SYDE660 SYSTEMS DESIGN GRADUATE WORKSHOP

FACULTY OF ENGINEERING

DEPARTMENT OF SYSTEM DESIGN ENGINEERING

Covid-19 X-Ray Image Detection Final Report

Instructor: Prof.Jonathan Kofman

Group Number 1:

Zihao Liu (ID: 20855065)
Yifei Huang (ID: 20750564)
Zihe Wang (ID: 20609740)

Date: October 23, 2020

Abstract

The COVID-19 pandemic has caused devastating effect to the globe and posts great threat to public health; test kits fell on shortage and medical institutions are overflowed with patients. To fight the pandemic, it is important to find a method that detects COVID-19 fast and effectively but does not require test kits or increase the workload of medical personnel. This project proposes a deep learning system, Modified Xception, to detect COVID-19 infections automatically from chest X-ray scans. Modified Xception takes in digital chest X-ray scans and classifies according to lung conditions using convolutional neural network. Modified Xception can serve as a prescreen for radiologist; it makes correct diagnoses with high certainty, and reports the uncertain, possibly incorrect diagnoses to radiologists and call for assessments. The prototype was trained with the 13,987 images of three lung conditions (COVDI-19 infection, non-COVID-19 infection, and no infection) and tested on 1,597 images; the testing performance achieved 92% accuracy in detecting COVID-19 infections and 95% accuracy in classifying all three lung conditions. However, the prototype made 99% of its diagnoses with high certainty. This indicates that some of its incorrect diagnoses were made with high certainty, and would not be reported for human assessment. Moreover, it was discovered that the prototype is prone to the effect of Gaussian noise in input images. In conclusion, the prototype proved its potential of serving as a diagnostic prescreen; however, the prototype is not yet ready to be trusted with independent diagnoses and requires further training and adjustment before commercialization.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Motivations

Starting in Wuhan, China, January 2020, the COVID-19 pandemic has left more than 100 countries in lockdown today. COVID-19 is an aerosol transmissible virus and spreads rapidly. It posts great threat on global public health and has infected more than 7,039,000 caused more than 404,000 deaths [29].

The biggest challenge in fighting the COVID-19 pandemic is to ensure effective detection; as the virus is extremely transmissible and leads to overflow in medical institutions, its detection cannot keep up with the increase of new cases, and effective quarantine cannot be ensured. A traditional method of virus detection is antibody serological test. This method usually takes 3 to 6 hours to develop in a medical laboratory. It requires a blood sample from the patient. It is very reliable in COVID-19 detection with accuracy of 0.97, and sensitivity of 0.94 [9].

Another method is the RT-PCR test, or reverse-transcription polymerase chain reaction test. Unlike the antibody test, this method detects the presence of COVID-19 virus, and requires only a nasal swab and a throat swab sample. It can take from 2 to up to 24 hours of development in a medical laboratory. It has an accuracy of 0.91 and sensitivity of 0.79 [11].

One major drawback of the two above detection methods is that they both require test kits, which soon fell in shortage after the initial COVID-19 outbreak. Laboratory development may require extra time for sample transportation. In addition, research indicates that RT-PCR test, which has been used widely in Drive-Thru tests, has increasing false negative rate as time since exposure increases [12].

Another commonly used detection method during outbreak is medical imaging. As COVID-19 infection manifests in lungs, radiologists can assess the CT or X-ray chest scan of patients and determine whether they have COVID-19 infection. This method only requires an X-ray/CT machine and a radiologist, which can be found in most hospitals. The generation and assessment of an X-ray scan usually takes 1 hour total. However, large volume of patients posts great workload on medical professionals, and a research shows the accuracy of radiologist detection of COIVD-19 is 0.82, and the sensitivity is 0.80 [2].

1.2 Design Statement

To effectively battle COVID-19, it is crucial to have a fast, safe and reliable way for detection. However, as of currently, laboratory tests are limited by supplies, and medical image assessment involves having highly trained medical professionals spending hours inspecting the chest X-ray scans of patients. Motivated by this, this project aims to **develop an automated COVID-19 analysis system that will pre-screen patients' chest X-ray scans for radiologists and thereby reducing their workload.**

1.3 Project Goals and Objective

Artificial intelligence has been applied to radiology to detect abnormalities, such as cancer and hemorrhage. This project also takes the approach of enabling automated detection of COVID-19 infection via artificial intelligence, specifically, deep learning algorithms. This project is dedicated to the implementation of a deep learning model that identifies COVID-19 infection in chest X-ray scans.

The model aims to identify COVID-19 (as well as non-COVID-19) infections with high accuracy. In addition, it is desired that the model identifies with high certainty, or confident predictions. In summary, the goal of this project is to create a model that can:

1. Effectively detect COVID-19 infections in chest X-ray scans
2. Effectively distinguish cases with infections other than COVID-19 and no infection
3. Identify all cases with high certainty

This project focuses on the processing of data, but not generation of data. Therefore, the generation of X-ray images, X-ray machines, and how X-ray images are transformed to digital form is out of the scope of this project. In addition, the project focuses on back-end development (the model and algorithms); front-end development, such as graphical user interfaces, is out of the scope of this project.

1.4 Expected Impact

With high accuracy, the system can serve as a as a substitute to RT-PCR and antibody tests when supplies are limited; in addition, it can also serve as pre-screening for RT-PCR and antibody test. The system can better prepare medical institutions for sudden eruption of epidemic, and effectively reduce the stress on medical workers. Moreover, when dealing with a global pandemic, the system can help medical institutions issue test kits and other resources more economically and minimize the effect of shortage.

It is also possible to use the system outside of the pandemic. Since the system is a deep learning model, it can also be tweaked towards identifying infections other than COVID-19 with more training. When the system can be implemented at most hospitals and used to identify common diseases, the efficiency of diagnostic radiology will be increased significantly. It will also lower the workload of radiologists and allow them to focus on the cases after prescreening.

However, systems as such are limited by the amount of training data. It is very difficult to compile all available data to train a model, since the data belongs to different institutions. Also, achieving high accuracy on available data does not mean the system can perform consistently well when implemented–not to mention there is little room for error when it comes to diagnosis. It is questionable whether patients are willing to leave their diagnoses to a machine. In addition, medical imaging exposes patients to radiation, which can potentially harm the human body. It is also questionable whether patients are willing to do X-ray scans first when test kits are available.

In summary, the system should be viewed as an emergency measure for sudden epidemic. It should be used to alleviate the overflow of patients, overwork of medical professionals, and shortage of test kits when epidemic erupts. It can serve as a prescreen for radiologist, and the radiologist only need to look at the cases that the system fails to provide a confident diagnosis to.

2 Requirements and Specifications

2.1 Design Requirements

The overall function of the system is to prescreen patients' chest X-ray scans for radiologists with high accuracy and certainty. The following sections elaborate the functional and user requirements for the system.

2.2 Functional requirements

A patient's chest X-ray can show one of the three lung conditions: COVID-19 infection (COVID-19, Class 0), non-COVID-19 infection (Pneumonia, Class 1), and no infection (Normal, Class 2). The basic function of the system is analyzing patients' chest X-ray scans and reporting the probability of the patient belonging to each of the three conditions, or classes. In addition, the system makes a diagnosis based on the class that has the highest probability. For example, for a patient has COVID-19 infection, the system output can look like: (0.85, 0.12, 0.03), COVID-19. The three numbers are the probabilities corresponding to class 0, 1, and 2, and the reported class is COVID-19.

It is possible for the system to make uncertain predictions, that is, when the highest class probability is lower than 0.75. This indicates that the system is less than 75% sure of the diagnosis; such diagnoses are not considered concrete and assessment by radiologist is required.

In summary, the system functions by determining the probabilities of each type of lung conditions and report the most probable condition for every input chest X-ray scan. If the most probable class has a probability below 0.75, the system reports the case number and all class probabilities, and calls for radiologist assessment. The flow chart in Figure 1 summarizes how the system functions as a prescreen.

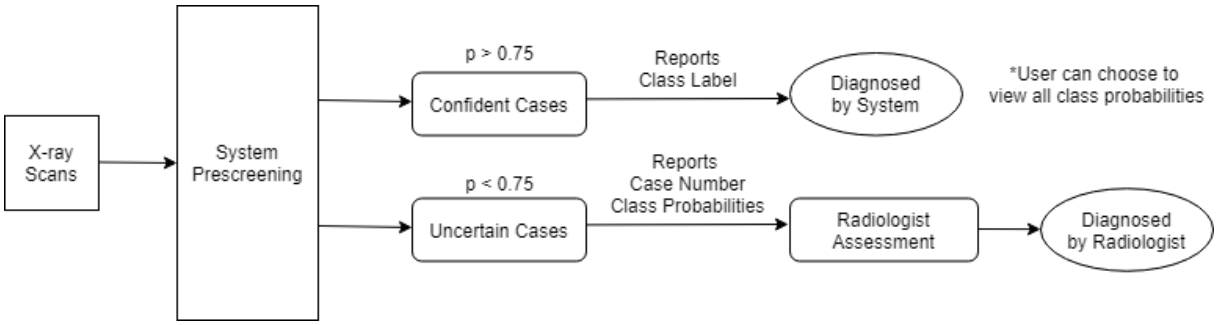


Figure 1: System Function Flow Chart

As the flow chart shows, the system needs to make reliable, independent diagnoses and minimize the number of uncertain cases to reduce radiologists' workload. In addition, since cases with confident diagnoses will not be assessed by radiologist, it is important to ensure that all diagnoses that the system makes with high confidence are correct.

2.3 User Requirements

The system should be easy to operate and does not require programming background, so medical personnel can use it without advanced training. The final product should also be able to give clear instructions on how to interpret the outputs. The number of inputs required from users should be limited to avoid possible mistakes or confusion; users do not need to make adjustment to any algorithms or hyperparameters. In addition, users should have the option to look at some or all system outputs, including class probabilities, predicted class labels, and list of uncertain cases.

The system should be able to adapt to most computer settings and operating environment. The system should be able to run under most operating systems (such as Windows 10 and MacOS 10.15). The system should not be too computationally expensive and should be able to run on regular (Intel and AMD) CPUs and be able to diagnose a single X-ray scan (one input image) in five seconds.

2.4 Design Solution and Specifications

This project aims to develop a deep learning model that satisfies the above functional and user requirements. The deep learning model is in the form of a computer program, and takes in digital chest X-ray scans and outputs diagnoses for each scan. In addition, it calls for radiologists' assessment for cases diagnosed with uncertainty. The following sections explain the design specifications, engineering parameters, and objectives set for the system to meet the requirements.

2.5 Engineering Parameters and Objectives

The system aims to achieve high accuracy in detecting COVID-19 infection, high accuracy in distinguishing COVID-19 infections, other pneumonia infections, and healthy conditions. Moreover, the system aims to make diagnosis with high certainty. These objectives can be quantified as below:

1. **90% COVID-19 accuracy**
2. **90% overall accuracy** (COVID-19, Pneumonia, and Normal class)
3. Less than 25% of diagnoses are uncertain (highest class probability $p < 0.75$), or **an ambiguity rate smaller than 25%**

These values can also be found in the QFD chart in the appendix. The target COVID-19 accuracy, or the system's sensitivity to COVID-19 is set to 0.900; this is higher than the sensitivity of RT-PCR test (0.794) and radiologist assessment (0.802). Low COVID-19 sensitivity indicates that more false negative cases can occur, which can undermine the epidemic control of COVID-19. Therefore, it is crucial that the system is highly sensitive to COVID-19.

During the COVID-19 epidemic, infections other than COVID-19 can also occur. It is important not only to isolate COVID-19 cases from other cases, but also to isolate other infections from healthy cases, as other infections may also be transmissible. Therefore, the system also seeks to perform well on the Pneumonia and Normal class, and achieve an overall accuracy of 90%.

When a diagnosis is made with less than 75% confidence, that is, the highest class probability is less than 0.75, the diagnosis is considered uncertain and requires radiologist assessment. The portion of cases that are uncertain is defined as the ambiguity rate. Since the purpose of the system is to prescreen chest X-ray scans for radiologists and hence reduce their workload, it is desired to minimize the need of radiologist assessment. The ambiguity rate of the system should be limited to under 25%; the remaining 75% of cases are diagnosed correctly with high certainty. This way, radiologists only need to assess less than one-fourth of the cases.

Another important engineering parameter is the specificity, or true negative rate. This parameter indicates how many false positive cases may occur. It is important to keep the specificity to COVID-19 high, since putting a patient without COVID-19 infection into COVID-19 quarantine will likely give that patient COVID-19 infection. Among the existing diagnosing methods, RT-PCR has the highest specificity (1.0) but also a low sensitivity (0.794). Radiologist has a lower specificity (0.837), whereas serological test has a better one (0.987). The goal specificity of the system is also a high 0.900.

Other engineering parameters in the QFD chart are calculated based on the above ones. Many of the goal engineering parameters are set such that the system's performance is better than RT-PCR test and radiologist assessment, but not as good as serological test. This is because the system is designed to work with radiologists; the cases missed by the system will be reported to radiologist for assessment. The combined performance of the system and radiologist should be able to approach that of serological test. In addition, pursuing higher performance of the system will be time-consuming and requires massive amount of training data, and will not be practical for the scope of the project. Nevertheless, it is extremely important to make sure that the system only makes correct diagnosis with high certainty, incorrect cases are all uncertain– otherwise radiologist cannot compensate the system's mistake.

2.6 System Input and Output

The system input is digital photos of chest X-ray scans (PNG and JPEG files); users only need to input a path for the image files. The size and color of input images may vary, and will be standardized during systems' data preprocessing. However, to ensure the system's stable performance, noisy images should be avoided.

Regarding output, the system reports a diagnosis of the most likely class (the classes with the highest probability) to which the input image belongs. The system should also output a list of uncertain cases, either their case numbers or filenames, and call for radiologist assessment. For these uncertain cases, the system will report all three class probabilities. In addition, the user can choose to view all class probabilities for any or all cases.

3 Engineering analysis

Prescreening chest scans is essentially an image classification problem. Therefore, convolutional neural network (CNN) was chosen as the basic framework for the system, as it has proven powerful in various image recognition tasks, especially in multi-class classification problems. Moreover, in case that CNN or multi-class classification does not work well, binary classification with decision trees was also considered as an alternative.

Several models were developed and trained with the *COVIDx* dataset, put together by Linda Wang and Alexander Wong, which contains 15,476 labeled chest X-ray scans from COVID-19 and non-COVID-19 patients[28]. Transfer learning of some pre-trained CNN models was also performed. Among all models, a modified version of **Google's Xception** achieved the best testing performance.

