 5.1 and 5.2 again in isolation. The results are slightly different between the first and second iteration of Model 5.1 probably due to randomness.

| **Model #** | **Number of Convolution Layers** | **Number of Epoch** | **Filters** | **Training Accuracy** | **Training Loss** | **Confusion Matrix** | **Test Accuracy** | **Test Recall** |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 10 | 16 | 0.9031 | 1.1846 | [[54,75] [180, 315]] | 59.1% | 81% |
| 2 | 2 | 10 | [32,16] | 0.8599 | 0.3508 | [[ 50 103] [184 287]] | 54.0% | 74% |
| 3 | 4 | 10 | [32,64,32,16] | 0.9248 | 0.1962 | [[ 70 132] [164 258]] | 52.6% | 66% |
| 4 | 4 | 10 | [128, 64, 32, 16] | 0.9265 | 0.1908 | [[ 65 122] [169 268]] | 53.4% | 69% |
| 5.1 | 3 | 20 | [64, 32, 16] | 0.9136 | 0.2157 | [[ 51 82] [183 308]]<br><br>[[85, 141] [149,249]] | 57.5%<br><br>53.5% | 79%<br><br>63.8% |
| 5.2 | 3 | +20 | [64, 32, 16] | 0.9328 | 0.1779 | [[73, 129][161, 261]] | 53.5% | 66.9% |
| 6 | 6 | 10 | [32, 64, 32, 64, 32, 16] | 0.9275 | 0.1897 | [[ 68 119] [166 271]] | 54.3% | 69% |
| 7 - Transfer Learning | MobileNetV2 | 10 | n/a | 0.9130 | 0.2565 | [[ 10 26] [224 364]] | 59.9% | 93% |

**Transfer Learning**

In addition to the 6 models, I used MobileNetV2 to run a Transfer Learning Model. By taking advantage of an already trained model that is much more sophisticated than any of the other 6 initial models, we expect better performance out of it.

Here is how accuracy and loss changed with more epochs:




**Best Model**

The transfer learning model was the best. The next best model was Model 1, which surprisingly was the simplest model. When comparing the two, the transfer learning model was slightly better in accuracy (+0.8%), and a lot better in recall (+12%).

When increasing layers or epochs, the model's accuracy did improve for the training data. That trend wasn't as true for the test data. Comparing 5.2 and 5.1, more epochs increased recall. Comparing Model 4 and 3, more filters increased accuracy and recall. However, everything else performed worse. This might have happened because the models overfitted the training data.

**Future Work**

If I were to improve upon this project, I would:

- Iteratively update models based on the test data's performance
- Try another Transfer Learning model
- Test on a symmetric dataset
- Continue creating models for Labels 2 and 3