AUTOMATED DIAGNOSIS OF COVID-19 USING X-RAY IMAGES

NG YIT TYN

FACULTY OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2021

AUTOMATED DIAGNOSIS OF COVID-19 USING X-RAY IMAGES

ABSTRACT

As of the end of 2019, the world suffered from a disease caused by the SARS-CoV-2 virus, which has become the pandemic COVID-19. This aggressive disease deteriorates the human respiratory system. Patients with COVID-19 can develop symptoms that belong to the common flu, pneumonia, and other respiratory diseases in the first four to ten days after they have been infected. As a result, it can cause misdiagnosis between patients with COVID-19 and typical pneumonia. Some deep-learning techniques can help physicians to obtain effective pre-diagnosis. The content of this report consists of deep-learning models, specifically a convolutional neural network (CNN) that build from scratch with no initialize weights, which allows us to obtain newly trained models to classify COVID-19, pneumonia, and healthy patients. Thanks to deep learning, an effective mechanism that can provide a promising solution was built. High accuracy of 96.38% (with a recall of 78.6% and a precision of 64.24%) was achieved in the detection of COVID-19 X-ray images from normal, and severe acute respiratory syndrome cases.

Keywords: COVID-19; pneumonia; classification; deep learning; convolutional; network

ACKNOWLEDGEMENTS

This project report would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

A special gratitude I would like to give to our Image Processing lecturer, Dr. Woo Chaw Seng, who contribute to stimulating suggestions and the lessons taught throughout the semester, had brushed up my image processing skills which help me to complete this project. The constant supervision and support as well as necessary information regarding writing the report to complete this project were much appreciated.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of Kaggle and Prashant Patel, who permitted to use all the chest X-ray dataset to complete this project. Last but not least, a special thanks goes to my course mates who had answered my questions and gave suggestions to complete the report.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

COVID-19 is a newly known disease that can be misdiagnosed as common pneumonia. Due to the easy propagation through the air and contact with contaminated objects and people, COVID-19 has become a major pandemic. According to the World Health Organization (WHO), as of 30 June 2020, there have been more than 10 million cases of COVID-19 and more than half a million confirmed deaths. COVID-19 happens due to the SARS-CoV-2 virus infecting the lung and the respiratory system. This disease is a major concern for the medical field due to the fuzzy symptoms that present early contagious people. Patients with COVID-19 can develop symptoms that belong to common flu, pneumonia, and other respiratory diseases in the first four to ten days since they were infected.

It is possible to support the diagnosis of respiratory and lung diseases through computer-assisted diagnosis (CAD). Within CAD, we can find techniques that obtain images from the internals of the body, such as chest X-ray (CXR) images, computed tomography (CT) as well as magnetic resonance imaging (MRI).

Nowadays, the most effective image examination for diagnosing COVID-19 on infected patients is the CT images due to their high sensitivity compared to CXRs. However, the analysis will be using CXRs instead of other examinations that provide more accurate results. The main reason is that getting CXRs is more accessible for people, especially in rural and isolated areas. There will be more potential data available.

There are several machine-learning (ML) techniques that can help develop CAD tools. One of the best methods in the current state for computer vision (CV) tasks is the convolutional neural networks (CNN). In this project, we propose to use a basic CNN, which allows to train models to classify, in this case, among COVID-19, pneumonia, and healthy patients.

CHAPTER 2: OBJECTIVES

In this project, we’ll explore the feasibility and difficulties in building a system that is capable of detecting various causes of pneumonia in chest X-rays, using a relatively promising CNN model and a constrained dataset.

The first objective is to classify normal, COVID-19, and pneumonia cases from chest X-ray images. Next, the second objective is to evaluate the accuracy and loss of chest X-ray classification. Both positives and negatives will be reduced to identify COVID-19 cases more accurately.

CHAPTER 3: DATA SOURCE

In this chapter, datasets that have been used to carry out this research experiment will be explained.

3.1 Data Source

The dataset used in this project is obtained from Kaggle, which was compiled by Prashant Patel 4 months ago (https://www.kaggle.com/prashant268/chest-xray-covid19-pneumonia). It was organized into two folders (train, test) while both train and test contain three subfolders (COVID19, PNEUMONIA, NORMAL). In total, this dataset consists of 6432 x-ray images, while test data occupied 20% of total images.

3.2 COVID-19 Dataset

There are not many publicly available datasets that allow researchers to try on this new disease. Nonetheless, Montreal University has opened a regularly updated dataset for COVID-19 and other respiratory diseases to contribute to diagnostic tools.

The COVID-19 Image Data Collection, presented by Cohen et al. contained 539 images from different respiratory diseases, such as pneumonia, SARS, MERS, among others (until before 1 June 2020).

In this dataset, 287 Chest X-Rays were specifically from patients with COVID-19. Most of them are from adult patients and include people from 12 to 87 years old and of different nationalities. Only COVID-19 images from this dataset are used in this research. (https://github.com/ieee8023/covid-chestxray-dataset).

Besides, 289 chest X-rays were obtained from two Github repo (Figure1-COVID-chestxray-dataset, Actualmed-COVID-chestxray-dataset) which were prepared by agchung. (https://github.com/agchung). 460 of them are used as training set, while 116 of them are used as test set.

3.3 Other Dataset

The normal and pneumonia dataset used in performing this experiment were collected from the article published in 2018 by the team of Kermany et al. This dataset contains a total of 5863 normal and pneumonia chest X-ray images which have been selected from retrospective cohorts of pediatric patients from 1 to 5 years old.

3418 images of pneumonia-infected patients from this dataset will be used in training dataset, while 855 pneumonia images will be used in test dataset. Otherwise, 1266 images of normal images from this dataset will be used in training dataset, while 317 normal images will be used in test dataset.

CHAPTER 4: ANALYSIS & DESIGN

All experiments are implemented using Python 3.6 programming language, Keras, and Tensorflow deep learning framework. Besides, Google Colab is used to run the program while OpenCV as our main graphic processor software.