This section will briefly go over some features of all models developed in the design process and how they were optimized. The final structure used in the system, **modified Xception**, will be explained in detail. Details and performances of other models can be found in Section 4.

3.1 System Input and Preprocessing

As previously mentioned, the input to the system is chest X-ray images. Although the size of chest scan may vary based on the X-ray machine or scanner, CNN demands data input with uniform dimension. Therefore, all input images were resized to the resolution of 480*480. In addition, to eliminate the possible tint variation caused by different scanners and cameras, the images were converted to grayscale. Loading and preprocessing were done by a Python package, *OpenCV*.

3.2 Basic CNN Structure

CNN consists of three types of layers: convolution layers, which apply filters and obtain features from input data, pooling layers, which reduces the size of input data by down-sampling, and fully connected layers, which learn all valuable features produced by previous layers. The system follows the common alternating convolution and pooling layer structure, and places fully connected layers towards the end.

There is no direct rule to determine the number of layers in a CNN; for this project, the number of layers were determined based on trial and error. Layers were added according to testing performance, until overfitting occurred. Regularization components were also introduced to reduce overfitting. In addition, the hpyerparameters associated with each layer and the training process were also determined through trial and error.

3.2.1 Convolutional Layers

There are many parameters associated with the convolutional layers. For input image I and filter (kernel) K, the two-dimensional convolution operation can be defined as follows [17].

$$(I * K)(i, j) = \sum_m \sum_n K(m, n)X(i - m, j - n) \tag{1}$$

where * is the discrete convolution operation. The square matrix K slides over the input image in steps specified by the stride value (S). A greater stride lets filters move faster, but may cause shrinkage of the output feature map and hence loss of information. Filter size (F) also affects the information captured; larger filters have higher speed but also output with smaller dimensions. For this project, smaller filter size and stride were preferred to perserve more information. To allow filters to slide over every pixel, zero-padding (P) was placed along the edges of input image. Below is a summary of the hyperparameters needed for a convolutional layer and their example values:

$$Filter\ Size\ F = 5$$
$$Stride\ Value\ S = 1$$
$$Amount\ of\ Zero - Padding\ P = 2$$
$$Number\ of\ Filters\ K = 64, 128, 256...$$

where number of fileters increases in power of 2 as the number of layers increases. The above values were found to be optimal and were used throughout all convolution layers in the design process, except for when doing transfer learning, where the values were set by developers. With an input dimension of $W_1 \times H_1 \times D_1$, the output dimension of a convolutional layer $W_2 \times H_2 \times D_2$ can be calculated as:

$$W_2 = (W_1 - F + 2P)/S + 1 \tag{2}$$
$$H_2 = (H_1 - F + 2P)/S + 1 \tag{3}$$
$$D_2 = K \tag{4}$$

where each $W_2 \times H_2$ layer is called a feature map, and there are K feature maps in total. With parameter sharing, a convolutional layer introduces $F \cdot F \cdot D_1$ weights per filter, thus $(F \cdot F \cdot D_1) \cdot K$ weights and K biases in total. Druing the design process, these parameters were initialized via random initialization, sampling from a uniform distribution of $U = (-\sqrt{K}, \sqrt{K})$, and optimized as the system learned.

3.2.2 Activation Function

After passing through a convolutional layer, data was fed to a nonlinear activation function. Activation function adds non-linearity to the data and allows for more complex learning. The most commonly used nonlinear activation function in neural networks is the ReLu function, shown in equation 5 below, which keeps all values greater than or equal to 0 as they are, and sets all values less than 0 directly to 0.

$$f(x) = max(0, x) \tag{5}$$

All CNN models developed in the design process used ReLu activations due to its cheaper operations and faster convergence in stochastic gradient descent. One disadvantage of ReLu units is that they are fragile and can ”die” during training; for example, a large gradient passing ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any data point again [24]. However, this situation did not occur during the design process.

If large portion of the network dies during training, alternatives such as Leaky ReLu (shown in equation 6, where α is a small constant) can be used.

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x) \tag{6}$$

3.2.3 Pooling Layers

Pooling layers are introduced to CNN to reduce computational costs via data reduction for each feature map. Pooling also requires a filter to be applied to feature maps, just like convolutions. During the design process, most filters were set to the size of 2x2 pixels with stride of 2 pixels, such that the pooling layer would reduce the dimension of its input by exactly half.

Two commonly used pooling, max pooling and average pooling, were used on the models developed. Max pooling keeps the larges value of an selected area, whereas average pooling takes the average of all values of that area.

3.2.4 Fully Connected Layers and System Output

Fully connected layers were placed at the end of CNN models to learn the results produced by previous layers and convert them to values that predict the best label to describe each input image. Since the system aims to predict three classes, the fully connected layers are responsible for converting the final output into a 3×1 vector. Different numbers of fully connected layers were tested through the design process, and it was found that 3 to 4 layers resulted the best performances.

The last fully connected layer is followed by a softmax classifier (Equation 7), which normalized the model output into values between 0 and 1.

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{7}$$

In a probabilistic view, the function can be written as:

$$P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \tag{8}$$

The output values can be interpreted as the (normalized) probability assigned to the correct label y_i given the image x_i and parameterized by weight W [24]. In addition, the models report the most probable class for each image.

3.3 Existing CNN Architectures

In addition to constructing a new model, transfer learning was done with some existing CNN architectures that have been proven effective in image classification problems. The models used in transfer learning also had the same components mentioned in the last section, but their architectures (such as number of different layers) and hyperparameters were set by the developer. Moreover, some of these models were already trained, so their weight and bias were initialized differently. This section lists the CNN architectures used in transfer learning.

3.3.1 Residual Convolutional Network

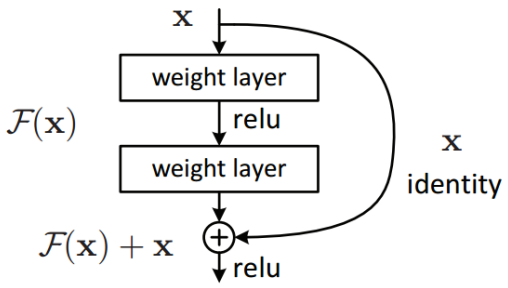


Figure 2: ResNet-50 structure [10]

He et al. proposed a deep neural network structure called residual network in 2015 [10]. Study shows that the residual network is easier to optimize and can increase the accuracy by adding depth. The core of this method is to solve the side effect (degradation problem) caused by increasing the depth, so that the network performance can be improved by simply increasing the network depth. Fig. 2 shows the basic idea of resdual block, which feed forward neural networks with "shortcut connection", and this structure make fitting easier without gradient vanishing.

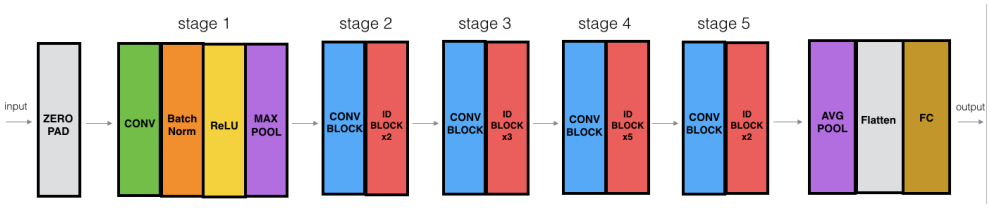


Figure 3: ResNet-50 structure [10]

Regarding the overall structure of the ResNet-50, the core idea is to stack the convolutional block and the identify block repeatedly (Fig. 3). The ResNet-50 model consists of 5 stages, and each stage has a convolution and identity block. Both convolution block and identity block has 3 convolution layers following a BatchNorm and a ReLu activation function, but convolution block has one more convolution layers on the shortcut. ResNet-50 has more than 25 million trainable parameters [10].

3.3.2 Densely Connected Convolutional Networks

Based on the shortcut (skip layer) technology proposed in ResNet [10], authors of DenseNet, Gao et al., [6] design a new connection mode. In order to maximize the information flow between all layers in the network, the author connects all layers in the network in pairs, so that each layer in the network accepts the features of all the layers before it as input. Since there are a large number of dense connections in the network, the author calls this network structure DenseNet. The schematic diagram of its structure is shown on Fig.

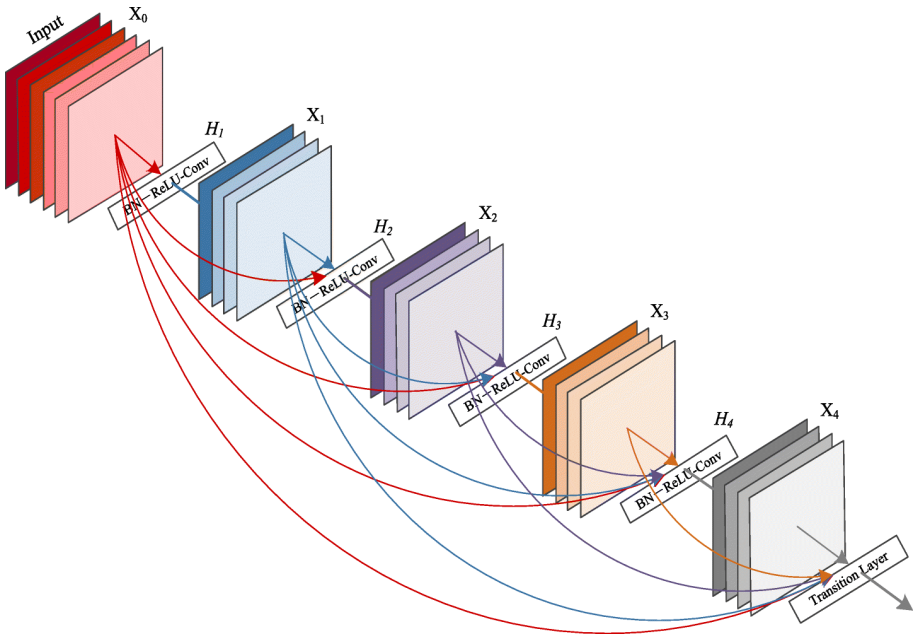


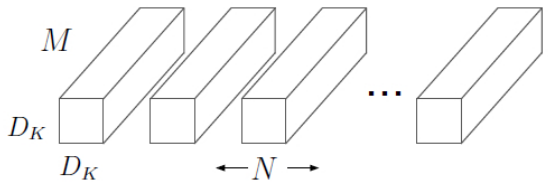
Figure 4: DenseNet structure [6]

This neural network structure mainly has the following two characteristics, firstly, reducing the problem of gradient vanishing during training stage. Fig.2 shows that each layer will receive the gradient information of all subsequent layers during the back propagation, therefore, as the network depth increases, the gradient near the input layer will not become smaller. Secondly, a large number of features are reused, a large number of features can be generated using a small number of convolution kernels, therefore the size of the final model is relatively small. Based on the above principles, authors designed DesNet-121, DesNet-169, DesNet-201 and DesNet-161 for ImageNet object recognition tasks. The organization of DenseNet is similar to ResNet, but the original ResNet identity block is replaced, and the down sampling process is slightly different.

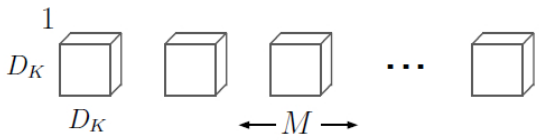
In general, DenseNet points out the concept of shared features and interconnection between layers, which greatly reduces the problem of gradient vanishing and difficulty in optimization in the training process. But the drawback is that the network cannot be designed too deep, because as the number of DenseNet network layers increases, the feature dimension of the model will increase linearly, so that the amount of calculation and memory overhead during the training process will increase explosively. DenseNet-201 has about 20 million trainable parameters [6].

3.3.3 Mobile Net

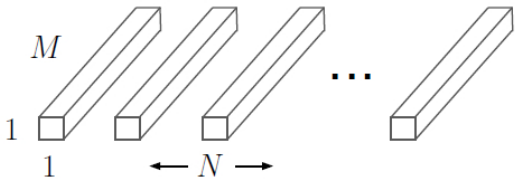
Mobile Net is a network architecture released by Google in 2017 [15]. As its name suggests, it is a convolution neural network with smaller model size, less trainable parameters and small calculations, and suitable for mobile devices. It aims to make full use of limited computing resources and maximize the accuracy of the model to meet various application cases under limited resources. It is one of the commonly used models deployed to the edge. The main innovation of mobileNet is to use *depthwise* separable convolution instead of ordinary convolution, and use width multiply to reduce the amount of parameters, which can trade for better while sacrificing little precision to data throughput.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 5: Standard Convolution Filters and Depthwise Convolution Filters [15]

The main innovation of Mobile Net is to replace ordinary convolution filters with depthwise convolution filters, and the core idea of depthwise convolution filter is to split ordinary convolution into two parts, Depthwise and Pointwise (Fig.36). Among them, Depthwise corresponds to convolution in groups, and Pointwise corresponds to concatenated information. However, Mobile Net V1 is an extreme performance of grouped convolution, that is, each input channel is calculated as a group separately.

Authors also mentioned that the use of depthwise separable convolutions can reduce the amount of calculation by 8-9 times. Mobile Net only has 88 layers and 4 million trainable parameters [15].

3.3.4 Xception Net

Xception was proposed by Google, Francois Chollet [30]. The author interprets the Inception module in the convolution neural network as an intermediate step between conventional convolution and deep separable convolution operations (deep convolution and then point-wise convolution). Therefore, the separable convolution in the depth direction can be understood as the Inception module with the largest number of towers. This discovery led author to propose a novel deep convolution neural network architecture inspired by Inception, where the Inception module has been replaced by deep separable convolution.

