Queen's

Master of Management in Artificial Intelligence

MMAI-894-2020
Deep Learning

Ofer Shai

Group Project
"Detecting Pneumonia from Chest X-Rays by Image-Based Deep Learning"
Mar 2, 2020

Team Adelaide:
Amit Kumar
Francis Bello
Tian Qu
Gunpreet Singh
Keerthi Sukhavasi
Mamta Gupta
Alexander Bakus

Table of Contents

Contents

1. Introduction

According to the World Health Organization, pneumonia is the single largest infectious cause of death in children worldwide. This disease is responsible for 15% of all deaths of children under the age of five, killing over 800,000 in 2017 or around 2,200 every day, mostly in South Asia and West and Central Africa. Pneumonia particularly affects children whose immune system may be weakened by malnutrition or undernourishment, especially in infants who are not exclusively breastfed.

Even though the number of children dying from pneumonia has decreased substantially over the last three decades, from 2 million deaths in 1990 to one-thirds by 2017, this disease still remains prevalent. Today, there are 1,400 cases of pneumonia per 100,000 children, or 1 per 71 children[1] making it a cause for concern globally.

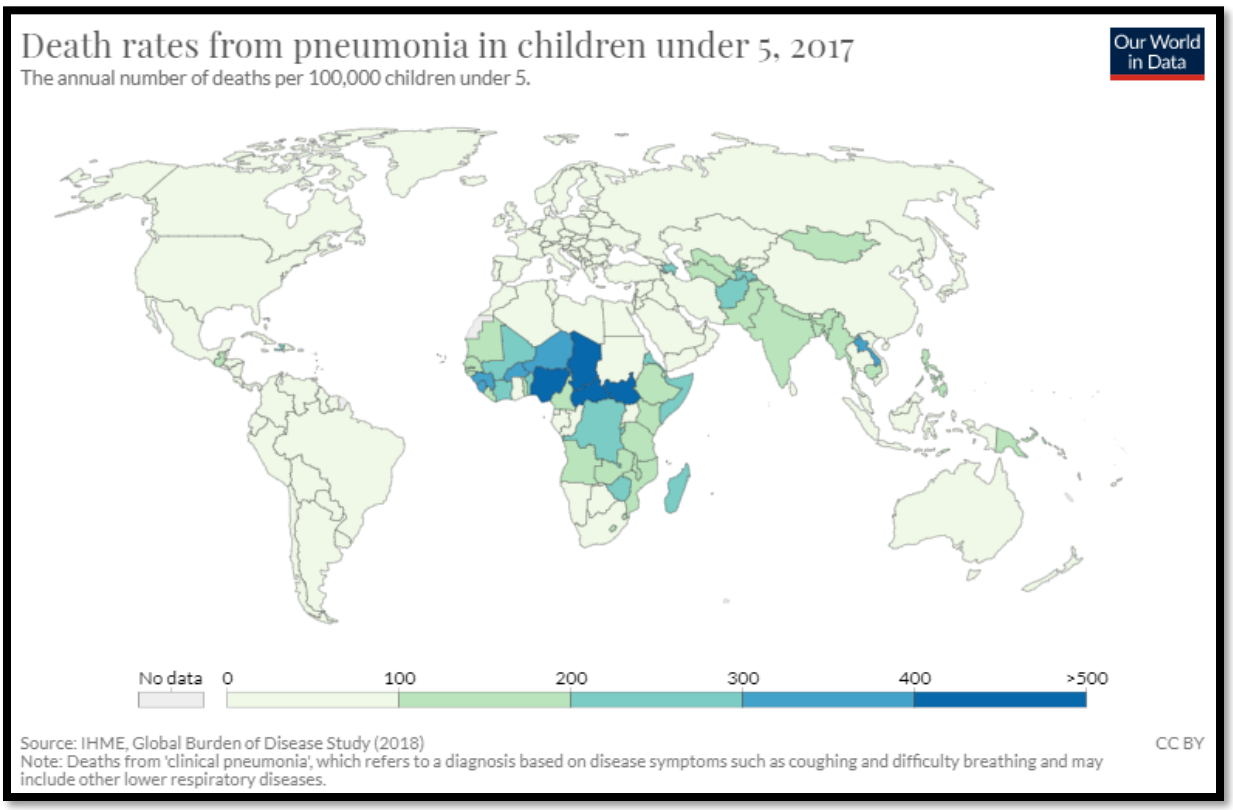


Figure 1.1

As the map shows, children are most likely to die from pneumonia across Sub-Saharan Africa and South Asia. Just 5 countries — India, Nigeria, Pakistan, the Democratic Republic of Congo, and Ethiopia – accounted for more than half of all deaths from childhood pneumonia in 2017.[2]

[1] World Health Organization. Retrieved from https://www.who.int/news-room/fact-sheets/detail/pneumonia

[2] Bernadeta Dadonaite and Max Roser (2020) - "Pneumonia". Published online at OurWorldInData.org. Retrieved from: https://ourworldindata.org/pneumonia

1.1. Objectives and Approach

Pneumonia is a bacterial, viral or fungal infection of one of both sides of the lungs that causes the air sacs, or alveoli, of the lungs to fill up with fluid or pus.[3] At present, detection of this disease is done by:[4]

- Blood test. Blood test are used to confirm an infection and to try to identify the type of organism causing the infection.
- Chest X-ray. This helps the doctor to diagnose the disease and determine the extent and location of the infection.
- Pulse Oximetry. This measures the oxygen level in the blood. Pneumonia can prevent the lungs from moving enough oxygen into the bloodstream.
- Sputum test. A sample of fluid from the lungs (sputum) is taken after a dep cough and analyzed to help pinpoint the cause of the infection.

This report focuses on the problem of differentiating a healthy lung from pneumonia-positive (bacteria and virus infected) via X-ray images. Our initial objective of this report was to come up with a CNN model to classify the X-ray images of the lung between normal vs pneumonia and to aid health professionals in quick detection of the disease. However, according to the new study, the researchers found that patients diagnosed with bacterial pneumonia had a higher risk of heart attack, stroke or death, compared with patients diagnosed with viral pneumonia. Therefore, we have extended our model from two classifiers to three classifiers: 1) normal vs 2) pneumonia-bacteria and 3) pneumonia-viral.

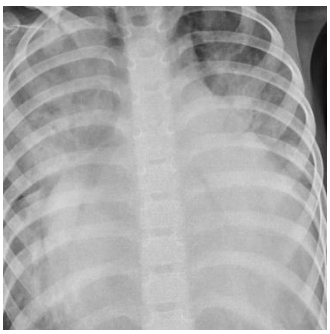
This report describes the approach we took in designing and developing the deep learning models to predict if a person has normal or has pneumonia caused by bacteria or virus using a CNN model. In recent times, CNN-motivated deep learning algorithms have become the standard choice for medical image classifications. Our work focuses on the dataset entitled “Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images”, however, we focused our work on chest x-rays and left the eye disease dataset[5]. The researchers made public almost 6,000 image files, separated into pneumonia bacteria type cases (2,700+), viral type (1,400+) and normal cases (1,500+). Since the distribution of images was not balanced, we implemented the up-sampling technique on the normal cases in order to avoid sample bias.

[3] Pneumonia. Retrieved from https://www.nhlbi.nih.gov/health-topics/pneumonia

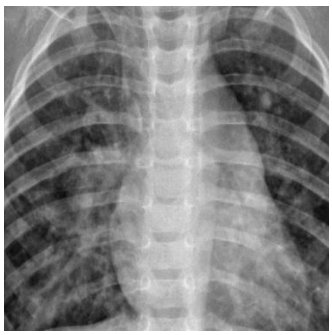
[4] Pneumonia. Retrieved from https://www.mayoclinic.org/diseases-conditions/pneumonia/diagnosis-treatment/drc-20354210

[5] Kermany, Zhang, Goldbaum (2018). “Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images”. Retrieved from https://data.mendeley.com/datasets/rscbjbr9sj/3

Normal



Bacterial Pneumonia



Viral Pneumonia

Figure 1.2 Sample Images

Our attempt to build a CNN classifier to train on these images and be able to differentiate a healthy lung from pneumonia-positive (bacteria and virus infected) image may open doors or opportunities to other practical application of Artificial Intelligence in the medical industry in order to help in the diagnosis, prevention and cure of pneumonia as well as other diseases both in children and adults in a timely manner.

1.2. Report Organization

The report is divided primarily into 6 sections, section [2] of the report describes the available dataset and the steps taken during image preprocessing to prepare the images for training and testing by the CNN model. The Experimentation section [3] describes the different approaches taken and various combinations of hyperparameters were tried to come up with the best 10 models. How these models can be used in production and what are the challenges and limitations in using them in production are described in the Extension to Production section [4]. The Design[5] and Implementation [6] sections describe the final model used for training and evaluating the model on test data. Finally, the Results section [7] presents the metrics evaluation and future scope.

2. Data Preparation

2.1. Dataset

The researchers of the project made available JPEG images of the different conditions of the lungs: normal, pneumonia caused by bacteria, and pneumonia caused by virus. They have made use of the convention when they named the files with:

- **Bacterial pneumonia**: “personXXXX_bacteria_YYYY.JPEG”
- **Normal lungs**: “IM-XXXX-YYYY.JPEG” or “NORMAL2-IM-XXXX-YYYY.JPEG” or “NORMAL2-IM-XXXX-YYYY-YYYY.JPEG”
- **Viral pneumonia**: “personXXXX_virus_YYYY.JPEG”

where XXXX or YYYY are some sequential codes.

The researchers pointed out that all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were graded by two expert physicians.[6]

The X-ray images are primarily of bacterial and viral pathogens, as well as normal lung radiographs to obtain a baseline ground truth, of children under the age of 5 particularly in low resource environments where occurrence of pneumonia in children has the highest incidence and mortality rates. The researchers utilized these images to detect pneumonia from pediatric chest X-rays and furthermore distinguished between viral and bacterial infections to facilitate and apply immediate and correct intervention. Bacterial pneumonia requires immediate medical antibiotic treatment while viral pneumonia is treated with supportive care. However, the corpus of the publicly available pediatric chest X-ray images represents a relatively small repository of data.