4.1 Methodology

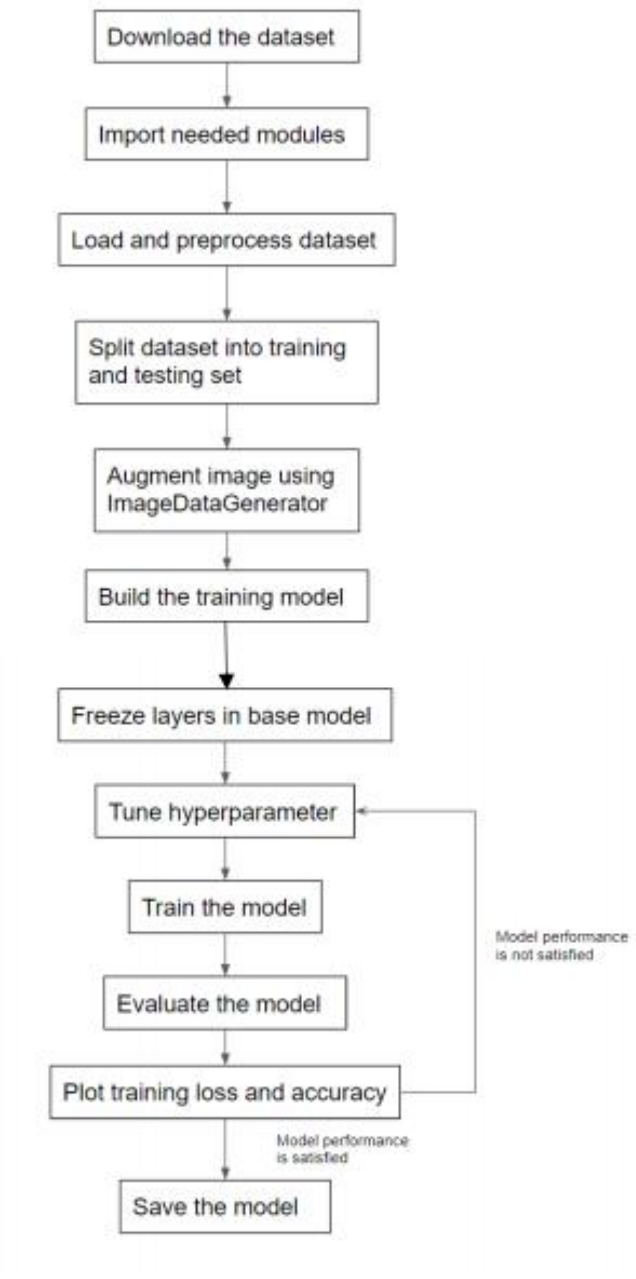

Download the dataset
Import needed modules
Load and preprocess dataset
Split dataset into training and testing set
Augment image using ImageDataGenerator
Build the training model
Freeze layers in base model
Tune hyperparameter
Train the model
Evaluate the model
Plot training loss and accuracy
Model performance is not satisfied
Model performance is satisfied
Save the model

Figure 4-1 Algorithm Flowchart

Before we start training the model, we need to get relevant data so that the model can learn from the data by identifying out certain relations and common features. As Google Colab can mount with Google Drive to access the directories in it, hence, we just need to download the chest X-ray dataset and upload it to Google Drive for analysis and training purposes.

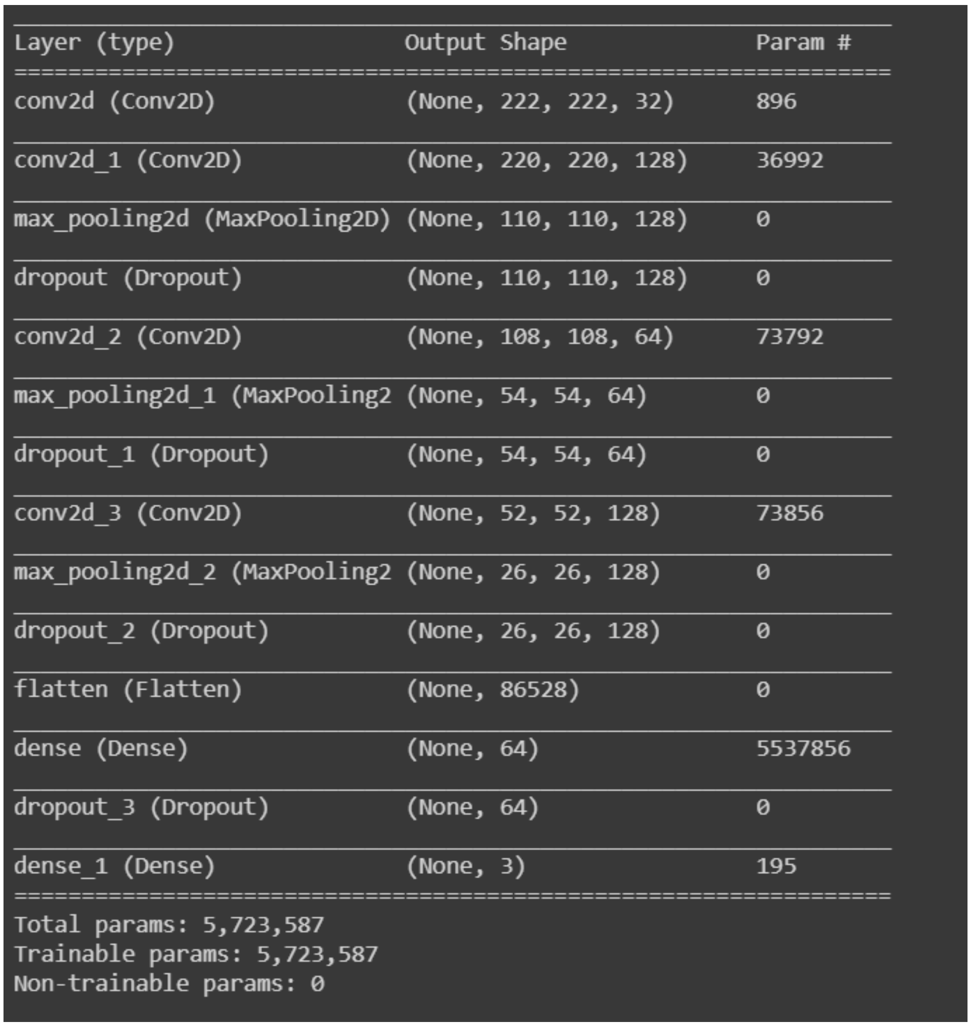
Then, we need to import the required modules or libraries in order to ease the job of building the code. There are several important libraries used in this project, such as tensorflow, keras, numpy, matplotlib and etc.

Image preprocessing by rescaling each image pixel with 1/255 to return value in the range of [0,1] is first performed on both train and test datasets. Next, data augmentation will be performed on the train dataset only as it helps the model become more generalize and robust. Test dataset will not perform any augmentation as there’s no point and benefit to augment it. This action can artificially create new training data from existing training data. Hence, it can help to increase the amount of data and prevent overfitting on the models. Minor alterations such as flips, translations, or rotations will be applied but only with the changes that make more sense. The X-ray images will only be rotated randomly for +- 15 degrees, zoomed on a range of +- 10%, and perform horizontal flip as existing chest X-ray wouldn’t present in vertical flip. The neural network will treat these augmented data as distinct images.