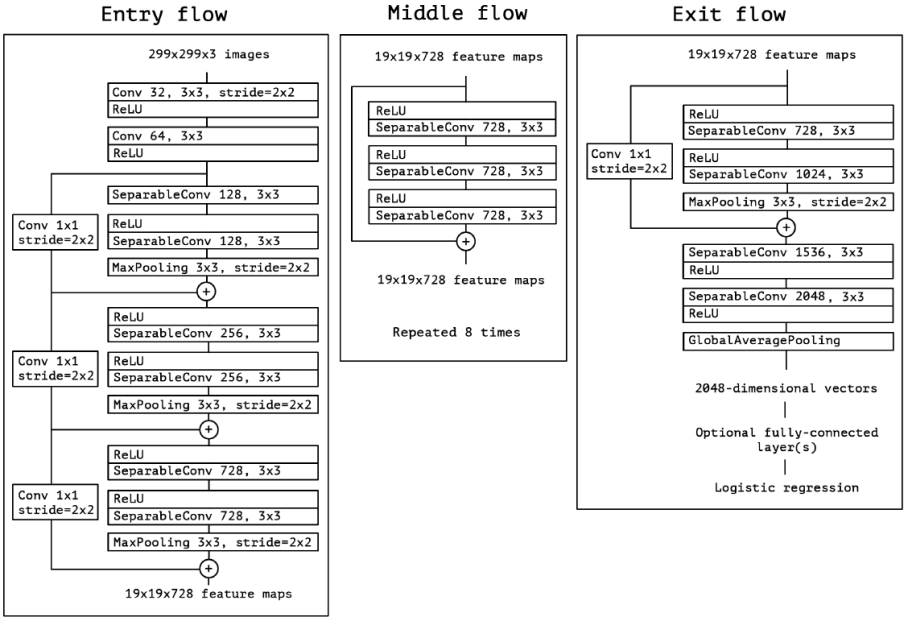


Figure 6: Xception architecture [30]

Fig. 6 shows the Xception architecture, which is slightly better than Inception V3 on the ImageNet dataset (Inception V3 is designed for this dataset), and is slightly better than Inception V3 on a larger image classification dataset containing 350 million images and 17,000 categories. Xception performance is significantly better than Inception V3. Since the Xception architecture has the same number of parameters as Inception V3, the performance improvement is not due to the increase in the number of parameters, but due to the more efficient use of model parameters. The author proposed a convolution neural network architecture based entirely on deeply separable convolutional layers. In fact, author made the following assumption: the mapping of cross-channel correlation and spatial correlation in the feature map of the convolution neural network can be completely decoupled. Because this assumption is a stronger version of the Inception architecture assumption, the author named the proposed architecture Xception, which stands for ”Extreme Inception”.

Fig. 6 shows a complete description of the network specifications. The Xception architecture has 36 convolutional layers, which form the basis of the feature extraction of the network. The

36 convolutional layers are constructed into 14 modules, all of which have linear residual connections around them except for the first and last modules. In general, the Xception architecture is a linear stack of deeply separable convolutional layers with residual connections, which makes the architecture easy to define and modify. Xception has 22 million trainable parameter [30].

Slight modifications, such as adding convolution or fully connected layers, were also made on these models to improve performance.

3.4 Binary Classification and Random Forest

In the case of multi-class classification does not function well, the project also considered binary classification models. Although it is intuitive to directly train a convolutional neural network (CNN) model to conduct the binary classification task, such an approach requires long training time and expensive equipment. Alternatively, one can leverage pre-trained CNNs, i.e. CNNs trained on ImageNet datasets, to extract deep features from the Chest X-ray images, followed by training classical machine learning algorithms such as ”Random Forest” on those deep features. This approach consumes much less training time and has a much lower requirement on computational equipment. Fig. 6. provides an overview of the method.

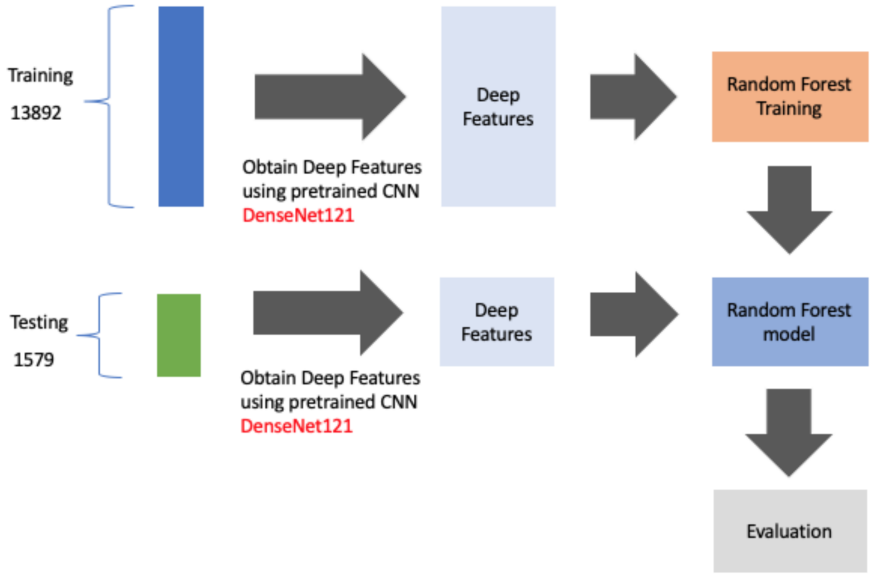


Figure 7: Binary Approach

Firstly, deep features of the training Chest X-ray images were obtained via DenseNet121, which introduced in section 3.4.2. Secondly, these features with the class labels were then used for training a Random Forest model. Thirdly, deep features of the testing Chest X-ray images were obtained via DenseNet121. Fourthly, the deep features were then inputted to the Random Forest model to predict if there are COVID-19. Fifthly, the evaluation was then conducted on the predictions of the Random Forest model. The random forest is a model that consists of many

decision trees. We can consider decision trees as a series of yes or no questions, which are asked to our data, and the decisions will lead to a predicted class. The random forest algorithm uses bagging and feature randomness when building each individual tree, and it tries to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. [32] It has the crucial feature of preventing over-fitting and it is easy to interpret. And they do not require feature scaling, categorical feature encoding, and does not require many parameters. In the training process, each single tree in a forest learns the feature from a data point, the samples are drawn using bootstrapping. The idea is training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data. Overall, the entire forest will have lower variance but not at the cost of increasing the bias.And in the training process, the final predictions are made by averaging the predictions of each single tree.[33]

3.5 Optimization and Regularization

Optimization of the models was done through updating the weights and biases to minimize the loss function:

$$L = \underbrace{\frac{1}{N}\sum_{i} L_i}_{\text{data loss}} + \underbrace{\frac{1}{2}\lambda \sum_{k}\sum_{l} W_{k,l}^2}_{\text{regularization loss}} \tag{9}$$

Where L_i is the cross entropy loss:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{10}$$

The choice of optimizer plays an extremely important role in deep learning model training stage. This project uses the Adam optimizer proposed by Kingman and Ba [13], which is a variation of Stochastic Gradient Descent (SGD). In practice Adam is currently recommended as the default algorithm to use; it has the advantage of both Adagrad and RMSprop and can handle sparse gradients and noisy problems. The below figure summarizes the algorithm of Adam optimizer.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 8: Adam Algorithm [13]

Kingma and Ba mention that in their case, good default settings for the hyperparameters are stepsize/learning rate $\alpha = 0.001$, decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In the design process, these values were used except for α, which was adjusted to 0.0005.

The SGD objective function $f(\theta)$ is the loss function $L(\theta)$ (the W in equation 9 is replaced by θ) and the gradient g is:

$$g \leftarrow \frac{1}{N} \frac{d}{d\theta} (\Sigma L(f(x^{(i)}; \theta)), y^{(i)}) \tag{11}$$

As shown in equation 9, L_2 regularization was used to avoid overfitting. L_2 regularization penalized the squared magnitude of parameters directly, and can help the network avoid overly large weight vector and prefer diffuse weight vectors. Hyperparameter λ determines the regularization strength. However, if λ is too large and regularization is too strong, underfitting will occur. λ was be first set to 0 for the system and adjusted as overfitting began to occur. It was increased gradually but limited to a value equal to or smaller than α, the learning rate, so the models would not learn to slowly.

Random oversampling was used to minimize the effect of skewness of dataset. Among the 13,897 images in the training dataset, only 473, or 3.4% belong to the COVID-19 lass. Training with the original dataset may cause the model to be insensitive to COVID-19 infection. Therefore, random oversampling was used while loading training images, such that some COVID-19 images would be selected repeatedly, and each of the three classes would make up roughly 33% of the training data.

Data augmentation was also used for regularization, as the training dataset was relatively small. Data augmentation used in this this project consists of two types of transformations, horizontal flip and rotation (with an angle of 20 degrees). These transformations were applied when the training data was loaded, at a probability of 50% each, to all three classes of images.

3.6 Hyperparameter Sensitivity Analysis

For this project, the sensitivity analysis aims to understand how important hyperparameters influence classification performance before the system is fully optimized. During the design process, the sensitivity analysis took a one-at-a-time (OAT) approach by analyzing the effect of one parameter on loss and keeping others fix[3]. After learnable parameters were initialized, different hyperparameters were tested with a few epochs of training.

A sensitivity analysis was done to evaluate how the number of convolutional layers of a CNN affect its performance. The analysis was done on a simple CNN model, with 2, 3, 4 and 5 convolutoinal layers. The filter size, stride, and zero-padding of the each convolutional layer were kept the same ($F = 5, S = 1, P = 2$), and number of filters incerased in power of two as layer increased ($K = 16, 32, 64, 128, 256$). Hyperparameters other than number of convolutional layers were kept the same. Max pooling with filter size 2 was used after each layer of convolution and three fully connected layers were placed at the end of the network.

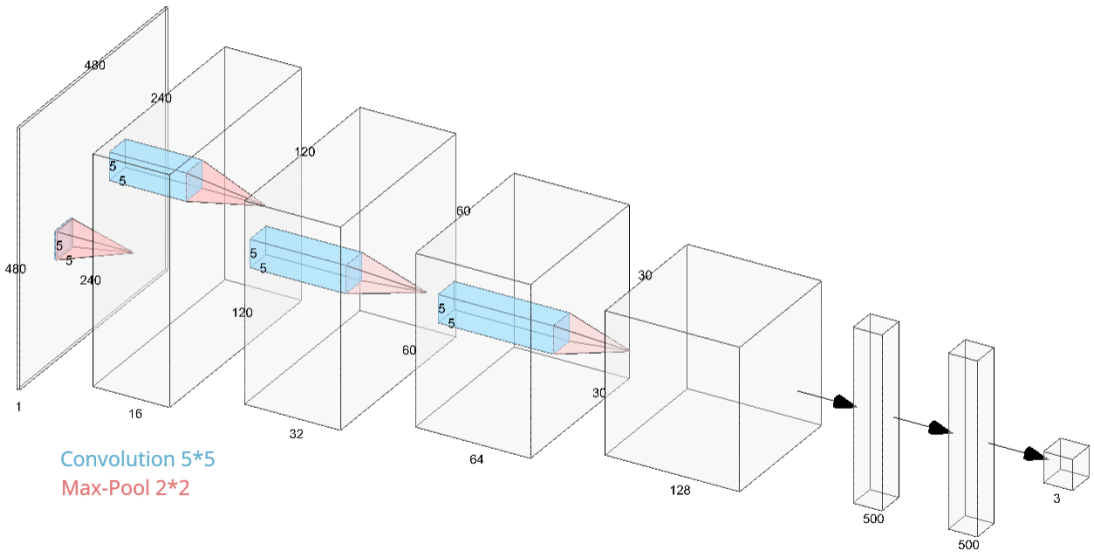


Figure 9: 4-Layer CNN Model Used in Sensitivity Analysis of Number of Convolutional Layers

Each model was trained over 13,897 images of the training set; oversampling was applied to ensure that the the number of images belonging to each class (COVID-19, Pneumonia, Normal) was roughly 33%. Data augmentation mentioned in the previous section was also applied to the training images. Learning rate was set to 0.001 and L_2 regularization strength was 0.0005. After each epoch of training, the models were tested on a set of testing images.

The testing performances (epoch average loss, overall accuracy, and COVID-19 accuracy) of the 2, 3, 4, and 5-layer CNN model are plotted in Figure 10.

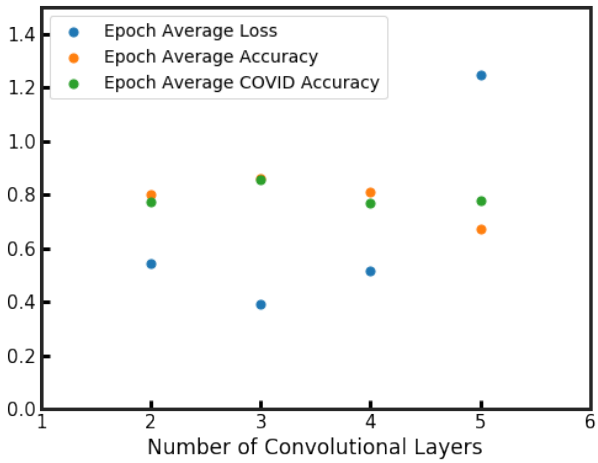


Figure 10: Epoch Average Loss, Overall Accuracy, and COVID-19 Accuracy of CNN Models with Different Number of Convolutional Layers

The results in Figure 9 show that from 2 to 3 convolutional layers, the loss decreased and both overall and COVID-19 accuracy increased. But from 3 to 5 layers, the loss increased again and overall accuracy decreased. This could be an indication of overfitting. Especially from 4 to 5 layers, the COVID-19 accuracy increased but overall accuracy decreased, and loss increased drastically. This is an indication that the 5-layer CNN model overfitted on COVID-19 images.

In conclusion, increasing the number of convolutional layers, or model complexity, can improve model performance, but will make the model more susceptible to overfitting. It is important to begin with simpler models and increase complexity with caution. In the case where a more complex model is required, regularizations should be applied to avoid overfitting.

As the design progressed, a simple CNN model with 4 convolutional layers was developed, shown in the below figure. Using this model, a sensitivity analysis on learning rate was performed.

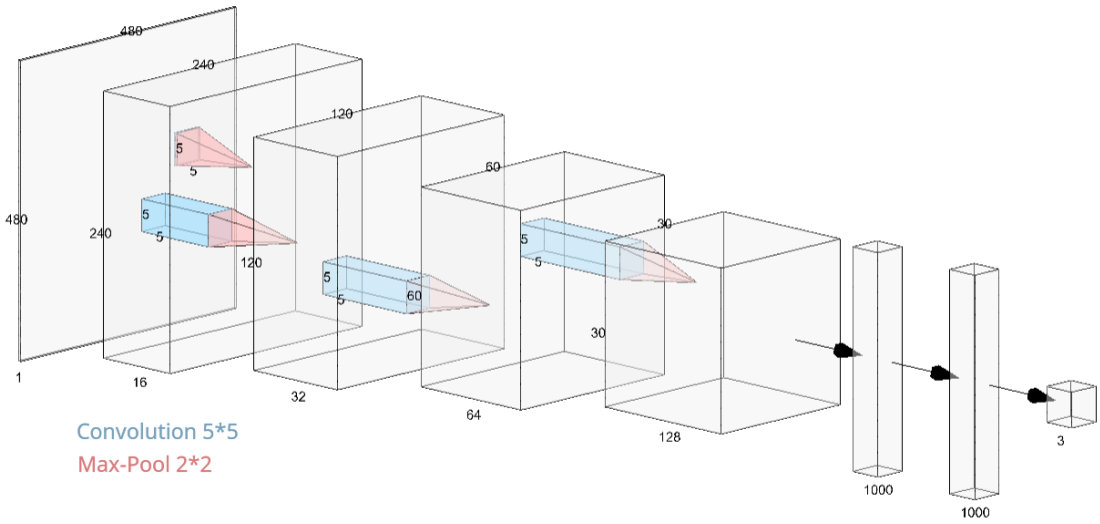


Figure 11: 4-Layer Simple CNN Model

Learning rate of $0.0005, 0.001$, and 0.01 were tested. The model was trained with 13,897 images for 30 epochs, same as in the sensitivity analysis of number of convolutional layers, at the three different learning rates. The model was tested on the test set after each epoch. The training and testing results at each learning rate are plotted in the below figures.