It is important to note that the dataset is not balanced considering the bacterial pneumonia class has almost twice the number of images in the dataset compared to either the normal or viral disease as shown below:

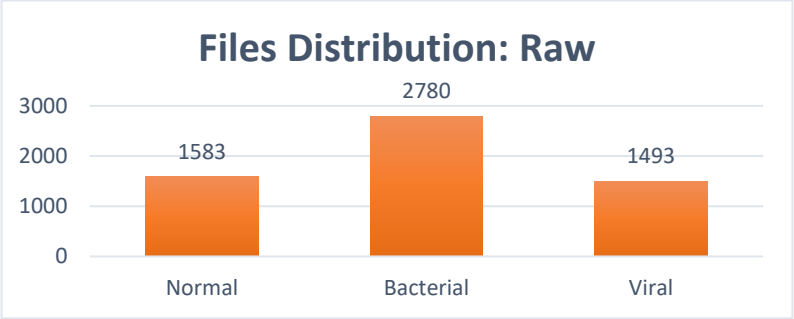


Figure 2.1 Original number of images

As observed, for every single image of a bacterial pneumonia there are 1.76 images of normal lung condition and 1.86 images of viral pneumonia. Given the dataset, making a prediction on these would be impartial and biased due to class imbalance towards the majority class resulting in a high probability of predicting a bacterial pneumonia most of the time.

Furthermore, the sizes of the images vary significantly from the smallest file to the largest file as shown by this chart:

[6] Kermany, Goldbaum. Cai, et al. (2018). "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning,"

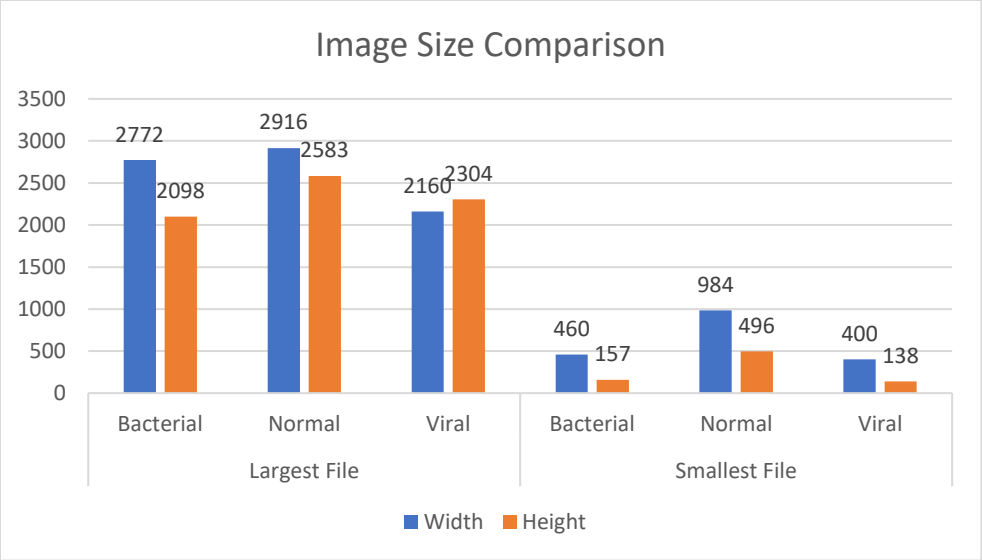
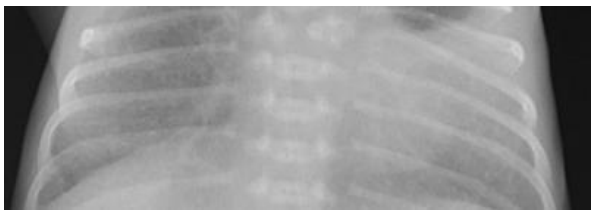
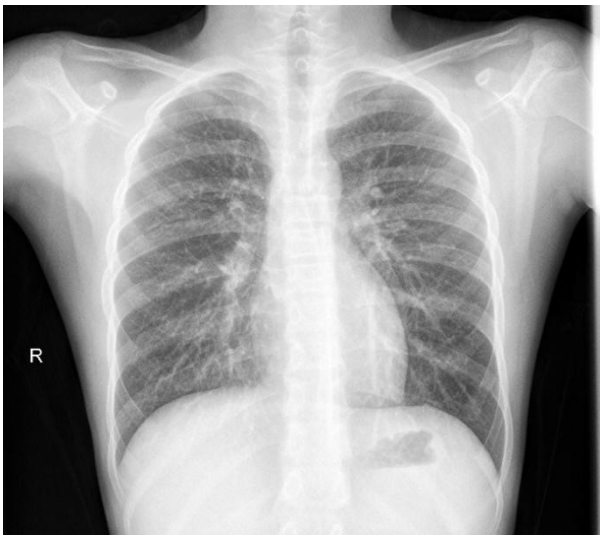


Figure 2.2 Image Size Comparison



Smallest file (400x138 pixels)



Largest file (2916x2583 pixels)

Figure 2.3 Smallest and Largest Files View

2.2. Image Preprocessing

From the image dataset, it is evident that there are a lot of inconsistencies in terms of size, shape and noise. To remove these inconsistencies, we followed the following steps to prepare and clean the images for our model to be trained and tested on.

2.2.1. Center crop the Images

Since the provided X-ray radiographs in the dataset had lungs in the center of the images, we would not lose any pneumonia information found in the center. Therefore, the first step in the preprocessing stage was to center crop the images primarily to get the center of the image and reduce the size. This further helped us in two major ways:

1. Smaller sized files allowed the model to be fine-tuned faster
2. Most of the "noise" or undesirable parts or foreign objects that are not part of the body were removed from the images

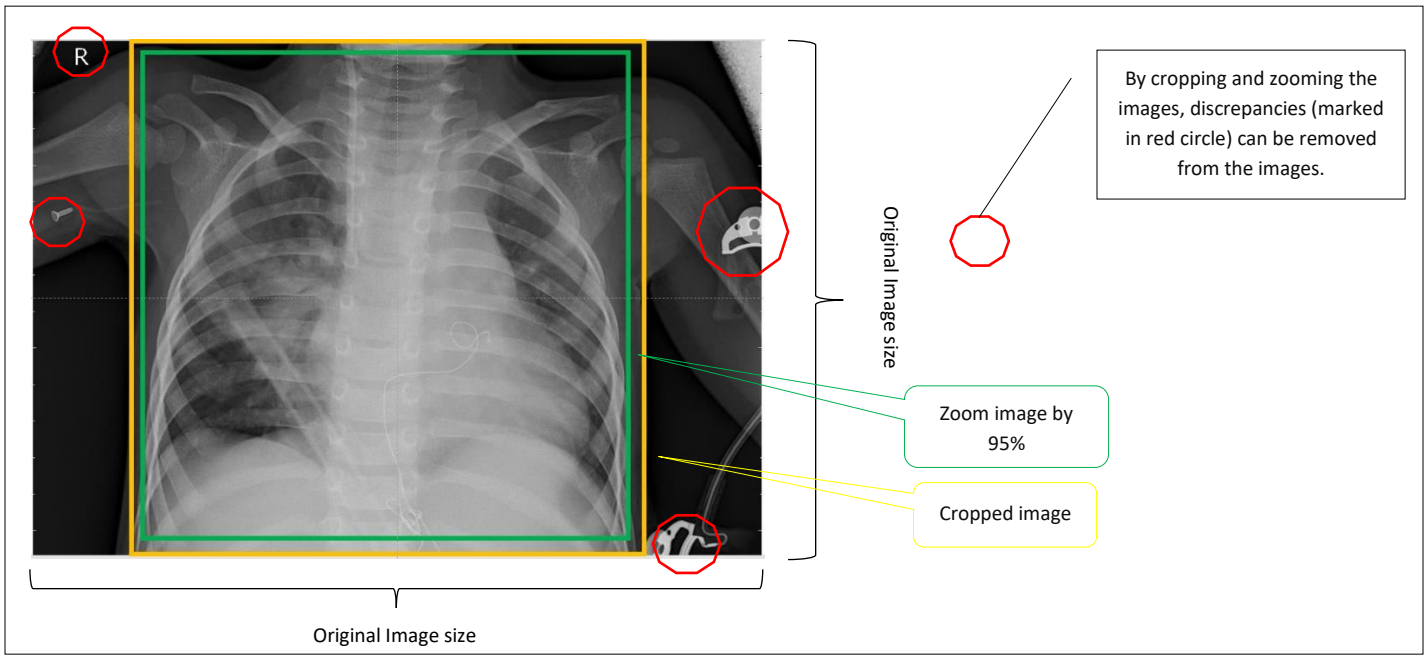


Figure 2.4 Visualization of center crop on the image

2.2.2. Reshape the Images

In any machine learning classification process, in order to minimize training bias, the tendency is to have equal or close enough ratios between each class, in this case, we have 3 classes (normal, virus, and bacteria) so ideally, we should have 1:1:1 ratio for all.

We noticed that all the images, totaling to 5,856, had varying resolutions. Since the CNN network takes one size input, we need to have all input images have the same and preferably higher resolutions.

Table 2.1 shows the minimum size of the image is 100x100 and as we increase our criteria to have higher resolution images of 960x960 the number of images available for model decreases significantly to 2,500. To minimize the tradeoff between the number of images available for fine

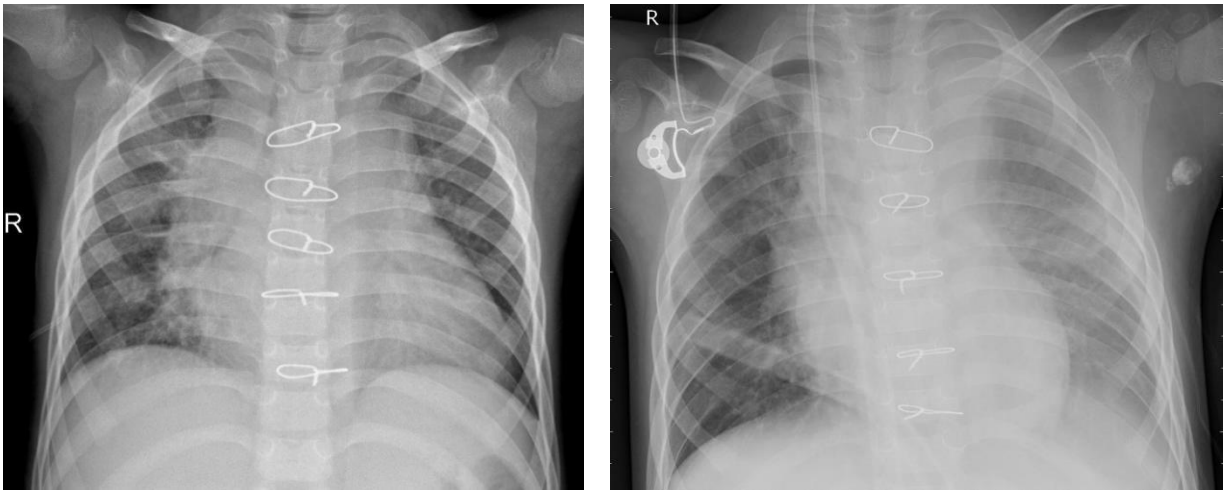
tuning the model and higher resolution, we have restricted our input image size to 300x300 pixels and preserved over 98% of the original samples.