After data augmentation, we will then start building the training model. Convolution Neural Network (CNN) will be used in this project.

4.2 Basic Convolution Neural Network

First, a basic convolution neural network is built from scratch instead of using a pre-trained model. It seems more optimal to train a small dataset on a smaller model with lesser parameters as this can help saving hours on transfer learning.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896       
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 128)     36992     
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 110, 110, 128)     0         
_________________________________________________________________
dropout (Dropout)            (None, 110, 110, 128)     0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 108, 108, 64)      73792     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 64)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 54, 54, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 128)       0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 26, 26, 128)       0         
_________________________________________________________________
flatten (Flatten)            (None, 86528)             0         
_________________________________________________________________
dense (Dense)                (None, 64)                5537856   
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195       
=================================================================
Total params: 5,723,587
Trainable params: 5,723,587
Non-trainable params: 0
```

Figure 4-2 Model Summary

For the input layer, a convolutional 2D layer with 32 filters, 3 x 3 kernel size, Rectified Linear Unit (ReLu) activation function and 224 x 224 x 3 input shape will be used. Hence, the images need to be resized into 224 x 224 with 3 channels to be compatible with the model input size.

Then, 3 simple hidden layers will be concatenated behind. The hidden layers are formed with a combination of convolutional 2D layer, max-pooling 2D layer, and a dropout layer. The convolutional 2D layer used in hidden layers have a similar structure to the input layer, besides the filters are 128, 64, and 128 for three hidden layers respectively. Max pooling layer is then applied with a pooling size of 2 x 2 to down-sample the dimension of feature maps by taking the maximum value over the window defined. A dropout layer with a 0.25 dropout rate is then added after the pooling layer to prevent overfitting.

The 3-dimension output features of the hidden layer are then flattened into 1-dimension to feed into the dense layer. The dense layer will produce output of 64 neurons, then dropout with a dropout rate of 0.5 will be performed on them to cancel out some neurons. The outputs will be fed into another dense layer with 3 outputs, which correspond to three predicted labels, which are COVID19, NORMAL, and PNEUMONIA. Softmax activation is used at the output dense layer so that the three predicted output scores added up will become 1. The highest score from the predicted outcome shows that image input has the highest probability belong to that respective class.

After the architecture of the model has been built up, the model will be compiled. The optimizer used in this model is Adam as it seems always works better (faster and more reliably reaching a global minimum) when minimizing the cost function in model training. A default initializes learning rate (1e-3) will be applied and it will decay with a rate of 1e-3 divide by the number of epochs.

The loss function used in this model is Categorical Crossentropy due to there are more than two label classes, and the labels provided are in one-hot representation. It will compute the loss between the labels and predictions. In addition, the metric used in evaluating the model is ‘accuracy’.

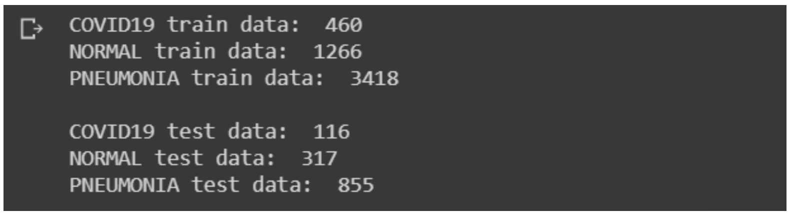
After the model is done compiled, image data will be feed into the model in batches via train generator and validation generator. There are 30 epochs with 322 steps per epochs in the whole training process and it took approximately 2.3 hours.

CHAPTER 5: EXPERIMENTAL RESULTS

In this chapter, the input data used, and the output results will be explained. The code, model and plot images are available in Google Colab, Google Drive or Github.

5.1 Exploratory Data Analysis

Exploratory data analysis is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. For classification problems, this might include looking at the distributions of variables or checking for any meaningful patterns across different classes.



```
COVID19 train data:  460
NORMAL train data:  1266
PNEUMONIA train data:  3418