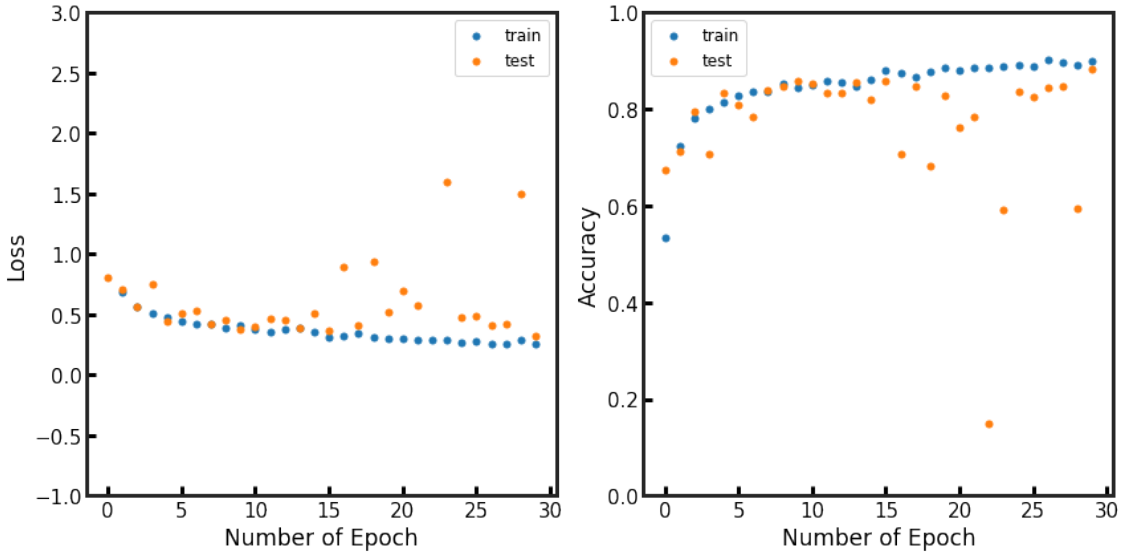


Figure 12: Loss and Accuracy of Learning Rate 0.0005

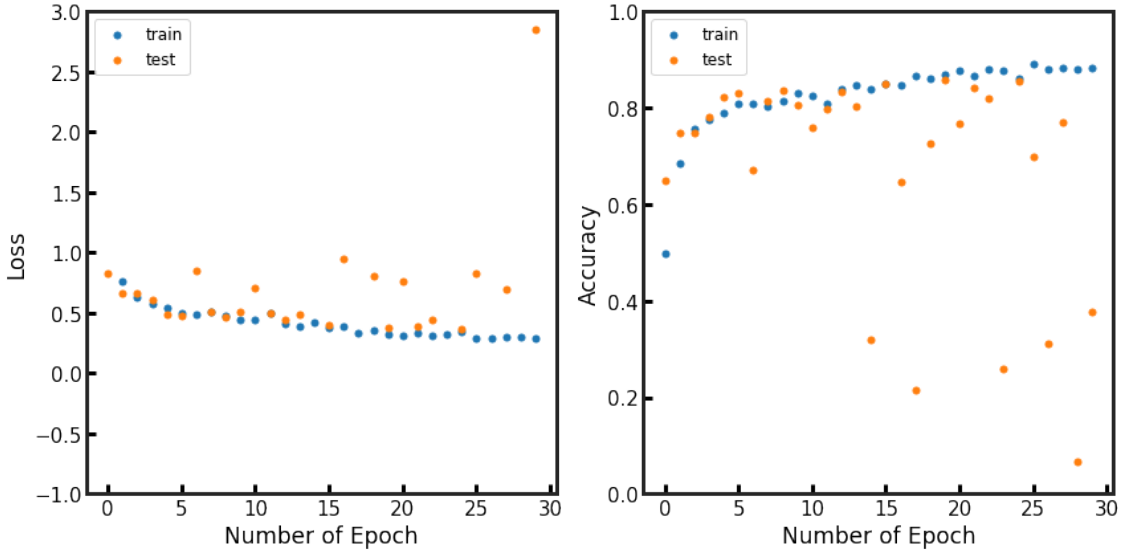


Figure 13: Loss and Accuracy of Learning Rate 0.0010

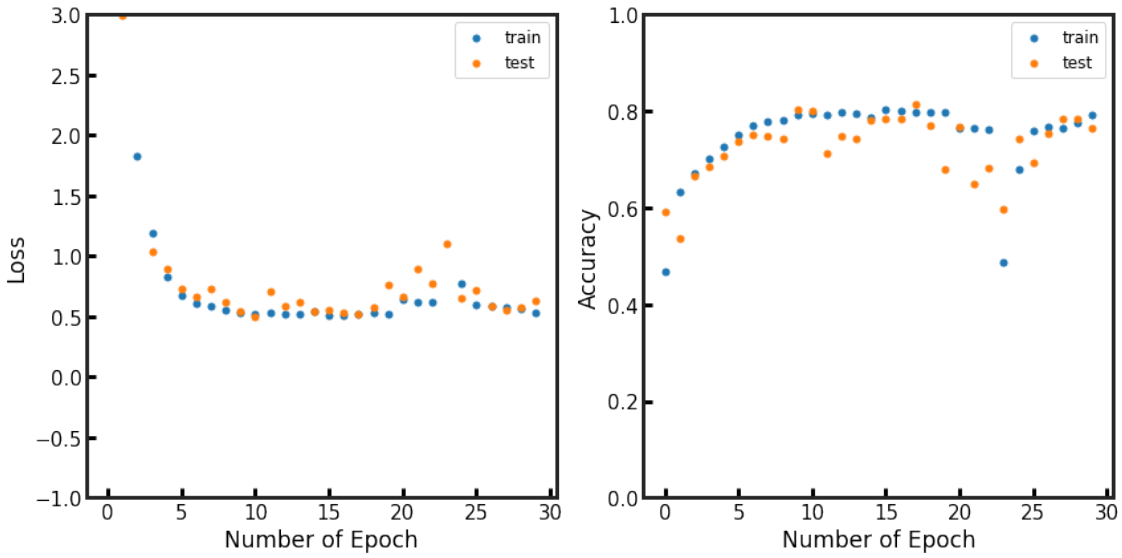


Figure 14: Loss and Accuracy of Learning Rate 0.0100

The results show that larger learning rate led to greater fluctuations in of both testing and training performances. In addition, loss increased slightly while accuracy decreased slightly when larger learning rate was used. The results indicate that larger learning rate may lead to poor convergence in training. When a large learning rate such as 0.0100 was used, the optimizer took large steps and could go beyond a local optimum again and again. This is indicated by the oscillations in loss and accuracy shown in Figure 14.

However, despite the better training convergence, the testing loss and accuracy of some epochs also fluctuate largely with learning rate 0.0010 and 0.0005. This could be because the training had not yet converged, as the model would take more epochs to train at smaller training step. Based on the sensitivity analysis on learning rate, the learning rate of 0.0005 was used for subsequent training's because it produced the smoothest training curves. More epochs (45-85) were used in training to achieve better convergence.

3.7 Final Design Solution

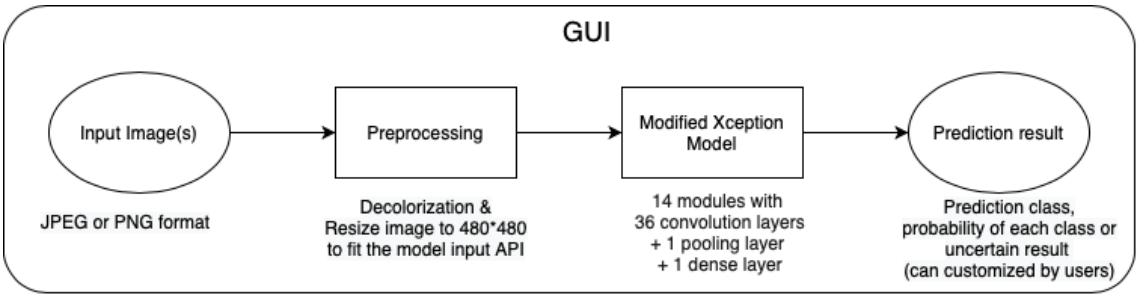


Figure 15: Final system workflow

Figure 15 shows the workflow of the final design of the system. The system input is digital X-ray scans in the format of JPEG or PNG. The system outputs are class prediction (diagnosis), probability of each class, and uncertain cases. Users can customize the outputs by choosing to view some or all of them. In addition, the ready-to-use system has a graphical user interface that wraps around the algorithms, such that users does not need to look into the back-end.

The system includes a preprocessing segment that resize the input images to uniform dimension and convert them to grayscale. The preprocessed images are fed into a deep learning model, the Modified Xception Model, which classifies the images and make diagnoses. The architecture of Xception is shown in Figure 16, and it was modified by adding a global average polling layer and a fully connected layer, as Figure 17 shows.

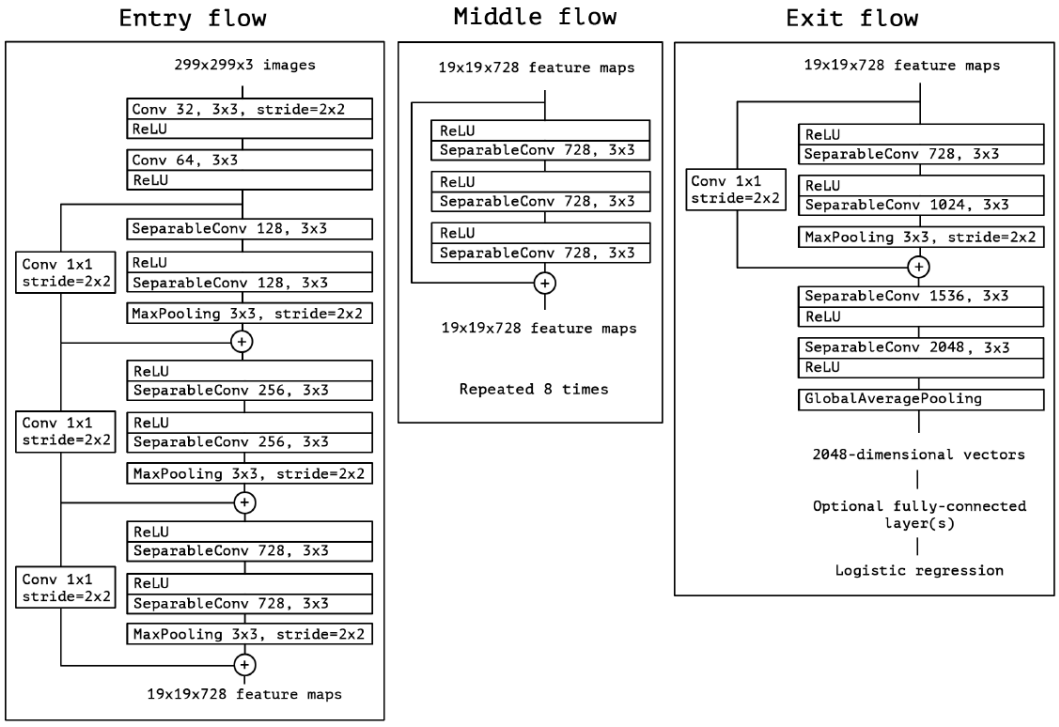


Figure 16: Xception architecture [30]

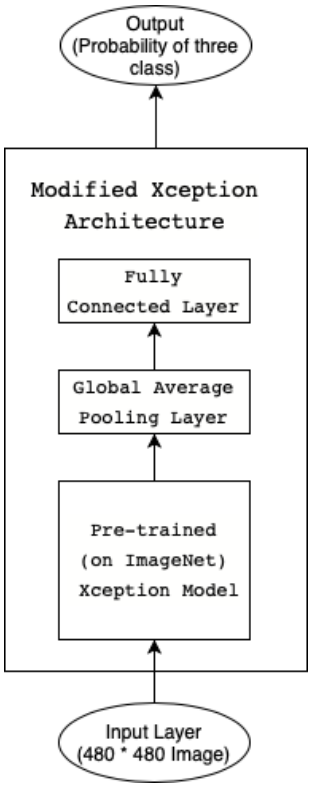


Figure 17: Modified Xception Architecture

The target users of the final design are medical personnel and hence the final design needs to be adjusted to fit their needs. With the global pandemic continuing to soar worldwide, it is also crucial to devise a plan to update the model periodically to keep up with the latest development of the disease.

It is the project's goal for the trained model to be readily accessible for the medical personnel. In addition to marketing the system as a stand-alone computer program, another way to do that is to embed the trained model as an add-on to the medical imaging programs available. Once an X-ray scan is done for the patient, the doctor will receive not only the medical image but also a summary of the result from the trained model. The summary report will include the probability of the patient having each of the lung conditions as well as the most likely condition that the patient has.

As the COVID-19 cases is still soaring worldwide, available X-ray images for COVID-19 will increase. The developers need to update the parameters or refine the model architecture slightly as new data becomes available. Changes to the model can be released periodically as a patch to the add-on of the medical imaging programs. The prototype created in this project had only been trained on a small dataset and is not ready to be commercialized. Although the system is designed to share workload with the doctors, without adequate training, the system can only be used as a reference for the doctors, but not work independently. The final design should be more sophisticated, trained adequately, and updated periodically.

4 Design Progression

The model architecture and algorithms of the system are changing as the design progressed. As mentioned in Section 3, multiple models were considered for the design problem: a relatively simple CNN model, existing CNN models that may have been pre-trained, and ensemble learning with the random forest. The first two models were multi-class classifiers which could classify 3 classes, while the third model was a binary classifier. Since training a model could be time-consuming, these models were developed in parallel. This section illustrates how these models evolved and the technical challenges encountered, also compares the results of the models.