Pixel size	> 100x100	> 300x300	> 480x480	> 512x512	> 640x640	> 720x720	> 850x850	> 960x960
Normal	1583	1583	1583	1582	1581	1576	1542	1476
Bacteria	2790	2735	2642	2532	2066	1593	916	588
Virus	1483	1462	1419	1397	1024	999	690	495
Total	5856	5780	5644	5511	4671	4168	3148	2559
Data Preservation Ratio	100.00%	98.70%	96.38%	94.11%	79.76%	71.17%	53.76%	43.70%
Ratio across all class	1:1.8:0.9	1:1.7:0.9	1:1.7:0.9	1:1.6:0.9	1:1.3:0.6	1:1:0.6	1:0.6:0.4	1:0.4:0.3

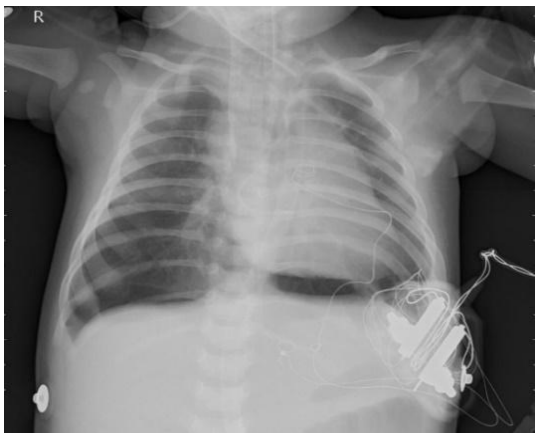
Table 2.1 Size of the image for each category (normal, bacterial and viral)

2.2.3. Manually removing the files

Even though center cropping removed noise from most of the images, there remained images that still contained “noise” or undesirable parts/foreign objects that were not part of the lungs but were captured when the X-rays were taken. These foreign objects include ECG sticker pads, breathing tubes (probably used in tracheostomy), intravenous clips or catheters, buttons, draining tubes from the lungs, sternal wires, coronary stent, gastrostomy tubes, labels/descriptions, as shown by some samples below:



Sternal wires / coronary stent




Gastrostomy tubes

Figure 2.5 Sample Images with Foreign Objects (noise)

To remove the effect of these images in the training of our model, a great effort was done to exclude these files from the dataset, thus reducing the file count even further, final count 3,253.

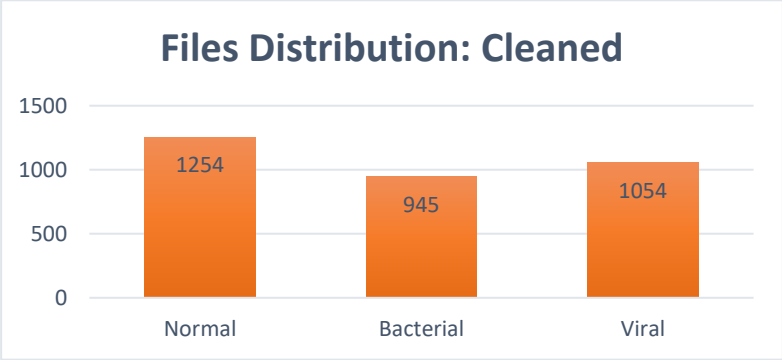


Figure 2.6 After preprocessing, the number of images available for training and testing the models

Once these preprocessing steps were completed, we proceeded with various experiment configurations of convolutional neural networks, the details of which will be discussed in the succeeding section.

3. Experimentation

The purpose of this section is to describe the team's approach and thought process to come up with a best model design by performing many trials such as testing different combinations of hyperparameters, varying the network depth, as well as up-sampling images by performing data augmentation.

3.1. Hardware Acceleration

Since we were testing model candidates with different hyperparameters, the team decided to utilize GPU to accelerate the training process which in this case was Nvidia GTX1080. We have followed as guidelines the following articles pertaining to hardware optimization:

- "Installing Tensorflow with CUDA, cuDNN and GPU support on Windows 10"[7]
- "The Best Way to Install TensorFlow with GPU Support on Windows 10 (Without Installing CUDA)"[8]

Package keras-gpu[9] was installed to enable the use of GPU in the training process.

```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  # Restrict TensorFlow to only allocate 5GB of memory on the first GPU
  try:
    tf.config.experimental.set_virtual_device_configuration(
        gpus[0],
        [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=5800)])
    logical_gpus = tf.config.experimental.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Virtual devices must be set before GPUs have been initialized
    print(e)
```

Code snippet 3.1 (Hardware Acceleration)

3.2. Train, Validation, Test Split

Common to many proper Machine Learning processes, the pre-processed images were split in a ratio of 70% for training, 20% for validation, and 10% for test evaluation.

The method used by the team to split the files in three folders (\normal, \bacteria, \virus) was ***split_folders.ratio***

3.3. Model Training

3.3.1. Tuning the Hyperparameters

When the team started building the classifier, the challenge was to decide which hyperparameters to start with. The team used the paper "An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare"[10] as a reference to pick different combinations of the hyperparameters to explore on.

---

[7] Kitson (2019). "Installing Tensorflow with CUDA, cuDNN and GPU support on Windows 10". Retrieved from https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781

[8] Kinghorn (2018). "The Best Way to Install TensorFlow with GPU Support on Windows 10 (Without Installing CUDA)". Retrived from https://www.pugetsystems.com/labs/hpc/The-Best-Way-to-Install-TensorFlow-with-GPU-Support-on-Windows-10-Without-Installing-CUDA-1187/

[9] Keras-GPU. (n.d.). "Deep Learning Library for Theano and TensorFlow." Retrieved from http://faroit.com/keras-docs/1.2.0/

[10] Okeke, Mangal, Uchenna, et al. (2019). "An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare". Retrieved from https://www.hindawi.com/journals/jhe/2019/4180949/

With all hyperparameters available for tuning (Table 3.1), a total of 648 possible combinations of the CNN setup could be created and evaluated.

Hyper parameters	Values Searched
Number of Convolution Layers	3, 4, 5
First Convolution Layer Size (subsequent layers are multiples of 2's of the first layer and their layer number-1)	16, 32
Number of Densely connected layers (actual number of neurons per dense layer is either 256 or 512)	0, 1, 2
Learning Rate	0.001, 0.01
Kernel Size	2, 3, 4
Dropout Rates	0%, 25%, 50%
Batch Size	32
Max Epochs	15
Activation Function	relu
Padding Method	Valid
Kernel Initializer	Random Uniform
Optimizer	Adam, SGD (Stochastic Gradient Descent)
Loss Function	Categorical Cross Entropy
Training Patients	2
Input Size	200x200x3

Table 3.1 Hyperparameters and values

3.3.2. ImageDataGenerator

To have more training images, the team implemented image augmentation with the ImageDataGenerator from tensorflow.keras.preprocessing.image with the following parameters:

Parameters	Values
Rescale	1./255
Shear Range	0.2
Rotation Range	40
Zoom Range	0.2
Width Shift Range	0.2
Height Shift Range	0.2
Horizontal Flip	True

Table 3.2 List of parameters/values to generate new images

.flow_from_directory

All of the training and validation images were augmented and generated on-demand with the method '.flow_from_directory'

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size, class_mode='categorical')
```

```
validation_generator = val_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size, class_mode='categorical')
```

Code snippet 3.2 (ImageDataGenerator and flow_from_directory)

3.3.3. Dynamically Generate Models

Each model was generated dynamically based on the current hyperparameter combination.

Relevant Code	Annotation

`model = Sequential()`
`model.add(Conv2D(layer_size,`
`kernel_size=(kernel_size, kernel_size),`
`activation='relu',`
`padding='valid',`
`kernel_initializer='random_uniform',`
`input_shape=input_shape))`
`model.add(MaxPooling2D(pool_size=(2, 2)))`
`model.add(Dropout(dropout_rate))`
	Input Layer

`for l in range(conv_layer - 1):`
`model.add(Conv2D(layer_size*(2**1),`
`kernel_size=(kernel_size,kernel_size),`
`activation='relu',`
`padding='valid',`
`kernel_initializer='random_uniform'))`
`model.add(MaxPooling2D(pool_size=(2, 2)))`
`model.add(Dropout(dropout_rate))`
	Hidden Convolution Layer 2,3,4,5

`model.add(Flatten())`	Flatten

`for _ in range(dense_layer):`
`model.add(Dense(layer_size*16))`
`model.add(Activation('relu'))`
`model.add(Dropout(dropout_rate))`
	Dense Layer (0, 1, 2)

`model.add(Dense(3, activation='softmax'))`	Output Layer

`model.compile(loss='categorical_crossentropy',`
`optimizer = optimizer,`
`lr = learn_rate,`
`metrics=['accuracy'])`
	Compile Model

Code snippet 3.3 (Code used to dynamically generate the model)

3.3.4. Training Progress Monitoring and Model Saving

As the searching and training sequence is expected to run for about a week, it was important for the team to be able to view the current progress and visualize the model performance through a monitoring platform.

The team chose TensorBoard as the tool to monitor the progress of each model's training and validation performance.

```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)
```

Code snippet 3.4 (Tensor Board Callback)

To minimize the chance of overfitting, EarlyStopping method was implemented to continually monitor the 'val_loss' with 'patience' set at 2 epochs.

```
es_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
verbose=1,patience=2)
```

Code snippet 3.5 (Earlystopping)

To save the model with its best epoch weight, ModelCheckpoint method was implemented and each model was given a name that described the hyperparameters set tagged with the current datetime to ensure uniqueness.

```
checkpoint_callback =  tf.keras.callbacks.ModelCheckpoint(filepath=(model_export +
mod  el_name+ model_time + '.h5'), monitor='val_accuracy',  save_best_only=True)
```

Code snippet 3.6 (Model Check Point)

3.3.5. Fitting the Model

With the use of training and validation image generation method, '.fit_generator' method was used as the method for fitting for each model.