COVID19 test data:  116
NORMAL test data:  317
PNEUMONIA test data:  855
```

Figure 5-1 Data Distribution

Figure 5.1 shows the image distribution of chest X-ray images. There are 5144 train images in total, with a combination of 460 COVID-19 data, 1266 normal data, and 3418 pneumonia data. Meanwhile, there are 1288 test images in total, with a combination of 116 COVID-19 data, 317 normal data, and 855 pneumonia data.

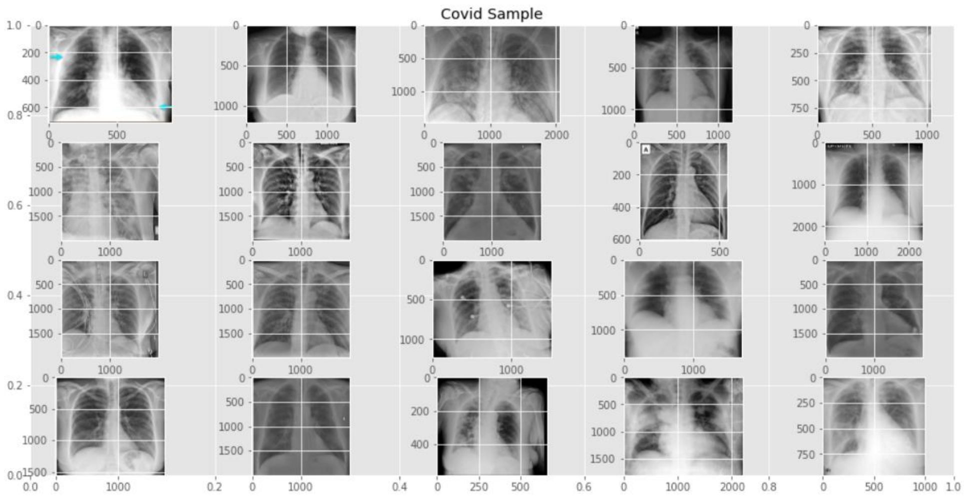


Figure 5-2 COVID-19 Class Sample Data

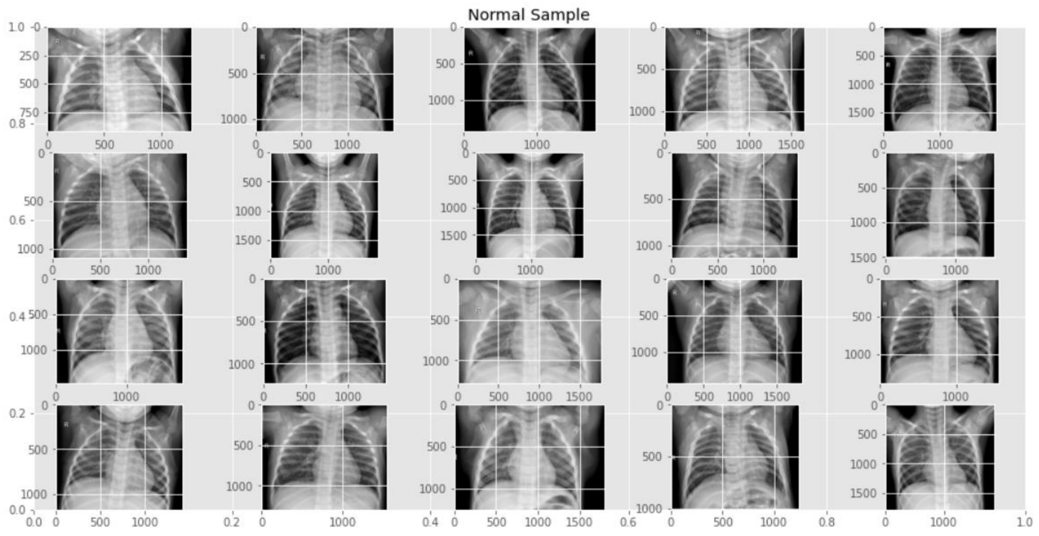


Figure 5-3 Normal Class Sample Data

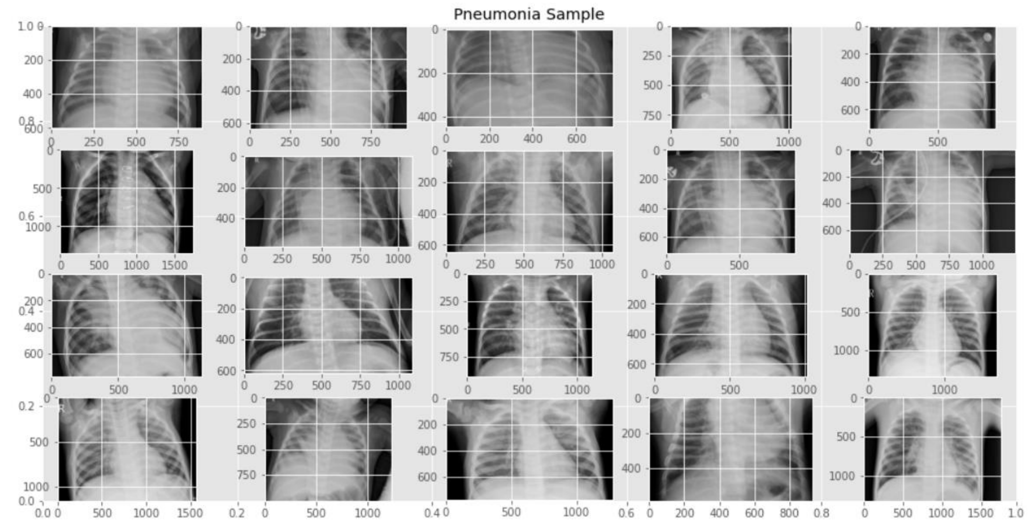


Figure 5-4 Pneumonia Class Sample Data

Figure 5.2 shows raw data that randomly sampled from the COVID-19 class, Figure 5.3 shows raw data that randomly sampled from the normal class, while Figure 5.4 shows raw data that randomly sampled from the pneumonia class.