4.1 Evolution of Design

This section explains how each model mentioned above developed, how they were trained, and why a model was or was not chosen as the final design. This section will also briefly evaluate the performance of each model at each design stage. The evaluation of final design can be found in Section 5.

Data preprocessing was identical for all models: resizing to 480*480 and converting to grayscale. The models were trained on the training dataset (13,987 images) and evaluated on the testing dataset (1,579 images).

4.1.1 Benchmark Study

A benchmark study was done before diving into complex models to determine a baseline objective for the project. Logistic regression is widely used in multi-class classification problems and was chosen for the benchmark study. In addition, principal component analysis (PCA) was used to filter less useful features and shorten computation time.

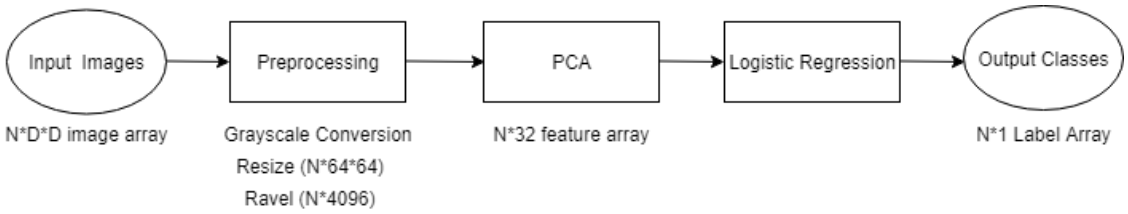


Figure 18: PCA and Logistic Regression Classifer Workflow

As shown in Figure 18, the training images were first converted to grayscale and resized to $64 * 64$. No data augmentation or oversampling was used. The resized images were raveled to an one-dimensional array and PCA was performed; 32 features were kept for 85% of variance expalained. Then, the PCA processed image array was used to train a logistic regression classifier for 40 epochs, at a learning rate of 0.0005 and L_2 regularizaiton strength of 0.0005. The testing performance of the logistic regression classifier is shown in Figure 19:

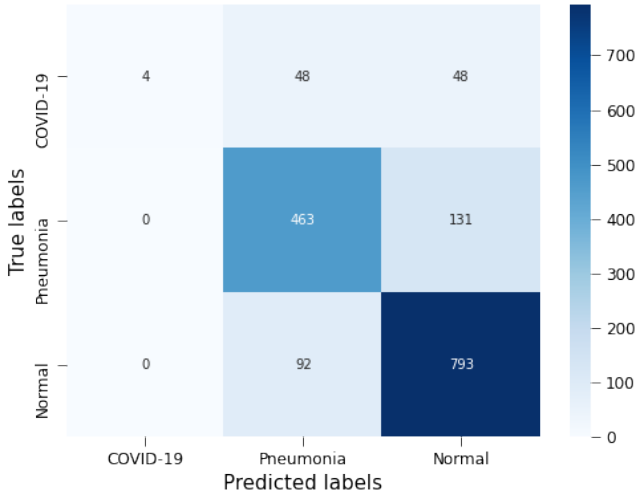


Figure 19: Test Set Performance of the Logistic Regression Model

The logistic regression classifier achieved an overall accuracy of 79%. For Normal and Pneumonia classes, the accuracies were 79% and 89%, but when classifying COVID-19 cases, the accuracy was only 4%. The classifier simply classified nearly all COVID-19 images as Pneumonia or Normal.

This indicates that the classifier did not learn much from the COVID-19 images, and was not punished much by the loss function from doing so. This is likely a result from the skewed training dataset: out of the 13,897 training images, there were 473 COVID-19 images, 5,458 Pneumonia images, and 7,966 Normal images. COVID-19 images were only 3.38% of the training data. The logistic regression classifier was a simple classifier, and with such a skewed training dataset, it was logical to use most of its capacity to learn from the majority classes and ensure overall performance, while sacrificing the minority class.

In summary, the benchmark study provided a baseline goal of 79% overall accuracy. The simple logistic regression classifier could distinguish Pneumonia and Normal classes well, but did not learn many features from the COVID-19 class. The result indicated the problem of skewed training dataset; if more complex models still could not learn features from COVID-19 cases, it might be necessary to adjust the composition of training data. The subsequent models developed for the project should at least have an overall accuracy above 79% and effort should be made to ensure that the final model performs equally well on all classes of images.

4.1.2 Random Forest

For the binary classification, the COVID-19 cases were considered as a positive class (class label = 1) while the other cases (normal and pneumonia) were considered as a negative class (class label = 0). The same procedure was conducted in the testing dataset.

	Training set	Testing set
Positive (Covid-19)	468	100
Negative (Normal and Pneumonia)	13,424	1,479
Total	13,892	1,579

Figure 20: Overview of dataset

All chest X-ray images, in the training and testing dataset, were resized to 480 x 480. Figure 15. summarizes the data distribution in the training and testing sets. In this study, DenseNet121 was implemented via the deep learning library Keras (http://keras.io/) v2.2.4 with Tensorflow backend. Its model weights were obtained through the default setting of Keras. Also, Random Forest was implemented using the latest version (v0.23.1) of the machine learning library sci-kit-learn. The number of trees was set to 100. All other parameters were set to default unless further specified. The deep features were obtained via using a GTX 1070 Ti graphics card. The random forest model was trained on a laptop computer with 8.0 GB RAM, and an Intel Core i5 @1.40GHz CPU (4 Cores). These settings were used in all experiments unless further specified.

For performance evaluation, the random forest model was evaluated on the testing set containing 1,579 chest x-ray images. Metrics including true positive (TP) denotes the number of COVID images identified as COVID, true negative (TN) denotes the number of Non-COVID images identified as Non-COVID(Normal+Pneumonia), false positive (FP) denotes the number of COVID images incorrectly identified as Non-COVID, false negative (FN) denotes the number of Non-COVID incorrectly identified as COVID. Results like accuracy, precision, recall (sensitivity), specificity, F1 score, and area under receiver operating characteristics (AUC) curve were computed. The detailed result is in the following table.

	Covid (Predicted)	Non-Covid (Predicted)
Covid (Actual)	51	49
Non-Covid (Actual)	0	1479

Figure 21: Confusion Matrix

	Random Forest (100 trees)
Overall Accuracy	0.95
Precision	1.00000
Recall (sensitivity)	0.51000
Specificity	1.00000
F1 score	0.67550
AUC	0.97808

Figure 22: Final result

4.1.3 The Simple CNN Model

A simple CNN model was developed for this project. The number of layers and complexity of the model was limited to avoid overfitting and long evaluation time. According to the sensitivity analysis on number of convolutional layers, 3 convolutional layers would be ideal, as overfitting occurred after 4 layers. After some adjustments on the fully connected layers, the structure was finalized to 4 convolutional layers and 3 fully connected layers, shown in Figure 23.

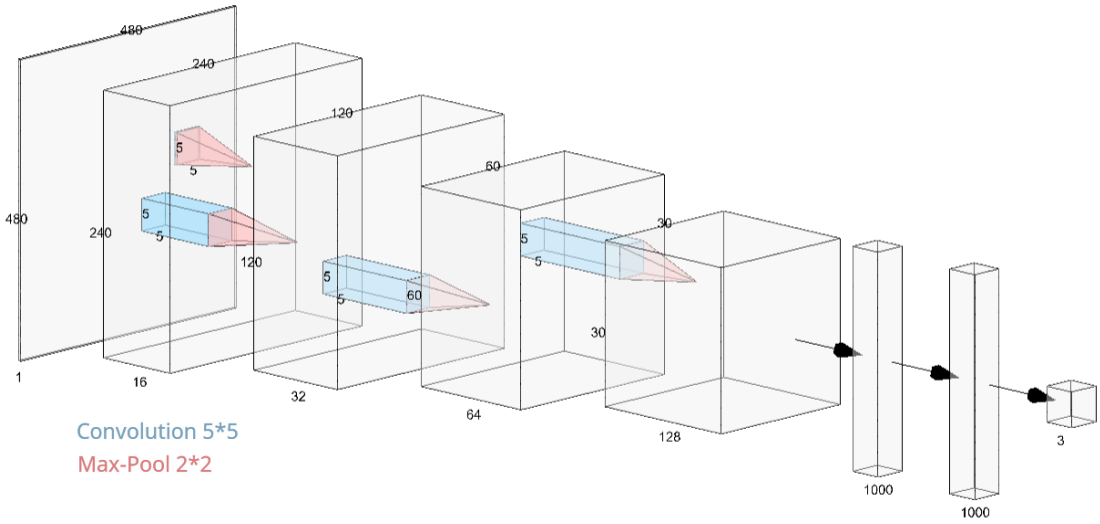


Figure 23: 4-Layer Simple CNN Model

The model was first trained on the original, skewed training dataset for 85 epochs. As mentioned in Section 3, data augmentation was applied to all training images. The testing performance of the model is shown in Figure 24. The model performed well on Pneumonia and Normal images, with 87% and 94% accuracy, but relatively poorly on COVID-19 images, with only 76% accuracy. The result shows that, compare to the benchmark study, the simple CNN model was able to learn from the COVID-19 images, but still preferred to classify images as Normal or Pneumonia because of the skewness of training data. Consequently, the COVID-19 class had a false negative

rate of 24%; the model trained with original training set was ineffective in detecting COVID-19 infection.

To overcome the skewness of data, random oversampling was applied before training images were loaded, such that COVID-19, Pneumonia, and Normal classes all composed roughly 33% of the training data. The testing performance of the model trained with oversampling is shown in Figure 25. The model achieved an overall accuracy of 90%; the model performed better on COVID-19 images, with 82% accuracy. However, the performance on Pneumonia and Normal classes were still much better, with 89% and 92% accuracy.

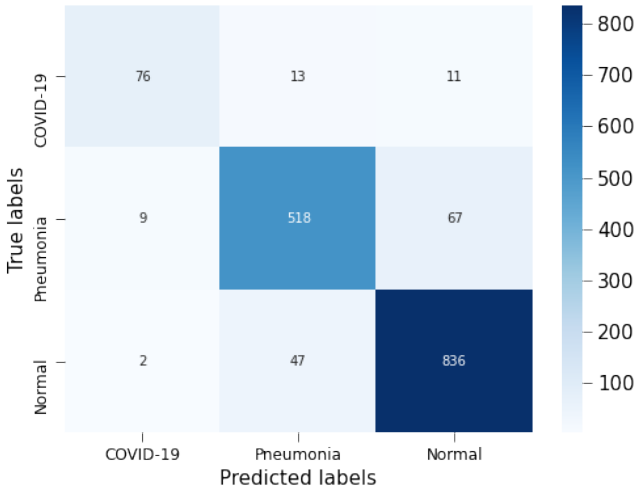


Figure 24: Test Set Performance of Model Trained without Oversampling

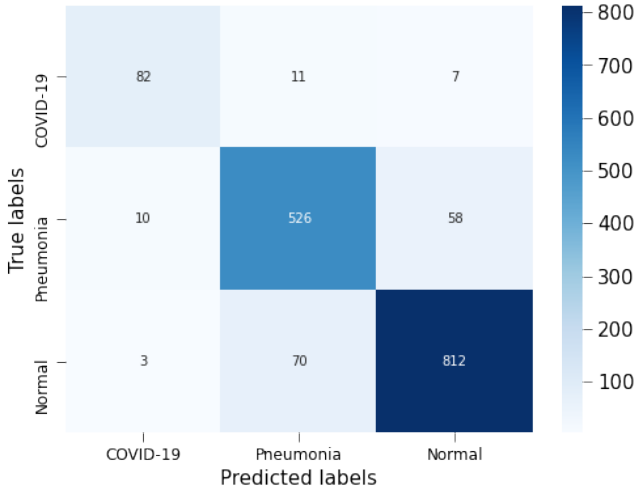


Figure 25: Test Set Performance of Model Trained with COVID-19 Oversampling

Figure 26 and 27 show the learning curves of model trained with and without oversampling. Under both training scenarios, the loss and accuracy improved over epochs and converged after around 50 epochs. Figure 28 and 29 show how the training and testing COVID-19 accuracy changed over training epochs; the COVID-19 accuracy was high in the beginning but dropped as training proceeded, and converged at a lower value after round 50 epochs.

According to the learning curves, the model under both training scenario had achieved an optimum after 50 epochs, and no overfitting occurred with further training. This could be an indication that the simple CNN model had reached its capacity after 50 epochs, such that with further training it could not achieve a better performance even on the training set.