```
model.fit_generator(train_generator,
                    steps_per_epoch=nb_train_samples // batch_size,
                    epochs=epochs,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    callbacks=[tensorboard_callback, es_callback, checkpoint_callback])
```

Code snippet 3.6 (.fit_generator)

3.4. Top Model Identification

After performing 648 experiments, we have chosen the top 10 models based on the highest test accuracy score as shown by the table below:

Name	Train Accuracy	Validation Accuracy	Test Accuracy
5_conv_32_nodes_1_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr	75.9%	75.2%	**77.2%**
4_conv_16_nodes_1_dense_2_kernelSz_adam_optimizer_0_01lr_0_0_dropr	73.8%	73.4%	74.8%
5_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_001lr_0_0_dropr	73.7%	73.3%	74.5%
4_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_01lr_0_0_dropr	73.5%	72.6%	74.2%
5_conv_32_nodes_0_dense_4_kernelSz_adam_optimizer_0_01lr_0_0_dropr	74.3%	73.7%	73.6%
4_conv_32_nodes_2_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr	71.0%	72.1%	70.9%
5_conv_32_nodes_0_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr	72.7%	73.0%	69.1%
5_conv_32_nodes_0_dense_2_kernelSz_adam_optimizer_0_001lr_0_0_dropr	70.5%	69.6%	68.5%
5_conv_32_nodes_1_dense_3_kernelSz_adam_optimizer_0_001lr_0_0_dropr	73.6%	68.8%	68.5%
5_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_01lr_0_0_dropr	66.0%	66.6%	61.9%

Table 3.3 - Top 10 models with hyperparameter details (Convolutional layer, nodes, dense layer, kernel size, optimizer, learning rate, dropout rate)

From the above 10 models, with different hyperparameters, we have taken the best model with highest test accuracy for our design and implementation purpose.

4. Design of the Best Model

Deep learning is a subset of machine learning that uses layered structure to process data and perform a given task. Data is fed through each layer where output from one is used as input to the other and continuous learning occurs throughout the network. Compared to conventional machine learning, deep learning models need fewer pre-processing steps as the networks take care of filtering and normalization tasks which otherwise usually require human and certain domain experts. In addition, the multi-layered structure of a neural network provides two major benefits: it works exceptionally well with vast quantities of unstructured data and needs less human intervention to optimize the model's performance as it learns on its own [11].

A type of deep learning model, Convolutional Neural Networks, is well suited for analyzing images, including medical X-rays which is what our data set comprises of. There have been many recorded cases of a CNN outperforming human clinicians, in terms of accuracy as well as time taken to analyze an image. For example, researchers developed a model capable of detecting neurological conditions 150 times faster than human radiologists and only took 1.2 seconds to process the image[12]. Given the benefits of Deep Learning and its proven usage in other medical applications, the team was able to justify developing a CNN to detect pneumonia from X-ray

[11] Bresnick (2018). "What Is Deep Learning and How Will It Change Healthcare?" Retrieved from https://healthitanalytics.com/features/what-is-deep-learning-and-how-will-it-change-healthcare

[12] Ibid.

images. The dataset we used for our CNN included 3 classes: normal, bacterial, and viral. We started with a total of ~6,000 images and following the pre-processing steps, our final working dataset comprised of 3,253 images, with a proportion of 38.5% normal x-rays, 29.1% bacterial pneumonia, and 32.4% viral pneumonia.

Based on all the experiments the team ran, as specified in the Experimentation section, the CNN architecture that performed the best, given the computational resources and time, is provided below.

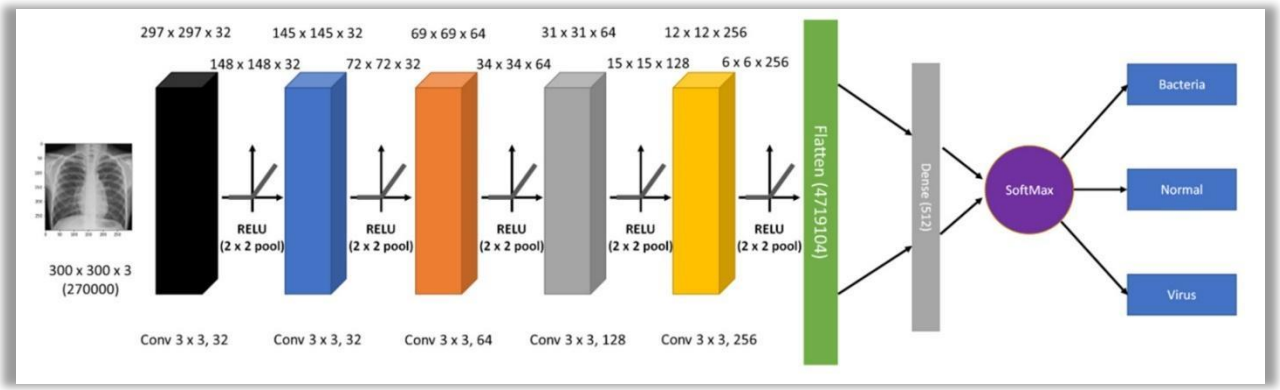


Figure 4.1 CNN Model Architecture

The best CNN model consists of 1 input layer, 4 hidden layers, 1 hidden dense layer, and 1 output layer with 3 classes (normal, viral, bacterial). The input for the CNN is an image of 300x300 pixels with 3 channels (or an array of 300 x 300 x 3 pixel values) since the image is encoded in RGB. On the first convolutional layer, we applied a 4x4 kernel with a stride of 1 and depth of 32 channels. This was followed by a 2x2 max pooling layer to reduce the size by half. By looping through different values in a grid search, we ended up with 4 more pairs of convolution and max pooling layers before flattening out the layer. The architecture has 1 hidden dense layer with 512 neurons and 1 output layer with 3 classes.

We have used our best model hyperparameters to see the effect of below parameters

- Input size
- Batch size
- Learning rate

4.1. Effect of Using Larger Input Size/Resolution

The original plan was to use larger image size (720x720) as input shape to the convolutional neural network. However due to time constraint, we did not have time to design or attempt this size. The team did manage to experiment on a list of smaller and different size of input images on two model architectures. The team hypothesized that higher resolution would improve model performance when trying to distinguish subtle differences such as two kinds of pneumonia cases (bacteria vs. virus) assuming the model design and compute power can support such input.

As the results below confirm our intuitions that larger input size does help a model's performance. However, the improvement seems to be limited to a few percentages but the cost of time in training models increases exponentially. This discovery is the primary reason that leads to the decision of using 200x200 input size for 'Grid Searching' and 300x300 for the retraining of top models.

Model A: (CNN layer: 32-64-128-256-512-flatten-256-256-3)

Input Resolution	Total Params	Epoch Time (sec)	Num Epoch	Test Accuracy	Total Training Time (min)
150	2159683	64	10	74.8%	11
200	3732547	103	15	73.6%	26
250	4912195	155	15	76.6%	39
300	8057923	220	22	78.1%	81

Table 4.1 – Model A Test accuracy with different input resolution

Model B (CNN layer: 32-64-128-256-512-flatten-512-3)

Input Resolution	Total Params	Epoch Time (sec)	Num Epoch	Test Accuracy	Total Training Time (min)
150	839683	67	9	72.4%	10
200	1888259	101	11	72.7%	19
250	2805763	153	15	73.0%	38
300	5427203	213	15	74.5%	53

Table 4.2 – Model B Test accuracy with different input resolution

4.2. Effect of Using Different Batch Sizes

The team was also interested in the effect of the batch size towards the model performance. It was observed that batch sizes larger than 16 have no effect on the performance.

Batch Size	Train Samples	Val Samples	Epochs	Steps	Epoch Time	Total Time	Test Accuracy
64	10000	2700	18	156	60	18	75.4%
32	10000	2700	13	312	64	14	74.5%
16	10000	2700	17	625	68	19	71.2%
12	10000	2700	7	832	69	8	68.2%
8	10000	2700	20	1250	65	22	71.2%

Table 4.3 –Test accuracy with different batch sizes

4.3. Effect of Learning Rate

The team has experimented with 5 different learning rates on the models: 0.0001, 0.001, 0.01, 0.1, and 1. The team found that the lower the learning rate the better the model performance was, however, it appeared that training the model with lower learning rate took longer duration.

Learning Rate	Train Samples	Val Samples	Epochs	Steps	Epoch Time	Total Time	Test Accuracy
0.0001	10000	2700	18	312	70	21	77.8%
0.001	10000	2700	14	312	70	16	74.8%
0.01	10000	2700	13	312	76	16	70.57
0.1	10000	2700	13	312	76	16	73.6%
1	10000	2700	13	312	64	14	72.1%

Table 4.4 –Test accuracy with different learning rates

4.4. Transfer Learning Comparison

To put our model's performance in more perspective, the team also ran transfer learning experiments with pre-trained models including VGG 16, VGG 19, Resnet 50 V2, Inception V3. The comparison table is provided below:

Model	Input Size	Train Samples	Val Samples	Epochs	Steps	Time per Epoch (seconds)	Total Time (minutes)	Test Accuracy
Team's CNN	150	10000	2700	18	312	**70**	21	**77.78%**
VGG16	150	10000	2700	10	312	115	19	75.98%
VGG19	150	10000	2700	8	312	130	17	67.87%
ResNet50V2	150	10000	2700	4	312	91	**6**	59.16%
InceptionV3	150	10000	2700	4	312	168	11	45.05%

Table 4.5 – Comparison between our Team model and transfer learning experiments