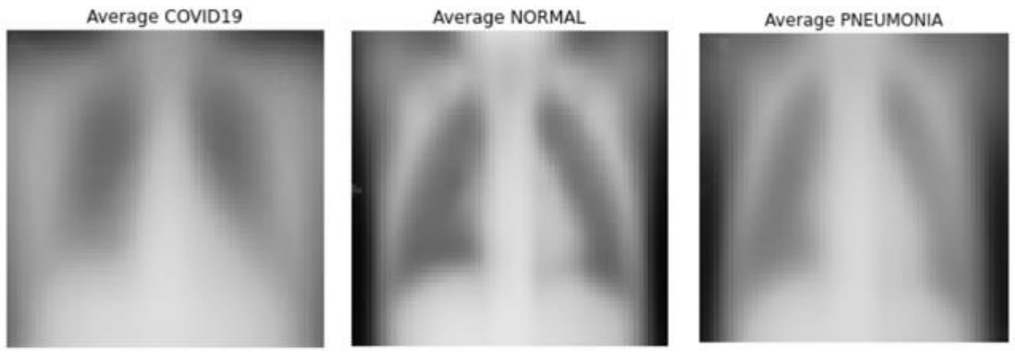


Figure 5-5 Average Images

Figure 5.5 shows the average image looks for each class. Based on the figure above, we can see that pneumonia X-ray shows higher obstruction around the chest area comparing COVID-19 and normal X-ray.

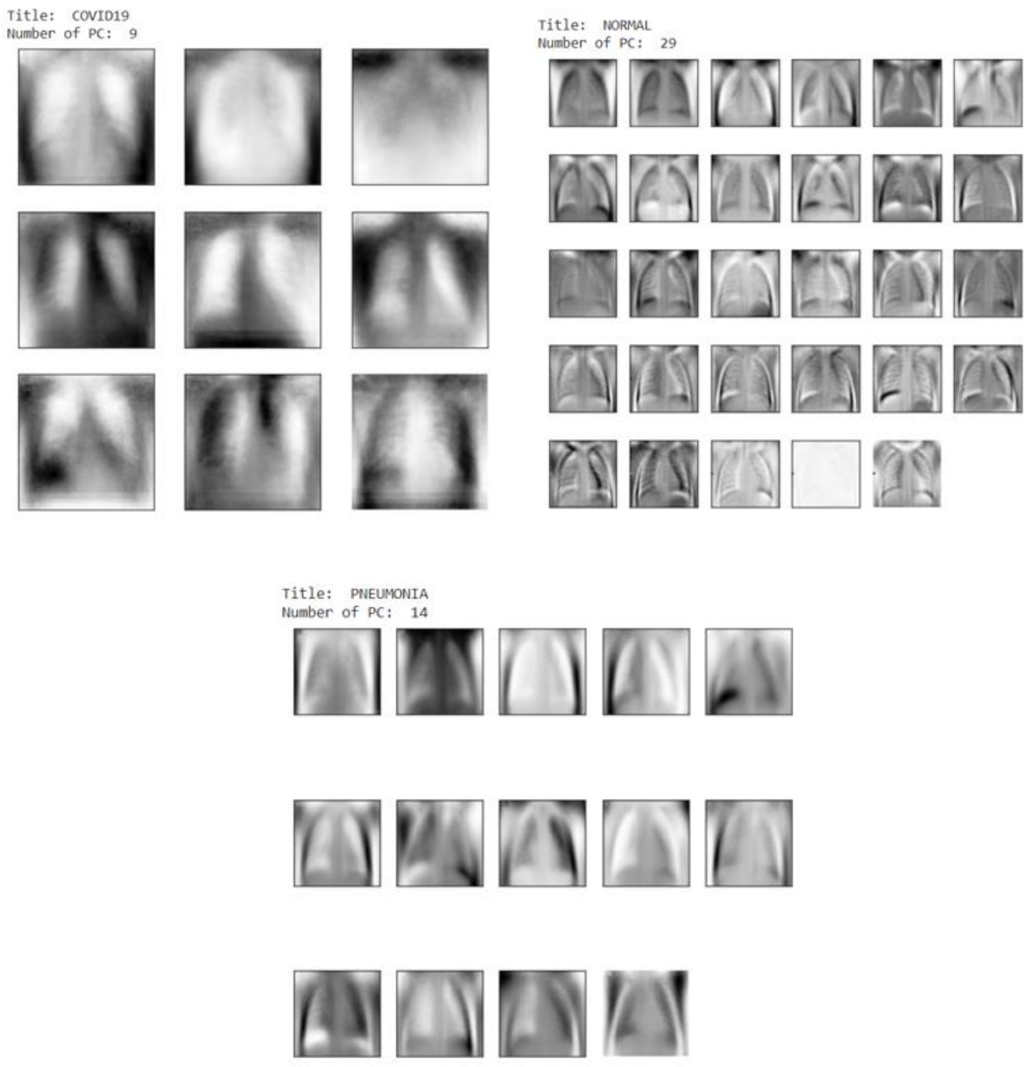
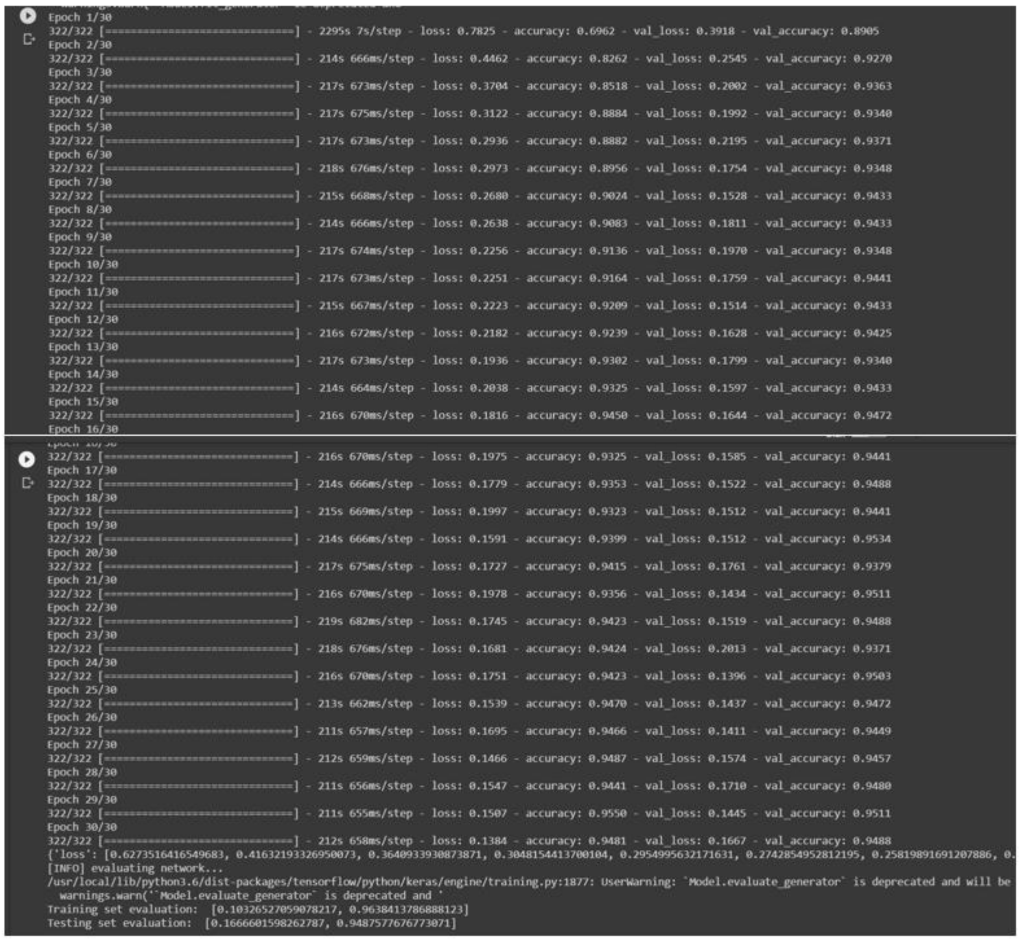


Figure 5-6 Eigenimages

Lastly, a dimension reduction technique, such as the principal component analysis (PCA) was used to visualize the components that describe each class the best. Here the principal components that describe 70% of variability for each class were visualized. Figure 5.6 shows that the eigenimages of healthy X-ray images have much more edge definitions around rib cages and organs compared to the COVID-19 class and pneumonia class.

5.2 Basic Convolution Neural Network