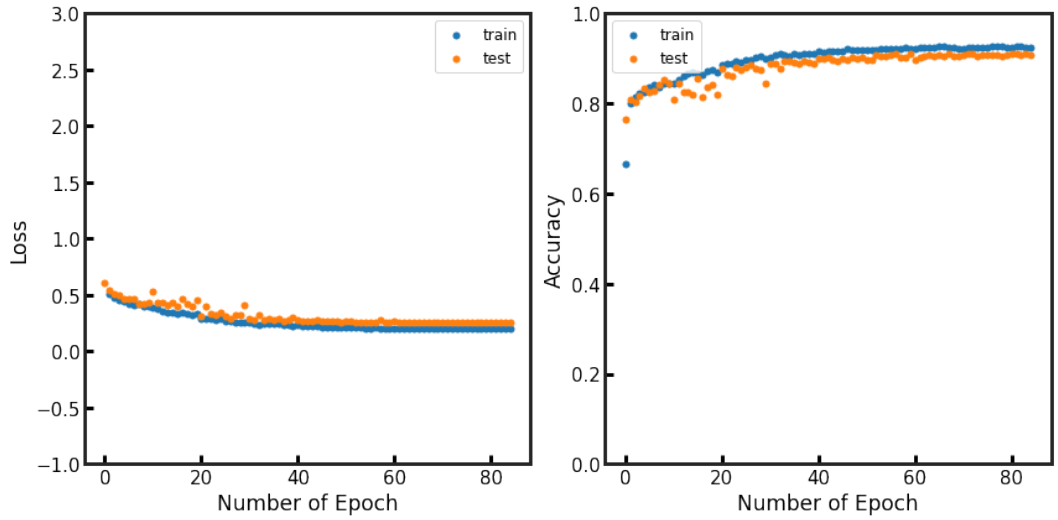


Figure 26: Learning Curves without Oversampling

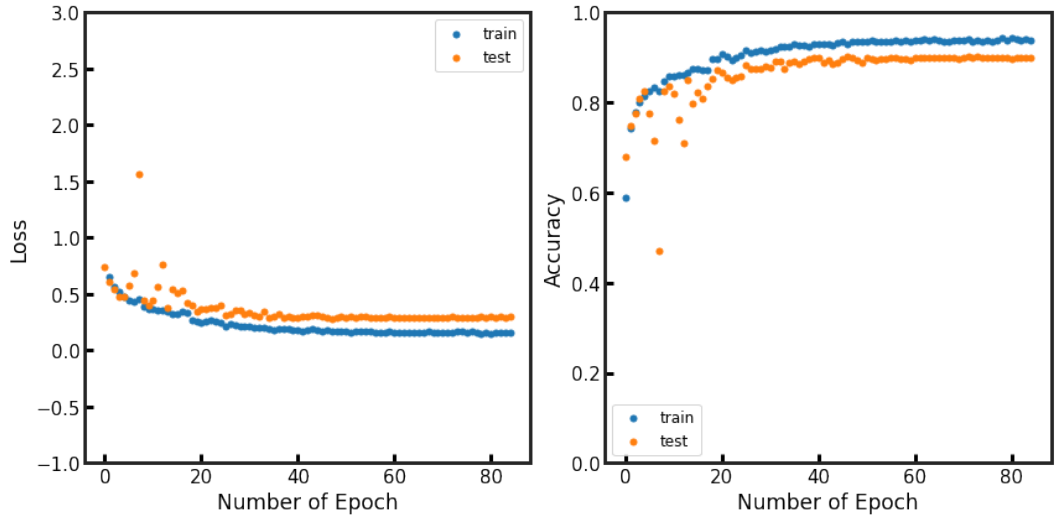


Figure 27: Learning Curves with COVID-19 Oversampling

In addition, the model achieved better overall accuracy after lowering its COVID-19 accuracy, according to Figure 28 and 29. Instead of saying that the model was overfitted on COVID-19 cases in the beginning of training, it is more reasonable to conclude that the model did not have enough capacity to maintain its performance on the COVID-19 class while improving its performance on Normal and Pneumonia classes. The limited capacity of the simple CNN model resulted it to compromise COVID-19 accuracy and adapt more to the Normal and Pneumonia classes; as a result, the COVID-19 accuracy dropped roughly 10% when the model reached an optimum.

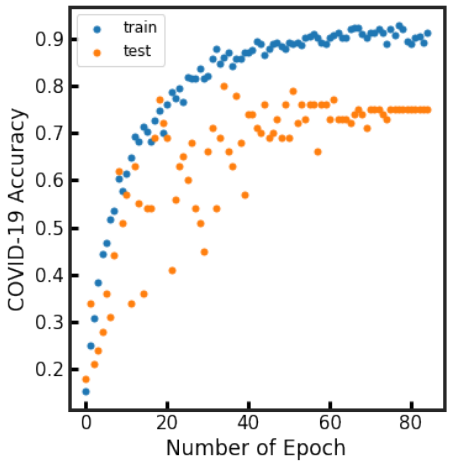


Figure 28: COVID-19 Accuracy Over Epochs without Oversampling

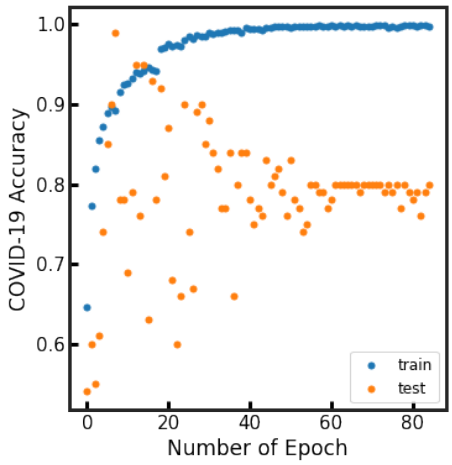


Figure 29: COVID-19 Accuracy Over Epochs with COVID-19 Oversampling

In conclusion, the 4-layer simple CNN model was not a good model for the classification problem. Despite that it achieved a 90% overall accuracy, its performance on COVID-19 images was not satisfactory. Even with COVID-19 oversampling, the model still had a 18% false negative rate in classifying COVID-19 images. This is likely resulted from the limited capacity of the simple 4-layer architecture; if the project were to move on with this model, its complexity should be increased by increasing the number of layers, adjusting the size of kernels and fully connected layers, and perhaps adding other components, such as residual blocks. However, due to time limitations, the project eventually took the approach of transfer learning by utilizing existing complex models.

4.1.4 Transfer Learning

Developers also faced with the huge time cost problem caused by lacking computing resources. It always took a few days to get a model and test it on the test set. Therefore, the strategy of model training was changed to use the transfer learning method.

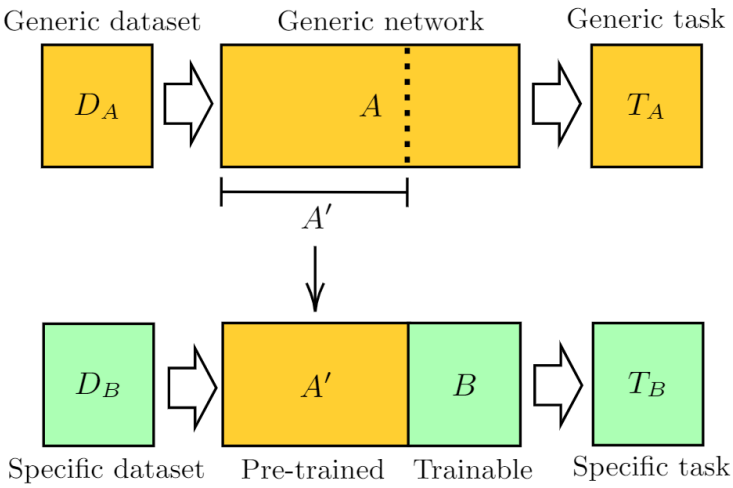


Figure 30: Transfer Learning [18]

Transfer learning can be considered there are two task systems A and B where task A has massive data resources and has trained a model using these data, but it is not the target task. Whereas task B is the developer's target task, but the amount of data is small and extremely precious which is difficult to obtain, such as the COVID-19 image data in this project. Thus, this project is a typical application scenario of transfer learning. Fig. 30 shows the basic concept of transfer learning.

It is worth noting that the new task system (chest X-ray classification) and the old task system (ImageNet classification) must have certain similarities in terms of data, tasks, and models. For example, if a trained speech recognition system was transferred to the image recognition system of the radiology, the result will not be too good.

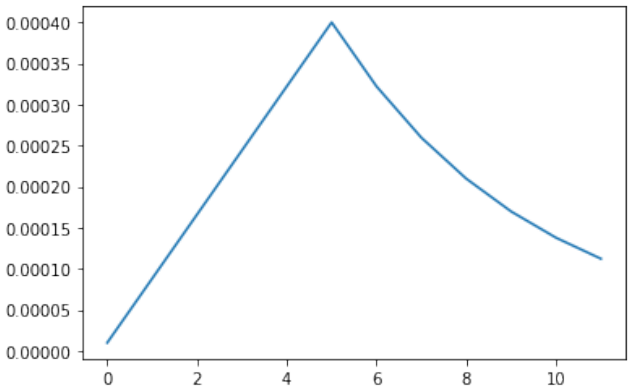


Figure 31: Learning rate in transfer learning

ResNet-50, MobileNet, DenseNet-201, and Xception model pre-trained on the ImageNet dataset were utilized as a generic network model for this project. Moreover, in order to make the model more suitable for this project's data set, a pooling layer and a dense layer were added to the pre-trained model. Then, these models were retrained for twelve epochs with the COVIDx dataset after implementing data augmentation. In order to keep the original weight of the model as much as possible but also let the model fit the COVIDx dataset at the same time, the learning rate is set very low at the beginning, as the number of epoch grows, the learning rate gradually increases, and then decreases. Fig. 31 shows the changing trend of learning rate in transfer learning.

The Xception model presents the best prediction result, about 95% accuracy on test set. Fig.32 and Fig.40 show the confusion matrix of the pre-trained Xception model on test set without and with transfer learning respectively. Obviously, the model after implementing transfer learning shows better prediction accuracy on the test set with three-hundred chest X-rays images. Other 3 models also provide good results shown as Fig. 34, 35, and 36, but their prediction results are still slightly inferior to Xception model. Therefore, the Xception model was finally selected as the one packaged into the system prototype.

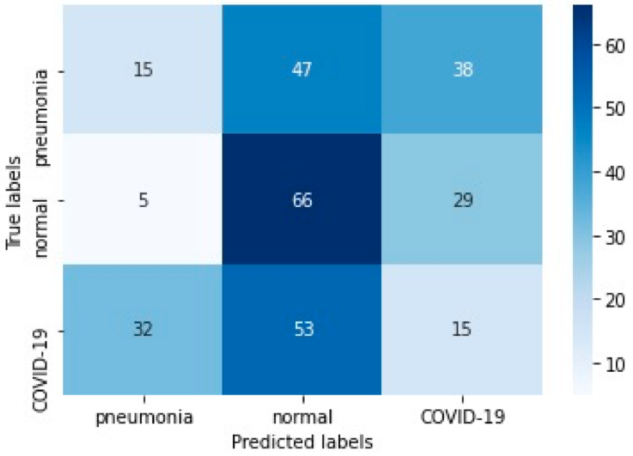


Figure 32: Pretrained Xception on test set without transfer learning

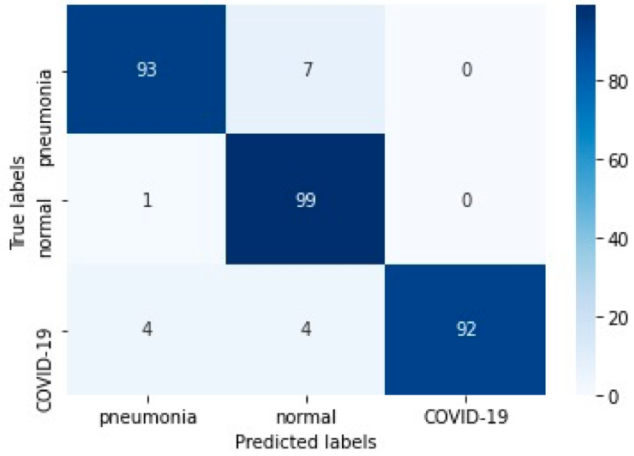


Figure 33: Pretrained Xception on test set with transfer learning

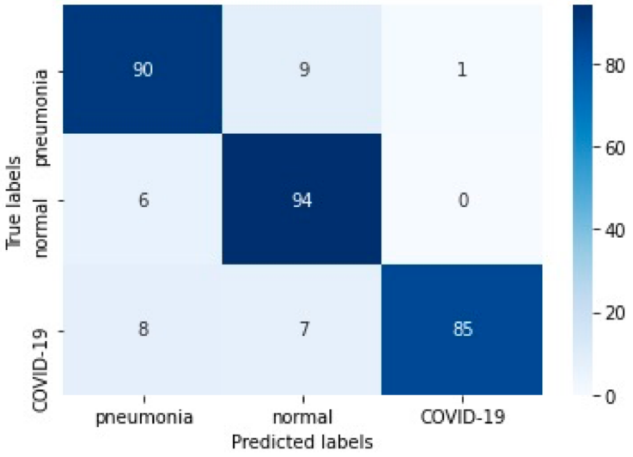


Figure 34: Pretrained ResNet-50 on test set with transfer learning (Accuracy 89.7%)

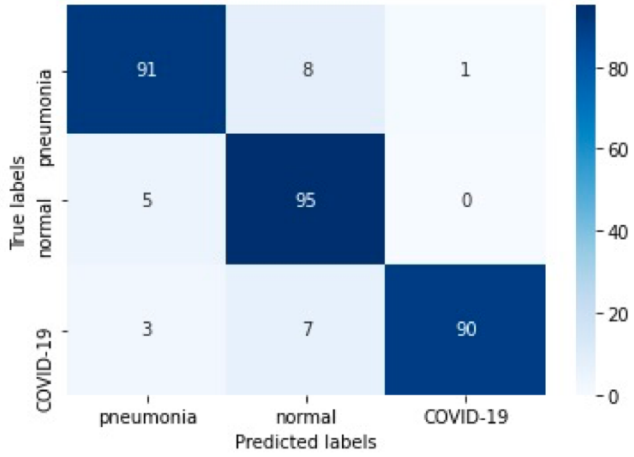


Figure 35: Pretrained DenseNet201 on test set with transfer learning (Accuracy 92.0%)

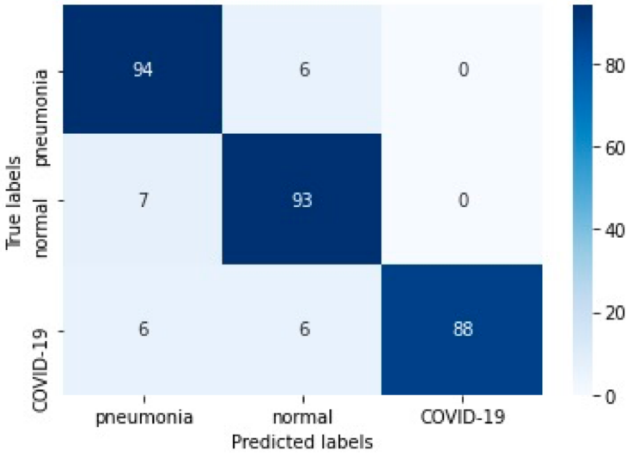


Figure 36: Pretrained MobileNet on test set with transfer learning (Accuracy 91.7 %)

4.2 Technical Challenges and Resolutions

4.2.1 Computational Power

Training deep learning models involves large amount of computation and can be very time consuming on regular computers. One solution is to move the calculation from CPU to GPU, which can accelerate the training process significantly, as GPU supports parallel computation. However, the laptops used by the members of this project did not support parallel computing.

In order to solve this challenge, a cloud computational platform, *Google Colaboratory* [8] was used to train the models. The GPU on Google Colab was able to speed up the training process by more than 5 times.

4.2.2 Training Bottlenecked by Data Loading

However, moving the training process to Google Colab introduced a new technical difficulty. When the model was first trained on Google Colab, the process did not accelerate significantly. It was later discovered that the training process was bottlenecked by data loading. The training images on Google Colab had to be loaded from Google Drive; retrieving image files and going back and forth between Google Drive and Google Colab was time consuming.

This problem was resolved by saving all image files into a HDF5 file, or a The Hierarchical Data Format version 5 file. Hierarchical Data Format is designated to save and organize large amounts of data and can be accessed much more efficiently. This allowed for faster data loading between Google Drive and Google Colab.