As observed from the comparison table above, our team's CNN model outperformed the other models in terms of overall accuracy. VGG 16 came the closest at 76% and Inception V3 had the worst performance at only 45%. In terms of training time per epoch, our CNN took only 70 seconds per epoch, however it took the most overall time for training at 21 minutes. VGG 16 took comparably the same amount of time for training at 19 minutes. ResNet 50 V2 performed the least amount of training time at 6 minutes, however the performance was quite low at only 59%.

Even though our model took the longest time to train, the difference in accuracy achieved was significant enough for us to conclude that our model is better in comparison with the popular models.

5. Implementation

This section explains the step-by-step process that we have undertaken from preprocessing and the different Python codes that we have developed in deeper detail.

5.1. Image Preprocessing

The following diagram showcases the steps we have performed for preprocessing.

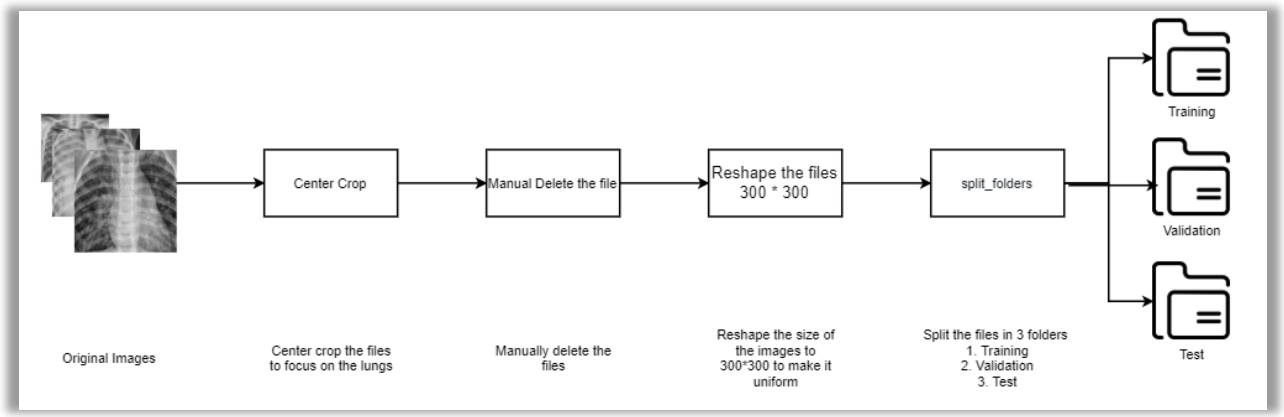


Figure 5.1 Image Preprocessing flow

Steps	Python code File	Functionality
Step 1	CropMaxSquare.py	Original Images stored in RawData\NORMAL and RawData\ PNEUMONIA
Step 2	CropMaxSquare.py	Create Directory center_crop_300 And Subdirectory center_crop_300/normal; center_crop_300/bacteria and center_crop_300/virus
Step 3	CropMaxSquare.py	Based on the filenames containing *normal*, *bacteria* and *virus* store the files in normal, bacteria and virus respective folder
Step 4	CropMaxSquare.py	Use function crop_center to center crop the image
Step 5	CropMaxSquare.py	Use function crop_max_square to zoom the image by 95%
Step 6	CropMaxSquare.py	Use method Image.resize to reshape the images to 300*300 pixels
Step 7	Manual Step	Manually delete the dirty files and stored the images in Center_crop_clean_300 maintaining the subdirectory structure of normal, bacteria and virus folder
Step 8	Create3Splits.py	Use method Split_folders to 1. move the files from input folder Center_crop_clean_300 to data_3C_from_center_crop_clean_300_to_300 and 2. Splits the files from each folder normal, bacteria and virus in 3 folders – 1) Training, 2) Validation and 3) Test Now the images are available in

		1. data_3C_from_center_crop_clean_300_to_300/train/normal 2. data_3C_from_center_crop_clean_300_to_300/train/bacteria 3. data_3C_from_center_crop_clean_300_to_300/train/normal similarly, for validation and test above folder structure is maintained

Table 5.1 Sequence of Steps Performed by Different Python Codes for Image Preprocessing

The following image shows the flow sequence performed during the hyperparameter tuning.

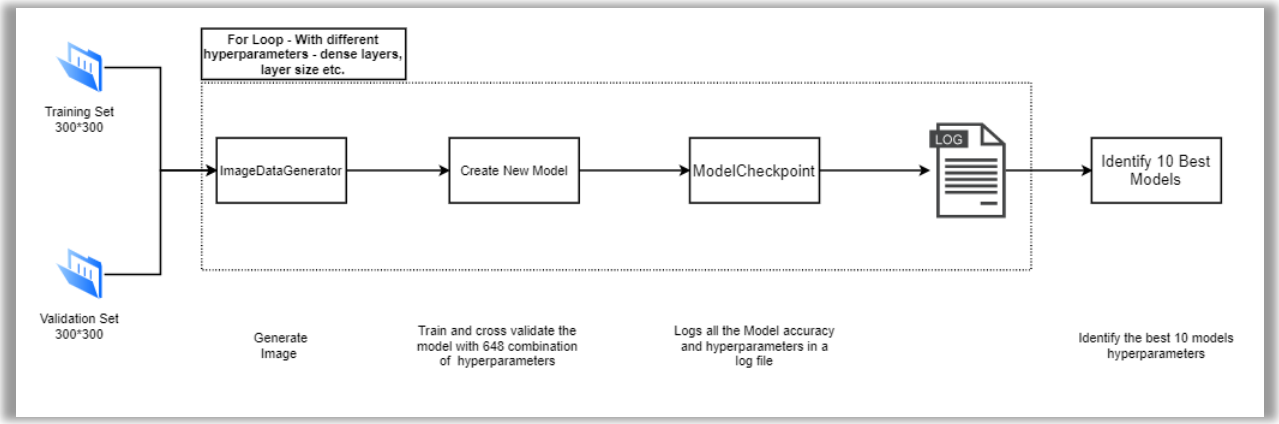


Figure 5.2 Identify the best 10 models

Steps	Python code File	Functionality
Step 1	model_withGridSearch.py	Set the Hardware acceleration settings
Step 2	model_withGridSearch.py	Tune the model with a combination of 648 hyperparameters in for loop
Step 3	model_withGridSearch.py	For each hyperparameter in Step 2 create folder with name as {conv_layer}_conv_{layer_size}_nodes_{dense_layer}_dense_{kernel_size} _kernelSz_{optimizer}_optimizer_{learn_rate}lr_{dropout_rate}_dropr_{model _time}_T Example: 5_conv_32_nodes_1_dense_4_kernelSz_adam_optimizer_0_001lr_0_0 _dropr to store the model, accuracy details
Step 4	model_withGridSearch.py	Create CNN sequential Model based on the hyperparameters > flatten > dense
Step 5	model_withGridSearch.py	Store the tensorboard_callback details in the log file
Step 6	model_withGridSearch.py	Generate 10,000 training and 2700 validation images with the method ImageDataGenerator
Step 7	model_withGridSearch.py	Fit the model with method fit_generator

Table 5.1 Sequence of Steps Using Python for Grid Search

For the 648 combination of hyperparameters, the team sorted the best model with highest accuracy and tested this model to arrive produce the results.

Steps	Python code File	Functionality
Step 1	evaluateModels.py	Image Data Generator – Use the existing test images and regenerate new images by rescale=1. / 255

Step 2	evaluateModels.py	Evaluate the Test set on the stored top 10 models
Step 3	evaluateModels.py	Report the accuracy

Table 5.3 Sequence of Steps Using Python for Model Evaluation

The following image describes running the best model selected out of the 648 combinations of hyperparameters and evaluating the result.

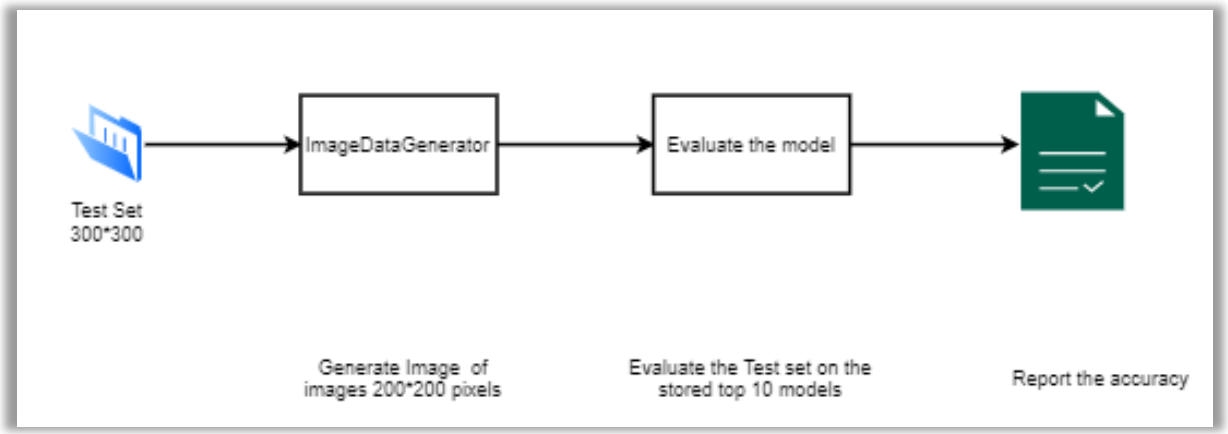


Figure 5.3 Evaluating the Best performing model

6. Extension to Production

Knowing the fact that pneumonia accounts for 15% of deaths in children under 5 years old, being able to detect infected patients through Deep Learning poses a lot of opportunities for the healthcare industry. However, we anticipate three major challenges in implementing the model our team built in a healthcare organization.

6.1. Generalization to new population

One of the major issues with most AI models is their generalizability, meaning how well the model can perform on new data population after it has been trained on a different set of data[13]. In a study from 2018 that looked at variable generalization of performance of a CNN to detect pneumonia in chest radiographs, the researchers trained the model on more than 150,000 chest X-rays from three hospital systems. The model performed extremely well when it was trained and tested on data from the same hospitals (internal data) but not when tested on radiographs from other hospitals (external data). To put it in perspective, they were able to achieve an AUC of 0.931 on internal data and only 0.815 on external data. While both numbers seem impressive, this is too big a gap to ignore when it comes to clinical applications. This implies that models trained on X-rays from one hospital are not sure to work equally as well on data from other hospitals[14].

[13] Kelly, Karthikesalingam, Suleyman, et al. (2019) "Key challenges for delivering clinical impact with artificial intelligence". Retrieved from https://bmcmedicine.biomedcentral.com/articles/10.1186/s12916-019-1426-2

[14] Zech, Badgeley, Liu, et al. (2018). "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study". Retrieved from https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002683

Generalization as a challenge can occur due to many different reasons including different equipment, operating conditions, Electronic Health Record systems as well as clinical policies. In our case, the CNN is trained from dataset belonging to one organization and we expect the same generalization challenge if it were to be implemented in a different organization.

A solution to this challenge can be curating large heterogeneous datasets of X-ray images from multiple sources to train the model rather than limiting it to one or two hospitals. Collaboration across health centers is vital to solving this problem as training data set needs to reflect the population before it is deployed in real-world settings and used for predictions.

6.2. Unified Data

The process of detecting pneumonia as it currently stands is not the most efficient or easy, since it requires a radiograph review by highly trained specialists and confirmation through various other medical data including clinical history, vital signs, and lab exams[15]. This poses another challenge of incomplete data when it comes to deployment in a clinical setting. Healthcare professionals may want to include all these indicators along with the medical imagery when using a CNN to detect pneumonia and that is where the logistical difficulties lie. Medical data is stored in silos, meaning different imaging archival systems, pathology systems, EHR and insurance databases[16]. A way to solve the problem is adoption of unified data format, for example, FHIR or Fast Healthcare Interoperability Resources, which provides a better method for aggregation of data. It is important to note that this external factor is an industry-wide challenge and could be a while before unified data formats are implemented. However, the need for it should encourage faster adoptions.

6.3. Model Explainability