```
Epoch 1/30
322/322 [==============================] - 2295s 7s/step - loss: 0.7825 - accuracy: 0.6962 - val_loss: 0.3918 - val_accuracy: 0.8905
Epoch 2/30
322/322 [==============================] - 214s 666ms/step - loss: 0.4462 - accuracy: 0.8262 - val_loss: 0.2545 - val_accuracy: 0.9270
Epoch 3/30
322/322 [==============================] - 217s 673ms/step - loss: 0.3704 - accuracy: 0.8518 - val_loss: 0.2002 - val_accuracy: 0.9363
Epoch 4/30
322/322 [==============================] - 217s 675ms/step - loss: 0.3122 - accuracy: 0.8884 - val_loss: 0.1992 - val_accuracy: 0.9340
Epoch 5/30
322/322 [==============================] - 217s 673ms/step - loss: 0.2936 - accuracy: 0.8882 - val_loss: 0.2195 - val_accuracy: 0.9371
Epoch 6/30
322/322 [==============================] - 218s 676ms/step - loss: 0.2973 - accuracy: 0.8956 - val_loss: 0.1754 - val_accuracy: 0.9348
Epoch 7/30
322/322 [==============================] - 215s 668ms/step - loss: 0.2680 - accuracy: 0.9024 - val_loss: 0.1528 - val_accuracy: 0.9433
Epoch 8/30
322/322 [==============================] - 214s 666ms/step - loss: 0.2638 - accuracy: 0.9083 - val_loss: 0.1811 - val_accuracy: 0.9433
Epoch 9/30
322/322 [==============================] - 217s 674ms/step - loss: 0.2256 - accuracy: 0.9136 - val_loss: 0.1970 - val_accuracy: 0.9348
Epoch 10/30
322/322 [==============================] - 217s 673ms/step - loss: 0.2251 - accuracy: 0.9164 - val_loss: 0.1759 - val_accuracy: 0.9441
Epoch 11/30
322/322 [==============================] - 215s 667ms/step - loss: 0.2223 - accuracy: 0.9209 - val_loss: 0.1514 - val_accuracy: 0.9433
Epoch 12/30
322/322 [==============================] - 216s 672ms/step - loss: 0.2182 - accuracy: 0.9239 - val_loss: 0.1628 - val_accuracy: 0.9425
Epoch 13/30
322/322 [==============================] - 217s 673ms/step - loss: 0.1936 - accuracy: 0.9302 - val_loss: 0.1799 - val_accuracy: 0.9340
Epoch 14/30
322/322 [==============================] - 214s 664ms/step - loss: 0.2038 - accuracy: 0.9325 - val_loss: 0.1597 - val_accuracy: 0.9433
Epoch 15/30
322/322 [==============================] - 216s 670ms/step - loss: 0.1816 - accuracy: 0.9450 - val_loss: 0.1644 - val_accuracy: 0.9472
Epoch 16/30
```