It is important to note that data loading was reasonably fast when images files are retrieved from local directories. Therefore, the users will not need to save their X-ray scans into HDF5 files when using the model; evaluation will not be slowed down if all data were retrieved locally.

4.2.3 Imbalanced Training Set Classes

As the benchmark study and simple CNN model show, the system's COVID-19 sensitivity is heavily affected by the imbalanced dataset. Even with more complex models used in transfer learning, the models still prefer to classify more cases as Pneumonia or Normal due to the scarcity of COVID-19 cases in training data.

To solve this problem, oversampling was applied on the training data. To ensure the that the system was sensitive to COVID-19 but not overfitted on COVID-19, the class composition was adjusted to 33% each by oversampling. This not only increased the system's sensitivity to COVID-19 significantly, by nearly 10%, but also maintained the system's good performance on Pneumonia and Normal classes.

4.2.4 Overfitting

The last technical challenge encountered in the project is overfitting. Overfitting occurred when complex models were used during transfer learning. For example, ResNet-50 could achieved 99% overall accuracy on training set, but only 86% overall accuracy on testing set. Overfitting was expected to happen because the dataset used for training was relatively small.

The overfitting problem can be improved by using more training data. To generate more training data, data augmentation was applied when training images were loaded. Two random transforms, horizontal flip and 20 degree rotation, were applied to each image at the probability of 0.50. More training epochs were run to expose the model to additional images created by data augmentation.

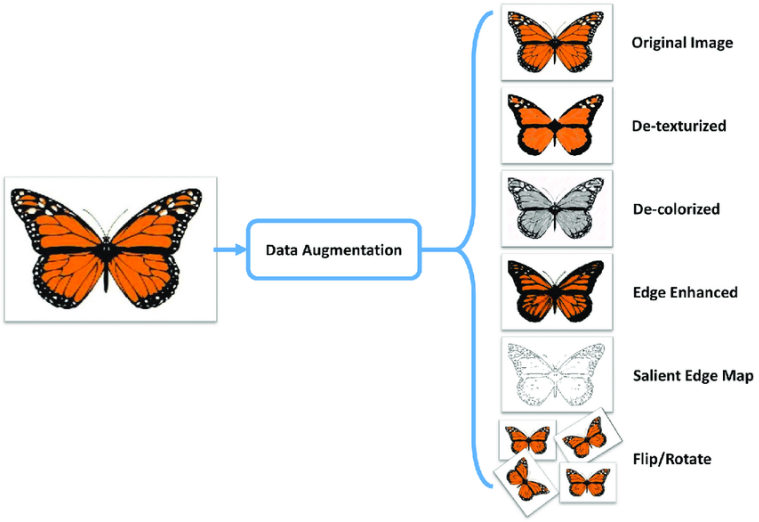


Figure 37: Data augmentation method [5]

4.3 Design Prototype

Table 1 below summarizes the testing performance of each model developed for the project. Modified Xception met all design goals and had been chosen for the prototype.

Table 1: Testing Performance of Each Model

Model	COVID-19 Accuracy	Pneumonia Accuracy	Normal Accuracy	Overall Accuracy
Benchmark Study	0.04	0.78	0.90	0.80
Random Forest	0.51	N/A	N/A	0.95
Simple CNN	0.82	0.86	0.92	0.90
Modified Xception	0.92	0.93	0.99	0.95

The architecture of Modified Xception is shown in the below figure; this is the same architecture that will be in the final design.

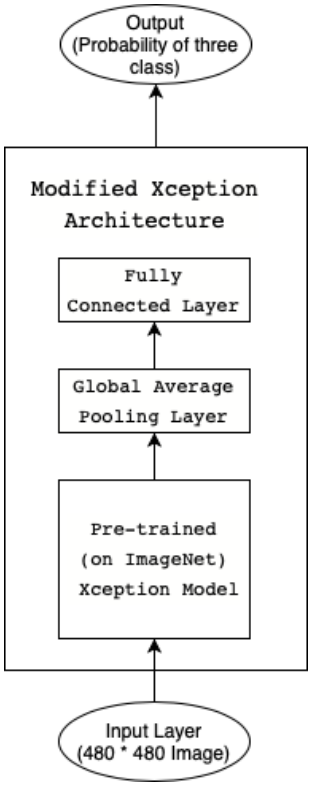


Figure 38: Modified Xception Architecture

The main difference between the prototype and the final design is system output. In the final design, system outputs are class probabilities, class diagnosis, and a list of uncertain cases, the user will have the option to view some or all of the outputs. In the prototype, however, the system only reports class probabilities and class diagnosis. In addition, no user interface is wrapped around the prototype. The below flow chart shows how the prototype works:

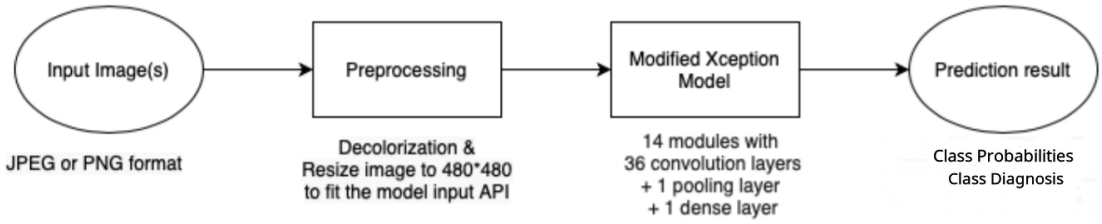


Figure 39: Prototype Work Flow

As mentioned in Section 3.7, the prototype has not been trained on adequate amount of data and cannot be trusted to make diagnosis independently. For a final design that can be released into the market, the system needs more training. The prototype was trained on the COVIDx dataset downloaded at the beginning of the project in May; however, the COVIDx dataset is constantly growing with more and more COVID-19 cases. Training the system again with the current COVIDx dataset may prepare it better for the market.

Moreover, the sensitivity analysis of input image noise in Section 5.4 showed that the prototype was prone to the effect of noise(random color or shade variations) in images. The robustness of final design should be improved by adding noisy images to training data.

5 Design testing and progression

5.1 Important Objectives

The success of the design prototype was evaluated by its classification accuracies, how much it could reduce radiologists' workload, and its robustness and reliability. The three main objectives are summarized as below:

1. **90% COVID-19 accuracy**
2. **90% overall accuracy**
3. **An ambiguity rate smaller than 25%**

Despite that the prototype met all three main objectives, testing also revealed some unexpected problems.

5.2 Testing Procedures

This project was tested on a testing set from the COVIDx dataset. There are 1579 images in total in the testing set. Analyses were mainly done from the following aspects: confusion matrix analysis to assess the prototype's accuracy, ambiguity measurement to see how much the prototype could share radiologists' workload, and input noise sensitivity analysis to test the prototype's robustness.

5.3 Confusion Matrix Analysis

In the analysis phase, confusion matrix was used to analyze the overall system performance Since the system is a multi-class classifier, some performance measures such as sensitivity and specificity will be calculated with micro-averaging. Below are some important performance measures that will be calculated, where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{14}$$

$$Specificity = \frac{TN}{TN + FP} \tag{15}$$

Besides overall accuracy, the sensitivity and specificity on COVID-19 infection are also important parameters. Sensitivity is the percentage of postivie cases that are correctly detected, and specificity is the percentage of negative cases that are correctly classified. It is desired that the

system's COVID-19 sensitivity and specificity are both greater than 0.90. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The target value of precision is also 0.90.

Moreover, F1 score is a measure of performance of the model classification ability. The F1 score is the weighted average of precision and sensitivity, therefore, this index could take both false positives and false negative into account.

$$F1 - Score = 2 \cdot \frac{precision * sensitivity}{precision + sensitivity} \tag{16}$$

Another indicator which is not affected by unbalanced data set is the Matthews correlation coefficient (MCC) score, shown in equation 17. Better MCC score indicates good performance in all four confusion matrix categories (TP, TN, FP, FN).

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{17}$$

Additional performance measures that will be used for evaluation and their target values can be found on the QFD chart.

Figure 18 shows the confusion matrix of the chosen model Xception with transfer learning.

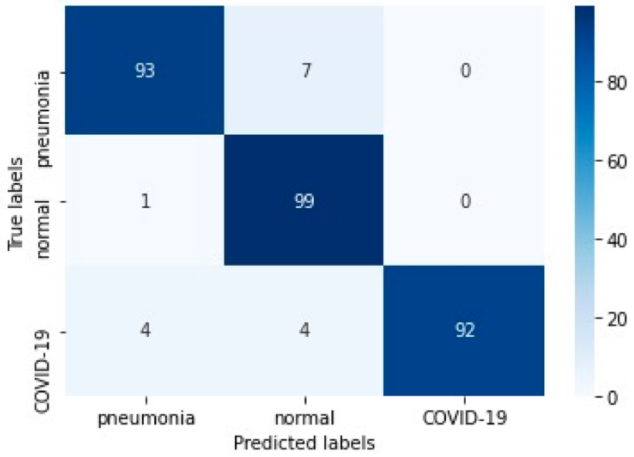


Figure 40: Pretrained Xception on test set with transfer learning

Table 2: Accuracy of Classification Performance

Performance	Accuracy	Precision	Recall	F1-score	AUC
Xception	0.951333	0.951425	0.951333	0.95115	0.947887

The overall accuracy of this project is 95%, which means 95% of the predictions got right among all the testing samples. However, the score of accuracy cannot be a good indicator when the dataset is imbalanced. So in order to better examine, precision and recall scores are also calculated. The precision score for this project is also 95%, which is a good indicator of the

results that are relevant. 95% of the correctly labelled ones are relevant. And the recall score is 95%, which means 95% of total relevant results correctly classified by this model. In addition, the F1 score of this model is 95%, and this score showed that this project have low false positives and low false negatives, which correctly identified the majority cases. The AUC score for this model is 94.8%, which shows that it has a high capability of distinguish between Covid-19, Normal and pneumonia.

5.4 Ambiguity Measurement

As mentioned in Section 2, ambiguity rate measures the number of cases that have uncertain diagnoses, that is, their highest class probability is below 75%. The final system aims to act as a prescreen for radiologist and be able to make reliable diagnoses independently; the system is designed such that it only calls for radiologist assessment for uncertain diagnoses. Out of 1,579 testing cases, only 17 were uncertain. Therefore, the ambiguity rate of the prototype is 1%.

This may look like a good result at first glance, because radiologists only need to assess 1% of the cases. However, the prototype has an overall accuracy of 95%. This indicates that at least 4% of the incorrect diagnoses were made with high certainty, and would not be reported for radiologist assessment.

5.5 Input Noise Sensitivity Analysis

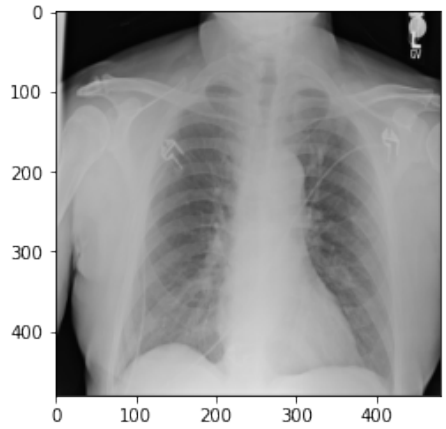


Figure 41: Original Chest X-Ray Scan

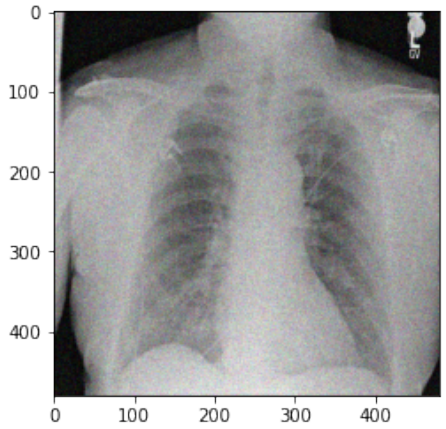


Figure 42: Noisy Chest X-Ray Scan

Sensitivity analysis was done to examine whether color variation in input images affect classification performance. Gaussian noise is commonly seen on digital images, and it causes random color variations. Although human eyes can still recognize shapes and objects when Gaussian noise is present, they might affect the judgement of the model. For the sensitivity analysis, Gaussian noise was added to the testing set. The model evaluated the noisy testing set and the performance was

compared with that on the original testing set. Figure 43 shows the model's performance on the noisy testing set.

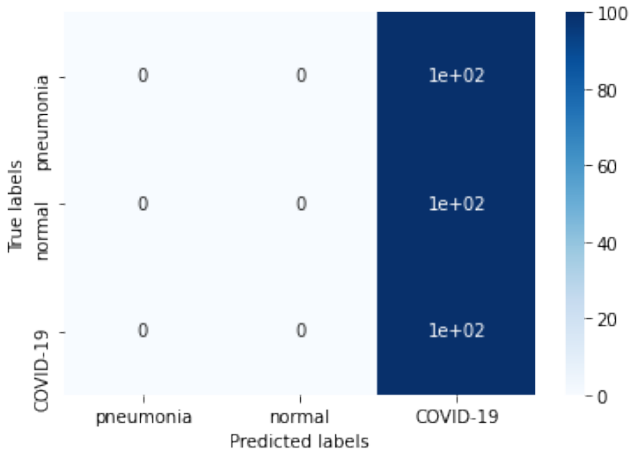


Figure 43: Sensitivity confusion matrix

After Gaussian noise was added, all images were classified as COVID-19 class. This result indicates that the model is highly susceptible to the effect of color variations. Although input images were converted to grayscale during preprocessing, the shade variations caused by Gaussian noise also affected the model's judgement.

Deep learning models are often considered "blackboxes" because it is not known for sure what features models choose to learn. For image classifications, the features can be edges, textures, colors, and etc. The effect of Gaussian noise on the modified Xception model indicates that the model classifies images based on features that involve colors or shades. Therefore, for the model to function properly, it is important to control the input images' quality such that they are not noisy. Or, alternatively, the model should be retrained with noisy images such that it can learn features that are resistant to color and shade variations.

5.6 Other Testing

To assess the prototype's practicality, the evaluation speed of the prototype was assessed. The prototype was removed from the cloud computation platform and moved to a standard laptop equipped with regular CPU (Intel Core i7 8th Gen) and did not allow for parallel computation. Images for evaluation was loaded locally in the form of PNG or JPEG, instead of HDF5. It was discovered that the prototype could run smoothly, and loading and evaluation of local image files was at a rate of 5 seconds per image.