The question of Explainable AI is a hot topic right now, especially for industries where stakes are high such as healthcare. Artificial Intelligence has made leaps and bounds in terms of accurate results. For example, some CNNs have surpassed the accuracy of diagnosticians when identifying features in imaging studies. Citing an example of a 2018 study where a CNN trained to analyze dermatology images was able to outperform the human clinicians with 10% more specificity when it came to identifying melanoma ***(B)***. However, the lack of interpretability when it comes to sophisticated AI algorithms is what hinders the adoption in the industry. This could be a big challenge for extending our team's CNN to production considering it is a deep learning model with very little insight into how it operates. Even though we understand the math behind its operation, it still acts as a "black box".

This challenge is a monumental one but in order to expedite implementation, we can position our CNN model as an aid to the clinical staff to flag the x-ray images when pneumonia is detected. Healthcare industry is commonly resource strained and clinicians are tasked with reading high

[15] Radiological Society of North America (2018). "RSNA Pneumonia Detection Challenge". Retrieved from https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview

[16] Kelly.

volumes of radiographs every day. By using our model, we're not eliminating their involvement but rather supplementing their work by detecting pneumonia and further passing it on for secondary examination. Our team's CNN can help speed up the process and free up the staff's time to focus on other important tasks.

6.4. Scalability

In addition to our model implementation, it is equally as important to consider the scalability. One of the biggest challenges the team has identified is the quality of the data, or the chest X-ray images. We started off with 6,000 images which contained high inconsistencies in regard to the size, shape and noisy data. These can be attributed to many factors such as the equipment, the patients, as well as the clinical process followed.

A lot of the work went into data preparation and pre-processing such as manually removing noisy images that contained foreign (such as ECG sticker pads, tubes etc.), center cropping the images to extract just the lungs portion, and reshaping the images to 300x300 pixels to obtain consistent size. This process was manageable considering we were working with only 6000 images. However, it's not a sustainable approach when scaling at a much larger level in the real-world.

Our team recommends a much more innovative and extensive approach for scaling this model which would be to create a Neural Network for each of the specific tasks. The first Neural Network should be trained and tasked for object detection, which in this case is to identify lungs from an X-ray image. The output from the Network should act as an input to the next CNN which would be to detect if the lung is infected or not. If positive with a certain level of confidence, that should then be fed to another Neural Network to detect if Pneumonia is viral or bacterial. This is more time and compute extensive, however it is much more robust and scalable for Pneumonia Detection from X-ray images.

7. Results

The highlights of this project are:

- Develop a deep neural network utilizing convolutional architecture
- Classify images for normal, bacterial and viral pneumonia with a reasonable accuracy
- Potential for use in biomedical radiography and imaging

7.1. Results Summary

We, Team Adelaide, developed a 5-layer convolutional neural network to classify 3 lung conditions, namely: normal, bacterial pneumonia, and viral pneumonia, where on each layer were 32, 64, 128, 256, 512 neurons, respectively. As presented in the Experimentation section of our report, our best model has achieved a weighted accuracy score of 77% after preprocessing the available data and feeding to our classifier.

The resulting test data confusion matrix:

		Predicted			
		Bacteria	Normal	Virus	Total
Actual	Bacteria	75	7	16	98
	Normal	1	126	2	129
	Virus	38	9	59	106
	Total	114	142	77	333

Table 7.1 Confusion Matrix

The following classification report is calculated:

	Precision	Recall	F1-Score
Bacteria	0.66	0.77	0.71
Normal	0.89	0.98	0.93
Virus	0.77	0.56	0.64
Accuracy			0.78
Macro Avg	0.77	0.77	0.76
Weighted Avg	0.78	0.78	0.77

Table 7.2 Classification Report

As shown on the table above, our classifier achieved an average accuracy score of 77%. With the quantity and quality of data that we have, we consider this score good and reasonable.

The normal case achieved the highest score amongst the 3 groups for precision, recall, f1-score, and support. This is attributed to the fact that the normal case images could be considered the cleanest batch and unique such that the absence of pneumonia markings on an x-ray radiograph would make the image a strong candidate of non-pneumonia case right away.

On the other hand, the viral and bacterial pneumonia cases faired lower precision scores versus the normal case. A close (near) precision score has been observed between viral and bacterial cases. Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations. In other words, precision is the ratio of predicting if an image is pneumonia-positive and if so, is it viral or bacterial infection from all the total positive predictions. High precision score relates to low false positive rate. Precision of 0.66 and 0.77 for bacterial and viral, respectively, could be considered reasonably good.

$$\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive})$$

Finally, the viral images scored the lowest in terms of recall, however, since it is above 0.5 it can be considered decent as well. Recall is the ratio of correctly predicted positive observations to all observations in the actual “true” class. In layman’s terms, recall says that of all the images that are positive of pneumonia (either viral or bacterial), how many did the model classify?

Recall = True Positive / (True Positive + False Negative)

The F1-score is the harmonic mean (weighted average) between precision and recall scores. F1 takes both the false positives and false negatives into account and is usually more useful than accuracy especially if the distribution of classes is imbalanced. Since all our three classes scored above 0.5 F1-scores, the model performed considerably good.

F1 = 2 * (Precision * Recall) / (Precision + Recall)

7.2. Recommendation

We feel that these scores could be considered good, albeit not stellar, for the data we had to train and test our model on. We are proposing the following recommendation that we believe could still make the classifier perform even better:

It goes without saying that more data would always be better. Having more data, in the form of images for our case, would allow our model "to see" more radiographs from each group. The model would benefit from training with more data and be able to generalize more and avoid overfitting.

Our attempt to up-sample the number of images to a higher number by using the ImageDataClassifier produced augmented images that were basically "clones" of the original images however twisted, cropped, or flipped they may be. Even though the quantity of trainable and testable images from the cleaned JPEG files were increased, no new patients/cases were introduced by doing so. However, this still added significant improvement to the performance of the model by the synthetic introduction of "new images". but the model could have the opportunity to further increase the accuracy and decrease the loss from more radiographs.

Training a neural network with a small dataset can cause the network to memorize all training examples, in turn leading to overfitting and poor performance on a holdout dataset. Small datasets may also represent a harder mapping problem for neural networks to learn, given the patchy or sparse sampling of points in the high-dimensional input space[17]

7.3. Future scope

To summarize our report, we consider our model as an exploratory or discovery tool towards designing and developing a convolutional neural network. Our classifier is not a deterministic tool to diagnose if a person has pneumonia or not, and therefore, we do not recommend production deployment at its current state yet.

It took us a minimum of 5 days only to explore different combinations of hyperparameters, network depth, batch sizes, and epoch counts to achieve a reasonable accuracy, not to mention the cleaning of the data that was done mostly manually. Even with these, not all hyperparameters were exhausted due to the limitation of computing power we were faced with and therefore settled