```
322/322 [==============================] - 216s 670ms/step - loss: 0.1975 - accuracy: 0.9325 - val_loss: 0.1585 - val_accuracy: 0.9441
Epoch 17/30
322/322 [==============================] - 214s 666ms/step - loss: 0.1779 - accuracy: 0.9353 - val_loss: 0.1522 - val_accuracy: 0.9488
Epoch 18/30
322/322 [==============================] - 215s 669ms/step - loss: 0.1997 - accuracy: 0.9323 - val_loss: 0.1512 - val_accuracy: 0.9441
Epoch 19/30
322/322 [==============================] - 214s 666ms/step - loss: 0.1591 - accuracy: 0.9399 - val_loss: 0.1512 - val_accuracy: 0.9534
Epoch 20/30
322/322 [==============================] - 217s 675ms/step - loss: 0.1727 - accuracy: 0.9415 - val_loss: 0.1761 - val_accuracy: 0.9379
Epoch 21/30
322/322 [==============================] - 216s 670ms/step - loss: 0.1978 - accuracy: 0.9356 - val_loss: 0.1434 - val_accuracy: 0.9511
Epoch 22/30
322/322 [==============================] - 219s 682ms/step - loss: 0.1745 - accuracy: 0.9423 - val_loss: 0.1519 - val_accuracy: 0.9488
Epoch 23/30
322/322 [==============================] - 218s 676ms/step - loss: 0.1681 - accuracy: 0.9424 - val_loss: 0.2013 - val_accuracy: 0.9371
Epoch 24/30
322/322 [==============================] - 216s 670ms/step - loss: 0.1751 - accuracy: 0.9423 - val_loss: 0.1396 - val_accuracy: 0.9503
Epoch 25/30
322/322 [==============================] - 213s 662ms/step - loss: 0.1539 - accuracy: 0.9470 - val_loss: 0.1437 - val_accuracy: 0.9472
Epoch 26/30
322/322 [==============================] - 211s 657ms/step - loss: 0.1695 - accuracy: 0.9466 - val_loss: 0.1411 - val_accuracy: 0.9449
Epoch 27/30
322/322 [==============================] - 212s 659ms/step - loss: 0.1466 - accuracy: 0.9487 - val_loss: 0.1574 - val_accuracy: 0.9457
Epoch 28/30
322/322 [==============================] - 211s 656ms/step - loss: 0.1547 - accuracy: 0.9441 - val_loss: 0.1710 - val_accuracy: 0.9480
Epoch 29/30
322/322 [==============================] - 211s 655ms/step - loss: 0.1507 - accuracy: 0.9550 - val_loss: 0.1445 - val_accuracy: 0.9511
Epoch 30/30
322/322 [==============================] - 212s 658ms/step - loss: 0.1384 - accuracy: 0.9481 - val_loss: 0.1667 - val_accuracy: 0.9488
{'loss': [0.6273516416549683, 0.41632193326950073, 0.3640933930873871, 0.3048154413700104, 0.2954995632171631, 0.2742854952812195, 0.25819891691207886, 0.
[INFO] evaluating network...
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.evaluate_generator` is deprecated and will be
  warnings.warn('`Model.evaluate_generator` is deprecated and '
Training set evaluation:  [0.10326527059078217, 0.9638413786888123]
Testing set evaluation:  [0.1666601598262787, 0.9487577676773071]
```

Figure 5-7 Model Training

Figure 5.7 shows the model training progress, which contains 30 epochs with 322 steps per epoch. Training loss, validation loss, training accuracy, and validation accuracy were keep tracked during the training process. After evaluating the model, the training accuracy shows 0.9638, and the testing accuracy shows 0.9488.

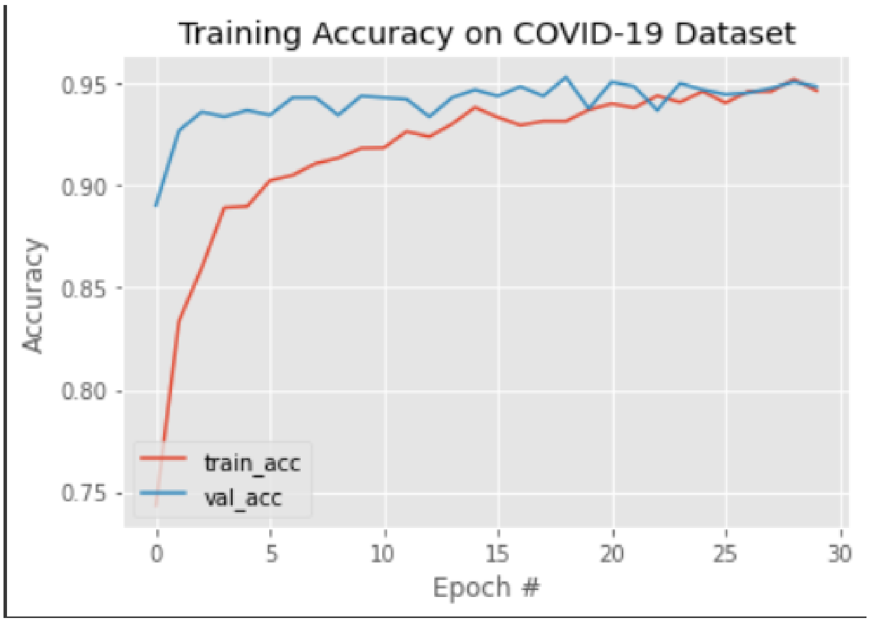


Figure 5-8 Training and Validation Accuracy

The line graph above shows the model accuracy throughout the training process. We can observe that both training and validation accuracy increasing towards the value of one and converge at the end of the training. Besides, there is no over-fitting nor under-fitting issue happened based on the accuracy graph plot.

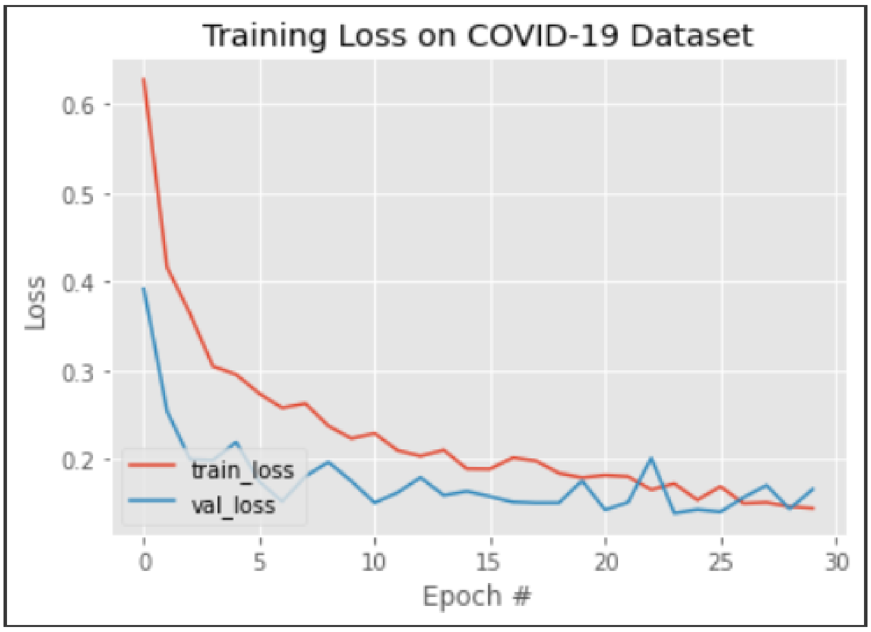


Figure 5-9 Training and Validation Loss

The line graph above shows the model loss throughout the training process. We can observe that both training and validation loss decrease towards zero and converge at the end of the training.

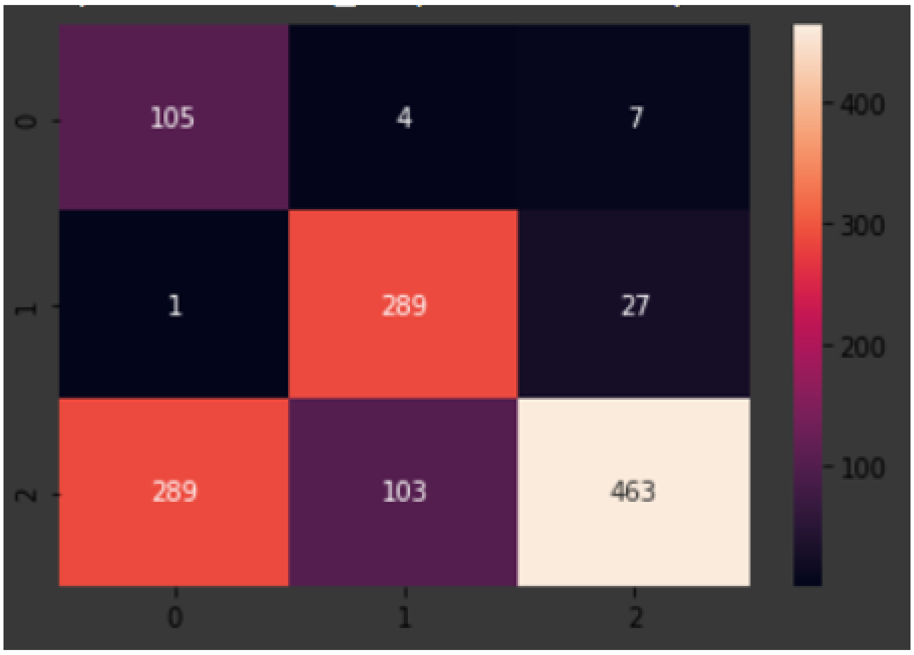
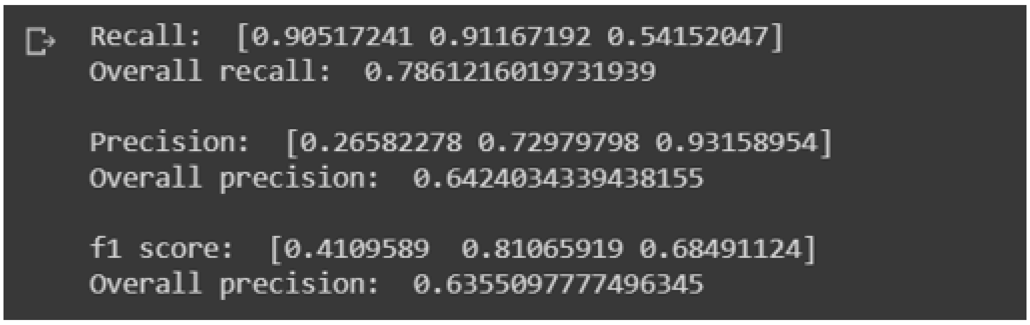


Figure 5-10 Confusion Matrix

Confusion matrix class indices are shown as: {0: Covid19; 1: Normal; 2: Pneumonia}. After decoding the confusion matrix, out of 116 COVID-19 affected patients, 105 patients were correctly classified, and 11 patients were wrongly classified. Out of 317 normal patients, 289 patients were correctly classified, and 28 patients were wrongly classified. Meanwhile, out of 855 pneumonia affected patients, 463 patients were correctly classified, and 392 patients were wrongly classified.

The results getting is good, but still, the accuracy still has room to improve in order to fulfill our intention.