5.7 Discussion

The prototype achieved a high overall test accuracy (95%) and AUC score (94.8%), and was effective in COVID-19 detection, with 92% sensitivity. The results showed that the prototype has the potential to be used as a supplement in clinical decision making. In addition, evaluation speed for one X-ray scan is 5 seconds, which proves the prototype's efficiency.

However, at this early stage of development, the prototype can only be used as a source of reference for radiologists instead of making independent diagnosis. The ambiguity rate lower than overall error rate poses a problem that many of the incorrect diagnoses would not be reported for human assessment. The reporting mechanism designed for the system assumes all diagnoses made with high certainty are correct; and the results indicate that this assumption could be problematic, as incorrect diagnoses made with high certainty would not be captured by the system. It is not known why the prototype made most incorrect diagnoses with high certainty; perhaps this problem can be solved by training the prototype on adequate amount of data. However, it may be necessary to find a more concrete criteria for capturing possibly incorrect cases.

6 Recommendations

6.1 Continuous Training and Update

As mentioned in previous sections, the prototype was trained on a relatively small dataset. In order for a deep learning model to make reliable, independent diagnoses, extensive training is required. In addition, even if the system is already released on market, it also needs to keep up with the evolution and mutation of illnesses. Therefore, it is necessary to train and adjust the model continuously with new available data, for both continuous development of the project or updating the system on market.

6.2 Better Criteria for Human Assessment

The biggest problem of the prototype is its criteria for calling for radiologist assessment. As the testing results showed, the prototype's diagnosis certainty was not a good criteria for capturing incorrect diagnoses. Despite that it may become more valid as the prototype trains on more data, finding a more concrete criteria may be a better option.

6.3 Improvement on Model Robustness

The sensitivity analysis on input noise in Section 5.4 reflected the problem that the prototype is highly susceptible to the effect of Gaussian noise (random color variations) in input images. To improve robustness, the prototype should be retrained on noisy training data.

6.4 Effect of Other Noise in Input Images

For further development of the project, the effect of other types of noise in input images on system performance should be examined. For example, the cropping and distortion of input images can potentially affect evaluation performance. It is important to see whether these noises affect system performance greatly, and if yes, they should also be added to the training images.

6.5 Add-on to Radiography Machines

One way to minimize noises in input images is to control the source of the images. Instead of a computer program, it might be better to create a new version of the system as an add-on to radiography machines, such that it can get first-hand, high-quality input images, minimize unwanted noises, and save some time by evaluating the scan as soon as the scan is created.

6.6 Adaptation to Other Diseases

The pandemic of COVID-19 will end eventually, and to elongate the system's product life cycle, it may be helpful to tweak the system towards diagnosing other common diseases. This will also require continuous training with available data and updating the model.

7 Conclusion

This project is considered an overall success. The prototype met most project objectives and was able to detect COVID-19 infections as well as non-COVID-19 infections effectively. The prototype could detect COVID-19 infection with 92% accuracy and non-COVID-19 infection with 93% accuracy. This result is better than two state-of-the-art systems, RT-PCR test and radiologist assessment. Based on the results, it is believed that the prototype has the potential of serving as an effective diagnostic prescreen with further training and adjustment.

However, the system is designed to capture possibly incorrect diagnoses by its diagnosis certainty. Testings show that the prototype could make incorrect diagnoses with high certainty (at least 4 out of 5% incorrect cases were certain). As a result, these cases would not be reported for human assessment. It may be necessary to change the reporting criteria if the system continues to make incorrect diagnoses with high certainty after further training. In addition, sensitivity analysis on Gaussian noise indicated that the system is prone to the effect of noisy images.

The prototype was trained on a relatively small dataset. Therefore, extensive training, adjustments, and examination on input noises are required to improve the system's robustness and reliability. Perhaps with further development, the system can overcome the above problems and become market-ready.

8 Project Management

8.1 Parallel Model Development

During the design process, parallel model development allowed for each project member to take different approaches to the design problem and eventually found a good solution. After spending some time training a simple CNN model but did not get good results, it was decided that each member should try a different approach. Parallel model development is thought to be a good decision made during the design process, because training deep learning model can be very time-consuming, and doing so parallel can save a lot of time; taking multiple approaches also helped the project avoid going into one dead-end.

8.2 Anticipation and Challenges

The technical challenge of limited computation power was expected and was readily resolved by purchasing cloud computational resources. The challenge of bottle-necked data loading, however, was not anticipated. It took some time and research to resolved. The time used was proven worthy; if the training time were bottle-necked, model development would take significantly longer. Other smaller technical challenges such as occasional RAM overflows were all resolved by online research.

8.3 Usage of Technical Skills

Many of the technical challenges were resulted from the members' lack of experiences in machine learning. The members' technical skills improved as the design progressed; none of the member had the experience in deep learning model development, but by the end of the project, each member has created a functioning classifier.

8.4 Communication

Communication during the design process could be improved. Despite that technical challenges were often solved together and knowledge of useful techniques were exchanged, with the parallel development of models, each member was not fully aware of the details of others' work.

8.5 Time Management

Time management of the project was overall satisfactory, but less time could be spend on training models and tuning hyperparameters. The extra time spent did not improve the performance further, but resulted the lack of time to make the coding structure cleaner and more organized.

8.6 Recommendations for Future Projects

For future projects, time management should be valued more. A plan with clear deadline of each phase of the project (such as a Gantt Chart) should be made to avoid spending too much time on one phase of the project while sacrificing the time for the other. In addition, if each member takes different approaches and design progresses in a parallel manner, it is very important to communicate periodically and make sure that the members are fully aware of each others' progress and plans.

References

[1] Artificial Intelligence (AI) In China: The Amazing Ways Tencent Is Driving It's Adoption. (n.d.). Retrieved June 10, 2020, from *https://www.forbes.com/sites/bernardmarr/2018/06/04/artificial-intelligence-ai-in-china-the-amazing-ways-tencent-is-driving-its-adoption/5d009aec479a* pages

[2] Bai, H. X., Hsieh, B., Xiong, Z., Halsey, K., Choi, J. W., Tran, T. M. L., Pan, I., Shi, L.-B., Wang, D.-C., Mei, J., Jiang, X.-L., Zeng, Q.-H., Egglin, T. K., Hu, P.-F., Agarwal, S., Xie, F., Li, S., Healey, T., Atalay, M. K., and Liao, W.-H. (2020). Performance of radiologists in differentiating COVID-19 from viral pneumonia on chest CT. Radiology, 200823. *https://doi.org/10.1148/radiol.2020200823* pages 1

[3] Borgonovo, E., and Plischke, E., 2016. Seneitivity analysis: A review of recent Advances. *European Journal of Operational Research 248(3)*, 869-887. pages 19

[4] Bozsolik, T., 2018. National Institutes of Health Chest X-Ray Dataset. *https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data* pages

[5] Data Augmentation Increases Accuracy of your model., Medium., Retrieved July, 2020 from: *https://medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722* pages 5, 36

[6] Huang, G., Liu, Z., Maaten, L., and Weinberger K., 2017. Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017-06*, p.2261-2269. pages 5, 13

[7] FDA Cleared AI Algorithms., Data Science Institute. Retrieved May, 2020 from: *www.acrdsi.org/DSI-Services/FDA-Cleared-AI-Algorithms* pages

[8] Google Colab Introduction., Google., Retrieved July, 2020 from: *https://colab.research.google.com/notebooks/intro.ipynb* pages 35

[9] Health, C. for D. and R. (2020). EUA Authorized Serology Test Performance. FDA. *https://www.fda.gov/medical-devices/emergency-situations-medical-devices/eua-authorized-serology-test-performance* pages 1

[10] He, K., Zhang, X., Ren, S., and Sun J., 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016-06*, p.770-778. pages 5, 12, 13

[11] He, J.-L., Luo, L., Luo, Z.-D., Lyu, J.-X., Ng, M.-Y., Shen, X.-P., and Wen, Z. (2020). Diagnostic performance between CT and initial real-time RT-PCR for clinically suspected 2019

coronavirus disease (COVID-19) patients outside Wuhan, China. Respiratory Medicine, 168, 105980. *https://doi.org/10.1016/j.rmed.2020.105980* pages 1

[12] Kucirka, L. M., Lauer, S. A., Laeyendecker, O., Boon, D., and Lessler, J. (2020). Variation in False-Negative Rate of Reverse Transcriptase Polymerase Chain Reaction–Based SARS-CoV-2 Tests by Time Since Exposure. Annals of Internal Medicine. *https://doi.org/10.7326/M20-1495* pages 1

[13] Kingma, D. P., and Ba, J., 2017. Adam: A Method for Stochastic Optimization. ArXiv:1412.6980 [Cs]. *http://arxiv.org/abs/1412.6980* pages 5, 17, 18

[14] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. *https://doi.org/10.1109/5.726791* pages

[15] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *https://arxiv.org/pdf/1704.04861.pdf*, arXiv:1704.04861. pages 5, 14

[16] Nishiura, H., Linton, N.M., and Akhmetzhanov, A.R., 2020. Serial interval of novel coronavirus (COVID-19) infections, *International Journal of Infectious Diseases, 93* 284-386. pages

[17] Ozturk, T., Talo, M., Yildirim, E., Baloglu, U., Yildirim, O., and Acharya, R., 2020. Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in Biology and Medicine, 121* pages 9

[18] Pennylane.AI: Quantum transfer learning Introduction (n.d.). Retrieved July 25, 2020, from *https://pennylane.ai/qml/demos/tutorial_quantum_transfer_learning.html* pages 5, 32

[19] Python Imaging Library., Python Help Documentation. Retrieved May, 2020 from: *http://omz-software.com/pythonista/docs/ios/PIL.html* pages

[20] Rahman, T., 2020. COVID-19 RADIOGRAPHY DATABASE (Winner of the COVID-19 Dataset Award) *https://www.kaggle.com/tawsifurrahman/covid19-radiography-database* pages

[21] RGB to GrayScale Conversion., TutorialPoint. Retrieved May, 2020 from: *www.tutorialspoint.com/dip/grayscale-to-rgb-conversion.htm* pages

[22] Seshia, S. A., Desai, A., Dreossi, T., Fremont, D. J., Ghosh, S., Kim, E., and Yue, X., 2018. Formal specification for deep neural networks. *In International Symposium on Automated Technology for Verification and Analysis* 20-34. pages

[23] Simonyan, K., and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv:1409.1556 [Cs]. *http://arxiv.org/abs/1409.1556* pages

[24] Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved June 10, 2020, from *http://cs231n.stanford.edu/* pages 10, 11

[25] Tahamtana, A and Ardebilib, A., 2020. Real-time RT-PCR in COVID-19 detection: issues affecting the results. *Expert Review of Molecular Diagnostics, 20(2020)* 454–454. pages

[26] Tencent Miying Launches AI-supported Auxiliary Diagnostic System., Tencent Holdings Limited. Retrieved May, 2020 from: *www.prnewswire.com/news-releases/tencent-miying-launches-ai-supported-auxiliary-diagnostic-system-300855493.html* pages

[27] Wang, L., 2020. COVIDx Dataset. *https://github.com/lindawangg/COVID-Net/blob/master/docs/COVIDx.md* pages

[28] Wang, L., Lin, Z., and Wong, A., 2020. COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images. pages 8

[29] WHO Coronavirus disease (COVID-19) outbreak situation. (n.d.). Retrieved September 9, 2020, *https://covid19.who.int* pages 1

[30] Chollet, F., 2016. Xception: Deep Learning with Depthwise Separable Convolutions. *https://arxiv.org/abs/1610.02357*, arXiv:1610.02357. pages 5, 15, 16, 23

[31] Xu, X., Jiang, X., Ma, C., Du, P., Li, X., and Lv, S., 2020. Deep Learning System to Screen Coronavirus Disease 2019 Pneumonia. *Cornell University, 2002,09334.* pages

[32] Tony, 2019. Understanding Random Forest and How the Algorithm Works and Why it Is So Effective *https://towardsdatascience.com/understanding-random-forest-58381e0602d2* pages 17

[33] Will, 2018. An Implementation and Explanation of the Random Forest in Python *https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76* pages 17

A Appendix: QFD Chart

User Requirements	Manufacture	Professional	End user	Accuracy	Balanced Accuracy	Sensitivity	Specificity	False Positive Rate	False Negative Rate	Precision	False Discovery Rate	F1 Score	Positive Likelihood	Negative Likelihood	Diagnostic Odds Ratio	MCC Score	FM Score	Run Time	Cost per Case	Competition A	Competition B	Competition C
Functional Performance																						
High accuracy	10	10	10	9	9	9	9									9				8.5	9.5	8
Fast results	8	9	9															9		6.5	6	9
Definitive diagnosis	9	9	9	1	3	9	9	9	9	9	9	9	9	9	9					9	9.5	7.5
Reliable and consistent (not affected by time infected)	10	10	10	3	3	3	3					3			3	3		1		6	7	8
Can be updated; resistant to mutation	9	8	7	3															1	1	1	8
Can handle sudden outbreak with large volume of cases	7	7	9															9	3	3	5	8.5
Human Factors																						
Easy to implement	7	7	7																	7	7	8
Easy to operate	6	4	8	1														3		6	6	8
Results are easy to interpret	4	3	7	1														1		7	7	2
Small amount of training required	6	3	7																1	3	3	1
Small room for human error	7	3	9	3	3															8	8	3
Low workload on operators	6	6	8															1		7	7	2
Financial Factors																						
Low initialization costs (laboratory setup and new equipment needed)	7	5	5																1	6.5	6.5	8
Low maintenance and operating costs	4	7	7																9	7	7	7
Low material consumption	3	5	7																9	1	2	9
Low engergy consumption	3	5	6																3	6	6	7
Units																		hour	$USD			
Competition																						
A Nasal/Throat Swab RT-PCR Test (Drive-thru Test)				0.915	0.897	0.794	1.000	0.000	0.206	1.000	0.000	0.885	inf	0.206	inf	0.833	0.891	2~24	51.31			
B Serological Antibody Test (Blood work)				0.973	0.961	0.936	0.987	0.013	0.065	0.967	0.333	0.951	73.9	0.065	1131	0.932	0.951	3~6	45.23			
C Medical Imaging & Radiologist Assessment				0.820	0.821	0.805	0.837	0.163	0.195	0.841	0.159	0.822	4.9	0.233	21	0.641	0.823	1	40-250			
Target Values				0.900	0.900	0.900	0.900	0.100	0.100	0.900	0.050	0.900	9.0	0.111	81	0.800	0.900	1~1.5	45			