[17] Brownlee (2019). "Train Neural Network With Noise to Reduce Overfitting".

on the most number of hyperparameters that our machine can work on. This does not include the size of the input data. Bigger array size would further consume more GPU in addition to the ones that the CNN architecture is already using.

Many more tasks should be done before our classifier becomes a useful helping tool to medical professionals in determining pneumonia. In addition to the recommendations already mentioned, an attempt could be done to make the model be able to tell where the location of the disease is mostly clustered or appearing. Thus, a medical professional is needed to add additional labels on the pneumonia-positive images where the infection is mostly concentrated in. With more available data and performing more trainings to achieve significantly better results, only then when we could recommend the possible use of our model as an aid in the medical field. It is not our attempt or intention to replace human doctors or nurses in the diagnoses of this disease by this tool we have developed.

Furthermore, a good design is to make convolutional neural networks transfer learning-ready. As such, redesigning our classifier to different sub-classifiers could be beneficial. This is akin to multiple doctors treating a patient. Multiple opinions from many doctors, not only one, and therefore increases the accuracy of the presence of the disease.

In the same manner, we are proposing an ensemble of CNNs for detecting pneumonia to be designed in the future. By this design, more transparency is introduced and therefore can pinpoint with higher accuracy at what step is the diagnosis currently happening.

An initial design of the ensemble architecture of CNN's on a high level is shown below:

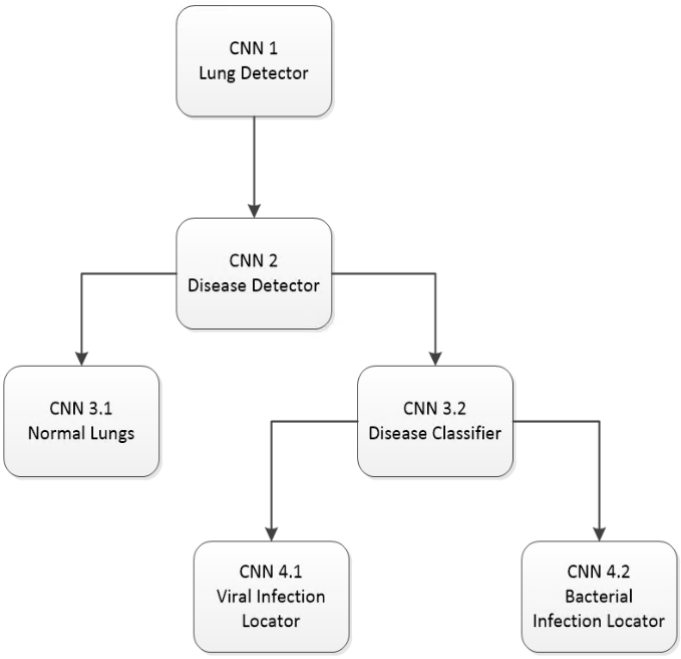


Figure 7.1 Initial design of the ensemble architecture of CNN's on a high level

This was our first attempt on developing a convolutional neural network using real public data. This allowed us to determine what worked best for this specific situation, what was flawed, how fast our executions were running, and how RAM (memory) was utilized, etc. Publicly available

neural networks could also be utilized using transfer learning, such as VGG Net, ResNet or GoogleLeNet. One of the challenges by using a readily available CNN would be determining how many last layers would be replaced in these networks to adapt them to predict pneumonia or not. On the other hand, an advantage is that those CNN's have been trained already.

We hope that our work would inspire future attempts and in turn, benefit people everywhere with quick diagnosis and treatment of not only pneumonia infections, but other diseases of the lungs and other parts of the body as well.

Acknowledgement

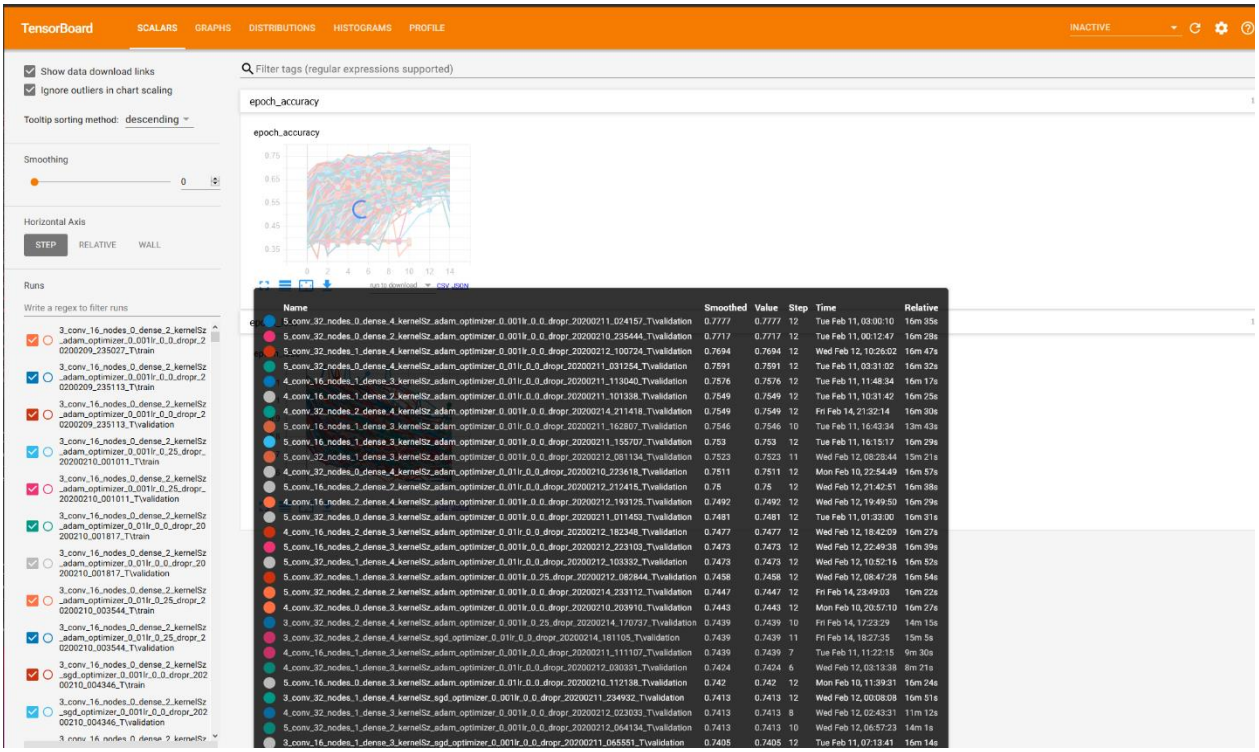
For this report, all Team Adelaide members contributed equally to the completion of this report. Starting from the identification of the dataset, business proposal, experimentation, designing of the top performing model, writing and reviewing the report each team member has put their effort.

Appendix

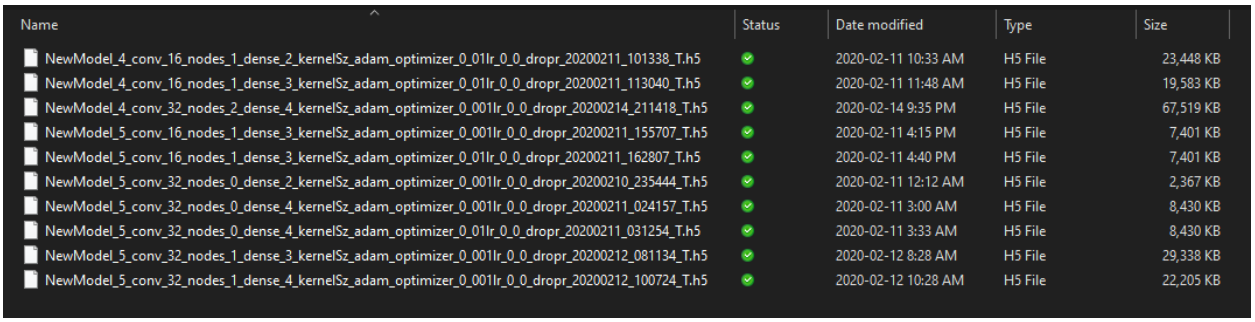
Top Models Re-Training

This section describes what all experiments the team has done on top 10 models to arrive on the best performing model.

After 5 days of training, top ten models with the highest validation accuracy were selected



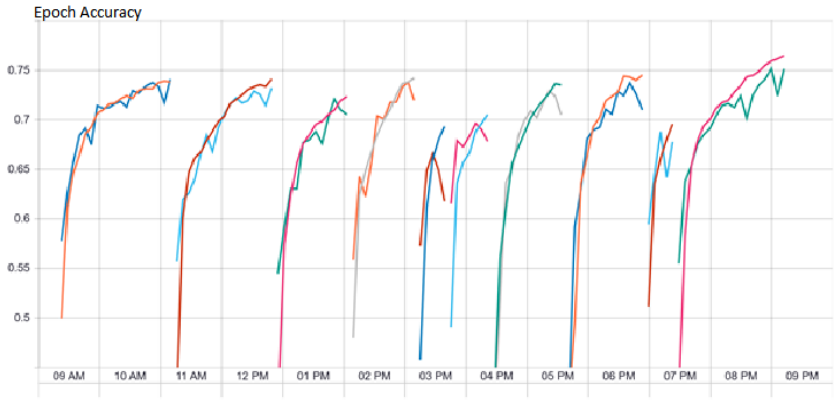
Top Models Re-Training

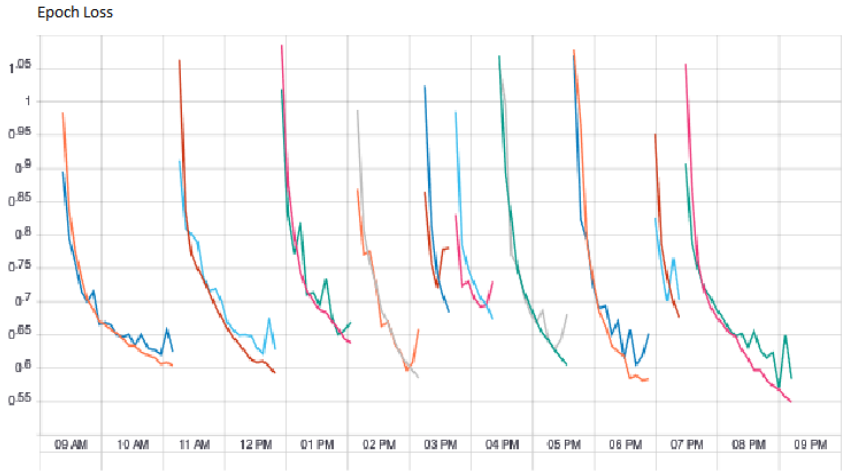


Top 10 Models Re-Training Results

Top preforming models' hyper parameter sets are collected, recreated and trained with 300x300 input size and evaluated on the test data set. These 10 models took an additional 10 hours to train and evaluate.

Name	Status	Date modified	Type	Size
ReTrained_4_conv_16_nodes_1_dense_2_kernelSz_adam_optimizer_0_01lr_0_0_dropr_20200226_091640_T.h5	✓	2020-02-26 11:09 AM	H5 File	55,701 KB
ReTrained_4_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_01lr_0_0_dropr_20200226_110937_T.h5	✓	2020-02-26 12:48 PM	H5 File	49,535 KB
ReTrained_4_conv_32_nodes_2_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr_20200226_124902_T.h5	✓	2020-02-26 1:50 PM	H5 File	178,110 KB
ReTrained_5_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_001lr_0_0_dropr_20200226_140305_T.h5	✓	2020-02-26 3:03 PM	H5 File	20,073 KB
ReTrained_5_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_01lr_0_0_dropr_20200226_150858_T.h5	✓	2020-02-26 3:27 PM	H5 File	20,073 KB
ReTrained_5_conv_32_nodes_0_dense_2_kernelSz_adam_optimizer_0_001lr_0_0_dropr_20200226_153846_T.h5	✓	2020-02-26 4:09 PM	H5 File	2,718 KB
ReTrained_5_conv_32_nodes_0_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr_20200226_162107_T.h5	✓	2020-02-26 5:21 PM	H5 File	8,673 KB
ReTrained_5_conv_32_nodes_0_dense_4_kernelSz_adam_optimizer_0_01lr_0_0_dropr_20200226_173321_T.h5	✓	2020-02-26 6:40 PM	H5 File	8,673 KB
ReTrained_5_conv_32_nodes_1_dense_3_kernelSz_adam_optimizer_0_001lr_0_0_dropr_20200226_185302_T.h5	✓	2020-02-26 7:10 PM	H5 File	80,026 KB
ReTrained_5_conv_32_nodes_1_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr_20200226_192246_T.h5	✓	2020-02-26 8:59 PM	H5 File	63,677 KB





Name	Train Accuracy	Validation Accuracy	Test Accuracy
5_conv_32_nodes_1_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr	75.9%	75.2%	77.2%
4_conv_16_nodes_1_dense_2_kernelSz_adam_optimizer_0_01lr_0_0_dropr	73.8%	73.4%	74.8%
5_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_001lr_0_0_dropr	73.7%	73.3%	74.5%
4_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_01lr_0_0_dropr	73.5%	72.6%	74.2%
5_conv_32_nodes_0_dense_4_kernelSz_adam_optimizer_0_01lr_0_0_dropr	74.3%	73.7%	73.6%
4_conv_32_nodes_2_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr	71.0%	72.1%	70.9%

5_conv_32_nodes_0_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr	72.7%	73.0%	69.1%
5_conv_32_nodes_0_dense_2_kernelSz_adam_optimizer_0_001lr_0_0_dropr	70.5%	69.6%	68.5%
5_conv_32_nodes_1_dense_3_kernelSz_adam_optimizer_0_001lr_0_0_dropr	73.6%	68.8%	68.5%
5_conv_16_nodes_1_dense_3_kernelSz_adam_optimizer_0_01lr_0_0_dropr	66.0%	66.6%	61.9%

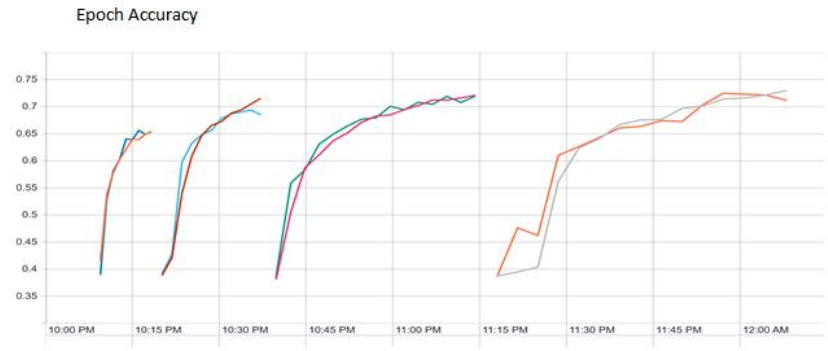
Looking at the retrained results, descending sorted by their test accuracy, most of the models training results tracks very well towards its Validation accuracy to within 1% difference (highlighted in yellow if outside of the range). This shows that the early stop setting was effective at making sure that the training process is not dramatically overfitting.

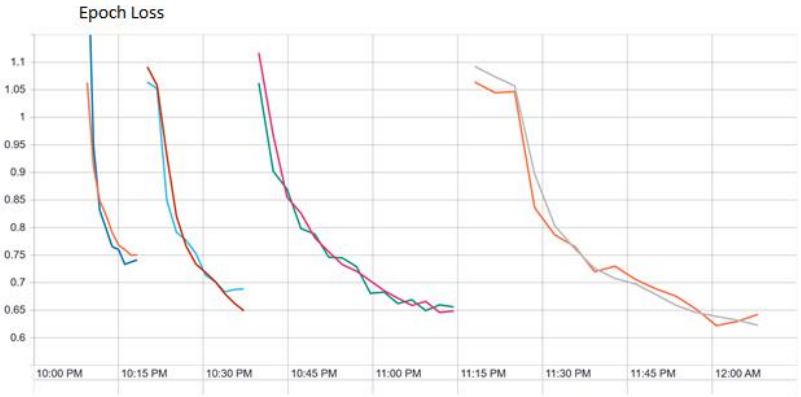
On the other hand, looking at the bottom 4 models, their test accuracy differs their training accuracy my more than 1% (highlighted in red), this shows that despite the effort in cleaning, balancing the class samples and early stopping strategies, some level of overfitting still occurred and having the test split is always the best practice to evaluating the true model performance.

The model 5_conv_32_nodes_1_dense_4_kernelSz_adam_optimizer_0_001lr_0_0_dropr achieved the best test accuracy.

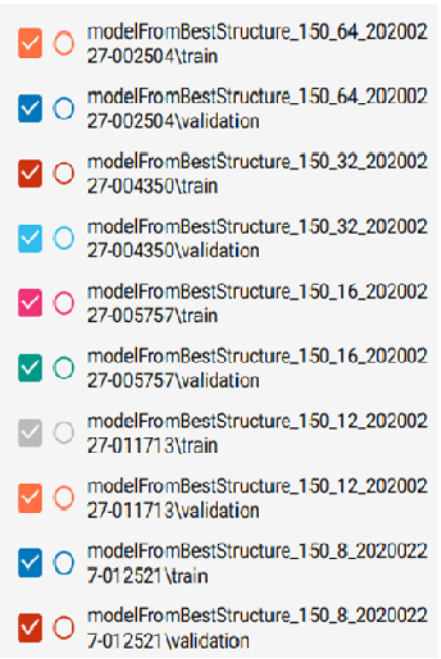
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 297, 297, 32)      1568
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 148, 148, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 145, 145, 32)      16416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 72, 72, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 69, 69, 64)        32832
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 34, 34, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 31, 31, 128)       131200
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 256)       524544
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 512)               4719104
_________________________________________________________________
activation (Activation)      (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1539
=================================================================
Total params: 5,427,203
Trainable params: 5,427,203
Non-trainable params: 0
_________________________________________________________________
```