```
Recall:  [0.90517241 0.91167192 0.54152047]
Overall recall:  0.7861216019731939

Precision:  [0.26582278 0.72979798 0.93158954]
Overall precision:  0.6424034339438155

f1 score:  [0.4109589  0.81065919 0.68491124]
Overall precision:  0.6355097777496345
```

Figure 5-11 Precision, Recall and F1 Score

Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted observations to the total observations. The higher the accuracy, the better the model performed. However, accuracy is a great measure only when the dataset is symmetric. Since chest X-ray data have imbalanced data, therefore other parameters such as precision, recall, and F1 score are required to evaluate the performance of the model.

Recall is the ratio of correctly predicted positive observations to all observations in the actual class. We have got a recall of 0.786 which is good for this model as it’s above 0.5.

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. In this model, the precision achieved 0.6424 which is still fine, but there’s still a lot of improvement space.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall. In our case, the F1 score is 0.636.

CHAPTER 6: DISCUSSION

Due to the high availability of large-scale annotated image datasets, convolutional neural network (CNN) based approaches are the state-of-the-art in various computer vision tasks, including image recognition. A considerable research effort is currently being directed toward further improving CNNs by focusing on model architectures and training techniques. Although Convolutional Neural Networks has tremendous success in the Computer Vision field, it also has unavoidable limitations.

6.1 Strengths of Convolutional Neural Network

Classification using the CNN model is widely used as they are powerful in achieving high accuracy with a minimum error rate and high performance in terms of precision. CNN's have shown remarkable classification results using standard architectures. One of the main advantages of the deep learning model like CNN is its capacity to execute feature engineering on its own. It will scan the data to search for features that correlate and combine them to enable faster learning without being explicitly told to do so. Besides, it can be trained using different data formats, such as pictures, videos, files, etc., and still derive insights that are relevant to the purpose of its training.

6.2 Limitations of Convolutional Neural Network

Although CNN performs well in image classification or recognition tasks, it portrays a few limitations that cause the model to be imperfect. First and foremost, the amount of data needed for training will be much higher comparing other algorithms or methods. The reason is that the task of a deep learning algorithm is two-folded. It needs to first learn about the domain, and only then solve the problem. When the training begins, the algorithm starts from scratch. To learn about a given domain, the algorithm needs a huge number of parameters to tune and "play around with". Besides, as the core of deep learning is black-box, the lack of transparency in the model "thinking" process makes it hard to predict and analyze when failures occur.

6.3 Expected Results

The trained model has fulfilled the expectation to classify the chest X-ray image into COVID-19, NORMAL, or PNEUMONIA. It performs well in both COVID-19 class and normal class, but it didn't perform well in pneumonia class as only around 60% were classified correctly. Besides, the dataset used is just a very small part compared to millions of real cases in the outside world. Hence, the current model can only be used to assist the medical front liner in examining the patients' condition, but human intervention is still required.

6.4 Suggestion for Future Improvements

The robustness of the model can be increased with more X-ray scans so that the model is generalizable. As COVID-19 data are the least in the current dataset and it's far from pneumonia for at least ten times, either compiled more COVID-19 data from other sources such as Github repository, medical school, open-source dataset, etc. or use sampling techniques such as under-sampling abundant class or over-sampling insufficient class to balance the dataset.

Besides, more hidden layers can be added to the model. Successive layers of the network learn successively more sophisticated features, which build on the features from preceding layers. Hence, deeper networks with more hidden layers are better in predicting image as they develop a more granular or detailed representation of a "thing" being recognized.

CHAPTER 7: CONCLUSION

With the aids of the model architecture and its core back-propagation algorithm, a deep-learning model such as Convolutional Neural Network can analyze and classify images into respective classes. However, images are represented in digital form, which image classification task is heavily relying on raw pixel data. Hence, image processing plays a vital role in finishing the task, and to ease the task in certain ways.

Raw images are represented in rows and columns, which each element in the matrix is called a pixel. By using the image processing technique, the image is first scaling into a range from 0 to 1 in order to standardize the pixel values, which originally pixel value range from 0 (black) to 255 (white).

Depending on the color scale there are various channels in an image, each channel representing the pixel values for one particular color. RGB (Red, green, blue) is the commonly used color scale and in this project, all images that have been used are RGB images.

Besides, data augmentations which used to increase the data available are using image processing techniques too. For instance, cropping that has been used in the experiment is just taking out slices from the image array randomly or just the border of the image. Another popular image augmentation technique, flipping is also carrying out by just changing the locations of pixels (reversing it from left to right).

In conclusion, image processing is an important field that are required to complete image related task.

REFERENCES

Wang, X.; Peng, Y.; Lu, L.; Lu, Z.; Bagheri, M.; Summers, R.M. ChestX-ray: Hospital-Scale chest X-ray database and benchmarks on weakly supervised classification and localization of common thorax diseases. In *Advances in Computer Vision and Pattern Recognition*; Springer: Berlin, Germany, 2019; pp. 369–392.

Allaouzi, I.; Ben Ahmed, M. A Novel approach for multi-label chest X-Ray classification of common thorax diseases. *IEEE Access* **2019**, *7*, 64279–64288.

Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.

Xu, S.; Wu, H.; Bie, R. CXNet-m1: Anomaly detection on chest X-Rays with image-based deep learning. *IEEE Access* **2019**, *7*, 4466–4477.

Ardakani, A.A.; Kanafi, A.R.; Acharya, U.R.; Khadem, N.; Mohammadi, A. Application of deep learning technique to manage COVID-19 in routine clinical practice using CT images: Results of 10 convolutional neural networks. *Comput. Biol. Med.* **2020**, *121*, 103795.

Pasa, F.; Golkov, V.; Pfeiffer, F.; Cremers, D.; Pfeiffer, D. Efficient deep network architectures for fast chest X-Ray tuberculosis screening and visualization. *Sci. Rep.* **2019**, *9*, 6268.