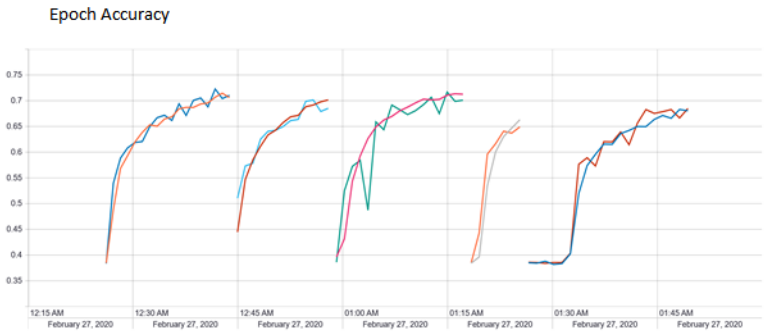
Effect of Using Larger Input Size/Resolution

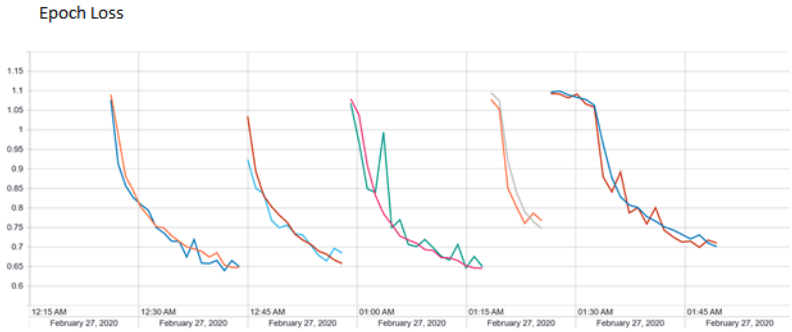




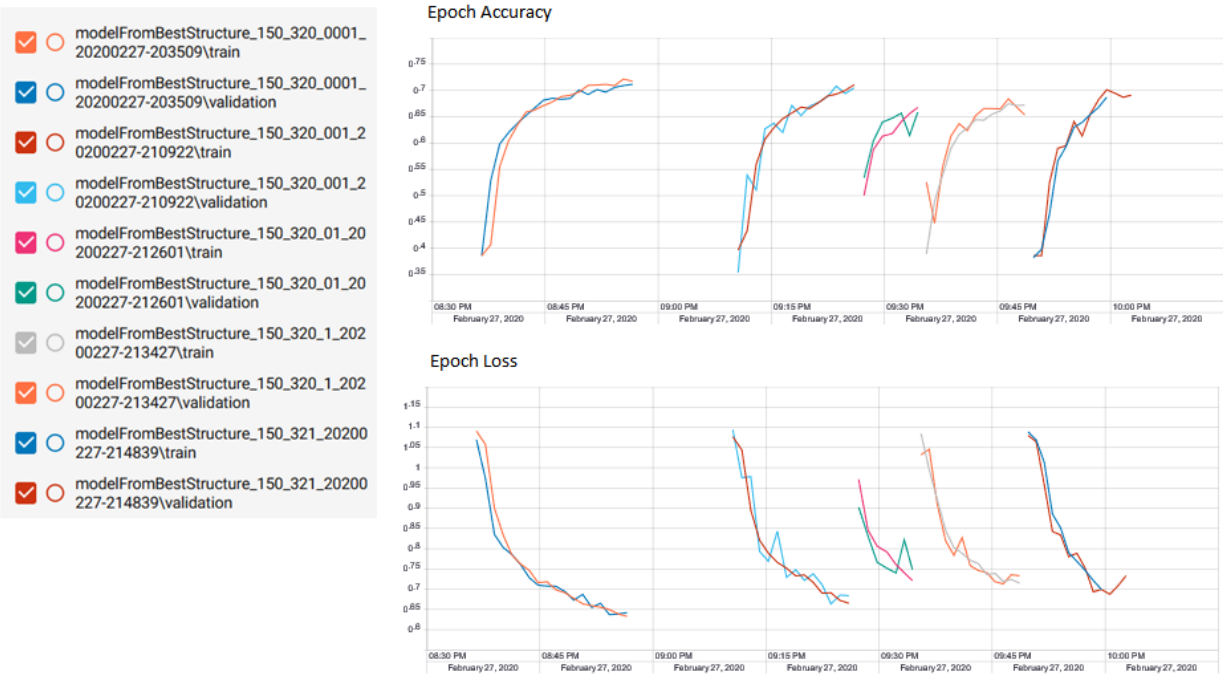
Effect of Using Different Batch Size





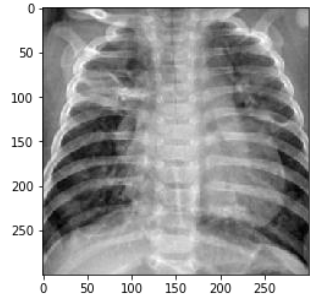
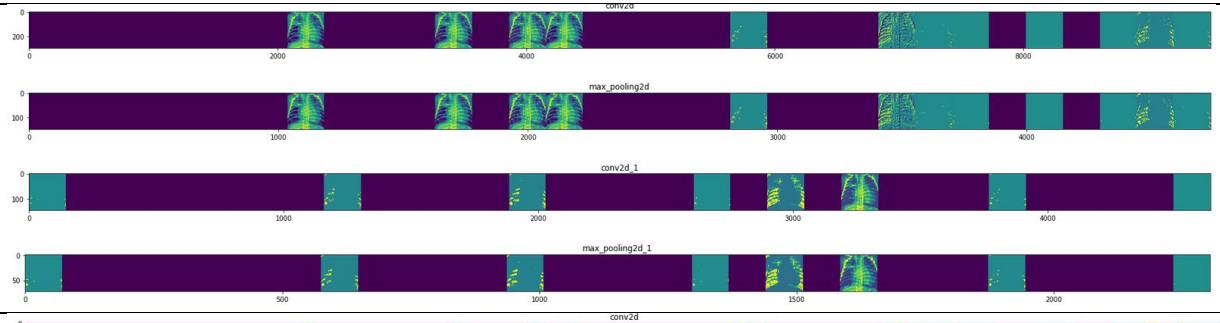
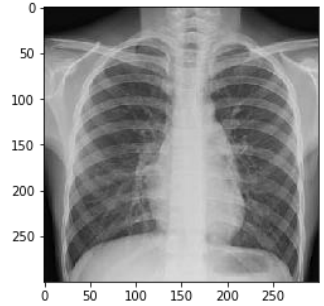
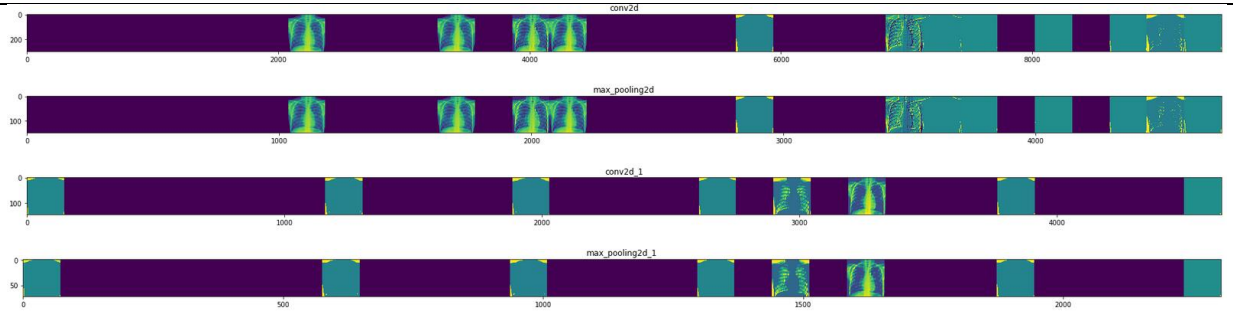


Effect of Learning Rate



Model Visualization

We try to visualize the first 2 layers of our top performing model and see how the model highlights the images. Objective of this activity was to see which part of the image the model is focusing. In future we can use to focus on the part of image it is focusing.

Image Type	Sample X-Ray Image	first 2 layers
Bacteria		
Normal		
Virus